

IBM DB2 10.1
para Linux, UNIX y Windows

Consulta de SQL - Volumen 2
Actualizado en enero de 2013



IBM DB2 10.1
para Linux, UNIX y Windows

Consulta de SQL - Volumen 2
Actualizado en enero de 2013



Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice B, "Avisos", en la página 1453.

Nota de edición

Este manual es la traducción del original en inglés *IBM DB2 10.1 for Linux, UNIX, and Windows SQL Reference Volume 2 Updated January, 2013 (SC27-3886-01)*.

Este documento contiene información propiedad de IBM. Se proporciona según un acuerdo de licencia y está protegido por la ley de la propiedad intelectual. La información contenida en esta publicación no incluye ninguna garantía de producto, por lo que ninguna declaración proporcionada en este manual deberá interpretarse como tal.

Puede realizar pedidos de publicaciones de IBM en línea o a través del representante de IBM de su localidad.

- Para solicitar publicaciones en línea, vaya a IBM Publications Center en <http://www.ibm.com/shop/publications/order>
- Para encontrar al representante local de IBM que le corresponde, vaya a la sección Worldwide Contacts de IBM Directory en <http://www.ibm.com/planetwide/>

Para realizar pedidos de publicaciones de DB2 desde DB2 Marketing and Sales, en los EE.UU. o en Canadá, llame al 1-800-IBM-4YOU (426-4968).

Cuando envía información a IBM, está otorgando a IBM el derecho no exclusivo de utilizar o distribuir la información de cualquier forma que considere adecuada sin incurrir por ello a ninguna obligación para con usted.

© Copyright IBM Corporation 2013.

Contenido

Acerca de este manual vii

Quién debe utilizar este manual	vii
Cómo está estructurado este manual	vii
Cómo leer los diagramas de sintaxis	viii
Convenios utilizados en este manual	x
Condiciones de error	x
Convenios de resaltado	x
Documentación relacionada	x

Sentencias de SQL 1

Cómo se invocan las sentencias de SQL	11
Incorporación de una sentencia a un programa de aplicación	11
Preparación y ejecución dinámicas	12
Invocación estática de una sentencia de selección	13
Invocación dinámica de una sentencia de selección	13
Invocación interactiva	13
Utilización de SQL con otros sistemas principales	13
Detección y proceso de condiciones de error y de aviso en aplicaciones de lenguaje principal	14
Comentarios de SQL	15
Compilación condicional en SQL	15
Acerca de las sentencias de control de SQL	18
Referencias a parámetros de SQL, variables de SQL y variables globales	19
Referencias a etiquetas de SQL	20
Referencias a nombres de condición de SQL	20
Referencias a nombres de sentencia de SQL	20
Referencias a nombre de cursor de SQL	21
Designadores de función, método y procedimiento	22
ALLOCATE CURSOR	27
ALTER AUDIT POLICY	29
ALTER BUFFERPOOL	33
ALTER DATABASE PARTITION GROUP	36
ALTER DATABASE	40
ALTER EVENT MONITOR	45
ALTER FUNCTION	50
ALTER HISTOGRAM TEMPLATE	54
ALTER INDEX	56
ALTER MASK	57
ALTER METHOD	58
ALTER MODULE	60
ALTER NICKNAME	69
ALTER PACKAGE	78
ALTER PERMISSION	82
ALTER PROCEDURE (externo)	83
ALTER PROCEDURE (derivado)	86
ALTER PROCEDURE (SQL)	88
ALTER SCHEMA	90
ALTER SECURITY LABEL COMPONENT	92
ALTER SECURITY POLICY	95
ALTER SEQUENCE	99
ALTER SERVER	103
ALTER SERVICE CLASS	106

ALTER STOGROUP	117
ALTER TABLE	122
ALTER TABLESPACE	194
ALTER THRESHOLD	211
ALTER TRIGGER	225
ALTER TRUSTED CONTEXT	226
ALTER TYPE (estructurado)	235
ALTER USAGE LIST	242
ALTER USER MAPPING	244
ALTER VIEW	247
ALTER WORK ACTION SET	249
ALTER WORK CLASS SET	264
ALTER WORKLOAD	270
ALTER WRAPPER	287
ALTER XSROBJECT	289
ASSOCIATE LOCATORS	290
AUDIT	292
BEGIN DECLARE SECTION	296
CALL	298
CASE	307
CLOSE	310
COMMENT	312
COMMIT	323
SQL compuesto	325
SQL compuesto (en línea)	326
SQL compuesto (incorporado)	331
SQL compuesto (compilado)	335
CONNECT (tipo 1)	353
CONNECT (tipo 2)	361
CREATE ALIAS	369
CREATE AUDIT POLICY	373
CREATE BUFFERPOOL	377
CREATE DATABASE PARTITION GROUP	381
CREATE EVENT MONITOR	384
CREATE EVENT MONITOR (actividades)	406
CREATE EVENT MONITOR (historial de cambios)	418
CREATE EVENT MONITOR (bloqueo)	425
Sentencia CREATE EVENT MONITOR (antememoria de paquete)	431
CREATE EVENT MONITOR (estadísticas)	438
CREATE EVENT MONITOR (violaciones de umbral)	451
CREATE EVENT MONITOR (unidad de trabajo)	464
CREATE FUNCTION	470
CREATE FUNCTION (escalar externa)	471
CREATE FUNCTION (tabla externa)	501
CREATE FUNCTION (tabla externa OLE DB)	523
CREATE FUNCTION (derivado o plantilla)	534
CREATE FUNCTION (tabla, fila o escalar de SQL)	549
CREATE FUNCTION MAPPING	567
CREATE GLOBAL TEMPORARY TABLE	572
CREATE HISTOGRAM TEMPLATE	585
CREATE INDEX	587
CREATE INDEX EXTENSION	609
CREATE MASK	616
CREATE METHOD	623

CREATE MODULE	629	GRANT (exención)	1123
CREATE NICKNAME	631	GRANT (privilegios de variable global)	1126
CREATE PERMISSION	645	GRANT (privilegios de índice)	1129
CREATE PROCEDURE	649	GRANT (privilegios de módulo).	1131
CREATE PROCEDURE (externo)	650	GRANT (privilegios de paquete)	1133
CREATE PROCEDURE (derivado)	667	GRANT (rol)	1136
CREATE PROCEDURE (SQL)	673	GRANT (privilegios de rutina)	1139
CREATE ROLE	684	GRANT (privilegios de esquema)	1144
CREATE SCHEMA	685	GRANT (etiqueta de seguridad)	1147
CREATE SECURITY LABEL COMPONENT	688	GRANT (privilegios de secuencia)	1150
CREATE SECURITY LABEL	691	GRANT (privilegios de servidor)	1153
CREATE SECURITY POLICY	693	GRANT (privilegio SETSESSIONUSER)	1155
CREATE SEQUENCE.	695	GRANT (privilegios de espacio de tablas)	1157
CREATE SERVICE CLASS	701	GRANT (privilegios de tabla, vista o apodo)	1159
CREATE SERVER	713	GRANT (privilegios de carga de trabajo)	1166
CREATE STOGROUP	717	GRANT (privilegios de objeto XSR)	1168
CREATE SYNONYM	721	IF	1169
CREATE TABLE	722	INCLUDE	1171
CREATE TABLESPACE	810	INSERT	1173
CREATE THRESHOLD	827	ITERATE	1185
CREATE TRANSFORM	845	LEAVE	1187
CREATE TRIGGER	849	LOCK TABLE	1189
CREATE TRUSTED CONTEXT	865	LOOP	1191
CREATE TYPE	872	MERGE	1193
CREATE TYPE (matriz)	873	OPEN	1206
CREATE TYPE (cursor)	880	PIPE	1212
CREATE TYPE (diferenciado)	883	PREPARE	1214
CREATE TYPE (fila)	891	REFRESH TABLE	1221
CREATE TYPE (estructurado)	896	RELEASE (conexión)	1225
CREATE TYPE MAPPING	921	RELEASE SAVEPOINT	1227
CREATE USAGE LIST	928	RENAME	1228
CREATE USER MAPPING	932	RENAME STOGROUP	1230
CREATE VARIABLE	935	RENAME TABLESPACE	1231
CREATE VIEW	945	REPEAT	1233
CREATE WORK ACTION SET	961	RESIGNAL.	1235
CREATE WORK CLASS SET	971	RETURN	1238
CREATE WORKLOAD	977	REVOKE (autorizaciones de bases de datos).	1240
CREATE WRAPPER	996	REVOKE (exención)	1244
DECLARE CURSOR	998	REVOKE (privilegios de variable global)	1246
DECLARE GLOBAL TEMPORARY TABLE	1004	REVOKE (privilegios de índice)	1249
DELETE.	1018	REVOKE (privilegios de módulo)	1251
DESCRIBE	1029	REVOKE (privilegios de paquete)	1253
DESCRIBE INPUT	1030	REVOKE (rol)	1256
DESCRIBE OUTPUT	1034	REVOKE (privilegios de rutina)	1259
DISCONNECT	1039	REVOKE (privilegios de esquema)	1263
DROP	1042	REVOKE (etiqueta de seguridad)	1265
END DECLARE SECTION	1078	REVOKE (privilegios de secuencia).	1267
EXECUTE	1079	REVOKE (privilegios de servidor)	1270
EXECUTE IMMEDIATE	1087	REVOKE (privilegio SETSESSIONUSER)	1272
EXPLAIN	1090	REVOKE (privilegios de espacio de tablas)	1274
FETCH	1095	REVOKE (privilegios de tabla, vista o apodo)	1276
FLUSH BUFFERPOOLS	1100	REVOKE (privilegios de carga de trabajo)	1282
FLUSH EVENT MONITOR	1101	REVOKE (privilegios de objeto XSR)	1284
FLUSH FEDERATED CACHE	1102	ROLLBACK	1285
FLUSH OPTIMIZATION PROFILE CACHE	1104	SAVEPOINT	1288
FLUSH PACKAGE CACHE	1106	SELECT	1291
FOR	1107	SELECT INTO.	1292
FREE LOCATOR	1110	SET COMPILATION ENVIRONMENT	1296
GET DIAGNOSTICS.	1111	SET CONNECTION.	1297
GOTO	1115	SET CURRENT DECFLOAT ROUNDING MODE	1299
GRANT (autorizaciones de bases de datos)	1117	SET CURRENT DEFAULT TRANSFORM GROUP	1301

SET CURRENT DEGREE	1303
SET CURRENT EXPLAIN MODE	1305
SET CURRENT EXPLAIN SNAPSHOT	1308
SET CURRENT FEDERATED ASYNCHRONY	1311
SET CURRENT IMPLICIT XMLPARSE OPTION	1313
SET CURRENT ISOLATION	1314
SET CURRENT LOCALE LC_MESSAGES	1315
SET CURRENT LOCALE LC_TIME	1317
SET CURRENT LOCK TIMEOUT	1319
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	1321
SET CURRENT MDC ROLLOUT MODE	1323
SET CURRENT OPTIMIZATION PROFILE	1325
SET CURRENT PACKAGE PATH	1328
SET CURRENT PACKAGESET	1332
SET CURRENT QUERY OPTIMIZATION	1334
SET CURRENT REFRESH AGE	1337
SET CURRENT SQL_CCFLAGS	1339
SET CURRENT TEMPORAL BUSINESS_TIME	1341
SET CURRENT TEMPORAL SYSTEM_TIME	1343
SET ENCRYPTION PASSWORD	1345
SET EVENT MONITOR STATE	1347
SET INTEGRITY	1350
SET PASSTHRU	1370
SET PATH	1372
SET ROLE	1374
SET SCHEMA	1375
SET SERVER OPTION	1377
SET SESSION AUTHORIZATION	1379
SET USAGE LIST STATE	1382
SET variable	1385

SIGNAL.	1397
TRANSFER OWNERSHIP.	1400
TRUNCATE	1412
UPDATE	1415
VALUES.	1433
VALUES INTO	1434
WHENEVER	1437
WHILE	1440

Apéndice A. Visión general de la información técnica de DB2 1443

Biblioteca técnica de DB2 en copia impresa o en formato PDF	1444
Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos	1446
Acceso a diferentes versiones del Centro de información de DB2	1446
Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet	1447
Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet	1448
Guías de aprendizaje de DB2.	1450
Información de resolución de problemas de DB2	1451
Términos y condiciones	1451

Apéndice B. Avisos 1453

Índice 1457

Acerca de este manual

El manual Consulta de SQL en dos volúmenes define el lenguaje SQL utilizado por la base de datos DB2 para Linux, UNIX y Windows.

Incluye:

- Información acerca de los conceptos de las bases de datos relacionales, los elementos del lenguaje, las funciones y los formatos de las consultas (Volumen 1)
- Información acerca de la sintaxis y la semántica de las sentencias de SQL (Volumen 2)

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 pueden utilizarlo los usuarios que accedan a las bases de datos mediante el procesador de línea de mandatos (CLP).

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.

Cómo está estructurado este manual

El segundo volumen del manual *Consulta de SQL* contiene información sobre la sintaxis y la semántica de las sentencias de SQL.

- “Sentencias” contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de todas las sentencias de SQL, incluidas las sentencias de procedimiento de SQL.

Cómo leer los diagramas de sintaxis

En este tema se describe la estructura de los diagramas de la sintaxis de SQL.

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

El símbolo `??---` indica el principio de un diagrama de sintaxis.

El símbolo `---?` indica que la sintaxis continúa en la línea siguiente.

El símbolo `?---` indica que la sintaxis continúa de la línea anterior.

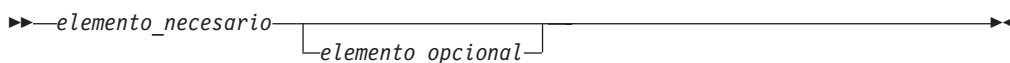
El símbolo `--??` indica el final de un diagrama de sintaxis.

Los fragmentos de sintaxis empiezan con el símbolo `+---` y finalizan con el símbolo `---!`.

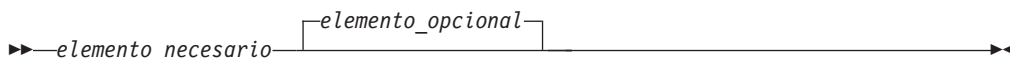
Los elementos necesarios aparecen en la línea horizontal (en la vía de acceso principal).



Los elementos opcionales aparecen bajo la vía de acceso principal.

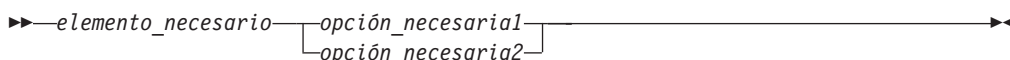


Si un elemento opcional aparece sobre la vía de acceso principal, ese elemento no tiene ningún efecto en la ejecución y sólo se utiliza para posibilitar la lectura.

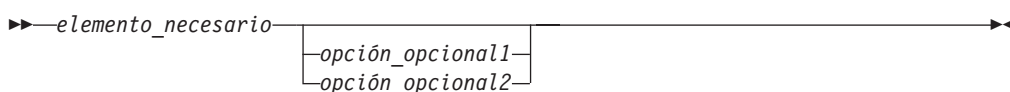


Si puede elegir entre dos o más elementos, éstos aparecen en una pila.

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

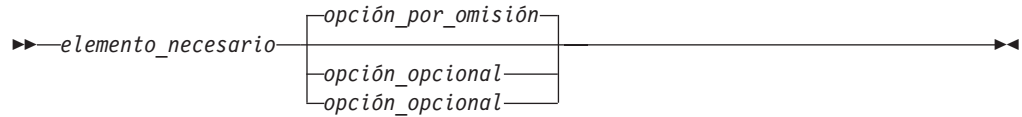


Si la elección de uno de los elementos es opcional, la pila entera aparece bajo la vía de acceso principal.

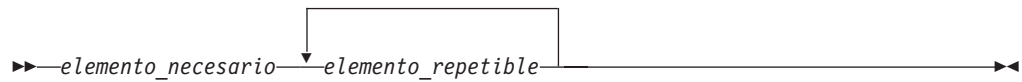


Si uno de los elementos es el valor por omisión, aparecerá sobre la vía de acceso principal y las opciones restantes se mostrarán debajo.

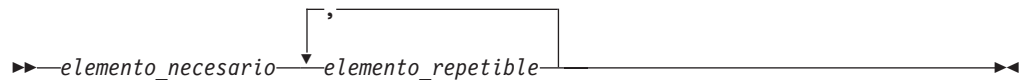
Cómo leer los diagramas de sintaxis



Una flecha que vuelve a la izquierda, sobre la línea principal, indica un elemento que se puede repetir. En este caso, los elementos repetidos se deben separar mediante uno o más espacios en blanco.



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

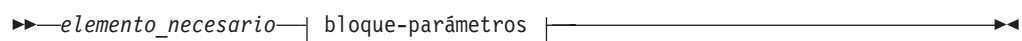


Una flecha de repetición sobre una pila indica que puede realizar más una elección en los elementos apilados o repetir una sola elección.

Las palabras clave aparecen en mayúsculas (por ejemplo FROM). Se deben escribir exactamente tal como se muestran. Las variables aparecen en minúsculas (por ejemplo nombre-columna). Representan nombres o valores proporcionados por el usuario en la sintaxis.

Si se muestran signos de puntuación, paréntesis, operadores aritméticos u otros símbolos de este tipo, debe entrarlos como parte de la sintaxis.

A veces una variable individual representa un fragmento mayor de la sintaxis. Por ejemplo, en el diagrama siguiente, la variable `bloque-parámetros` representa el fragmento de sintaxis completo que está etiquetado **bloque-parámetros**:



bloque-parámetros:



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



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

```
elemento_necesario elemento1 elemento2 elemento3 elemento4  
elemento_necesario elemento1 elemento3 elemento2 elemento4
```

Convenios utilizados 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 devuelve un error de SQL (SQLSTATE 42723).

Convenios de resaltado

Este tema trata acerca de los convenios utilizados en la Consulta de SQL.

- La **negrita** indica mandatos, palabras clave y otros elementos cuyos nombres están predefinidos por el sistema.
- La *cursiva* indica uno de los elementos siguientes:
 - Nombres o valores (variables) que el usuario debe proporcionar
 - Énfasis general
 - La introducción de un término nuevo
 - Una referencia a otra fuente de información

Documentación relacionada

Las siguientes publicaciones pueden resultarle útiles al preparar las aplicaciones:

- *Iniciación al desarrollo de aplicaciones de bases de datos*
 - Presenta el desarrollo de la aplicación DB2 e incluye los requisitos previos de la plataforma, el software de desarrollo soportado y una orientación sobre las ventajas y limitaciones de las API de programación soportadas.
- *DB2 for i5/OS SQL Reference*
 - Este manual define el soporte para SQL de DB2 Query Manager y SQL Development Kit en System i. 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 i5/OS que ejecutan DB2.
- *DB2 for z/OS SQL Reference*
 - Este manual define el SQL utilizado en DB2 para z/OS. Proporciona formatos 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 que ejecutan DB2.
- *DB2 Spatial Extender User's Guide and Reference*
 - Este manual describe cómo grabar aplicaciones para crear y utilizar un sistema de información geográfica (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.
- *IBM SQL Reference*
 - Este manual contiene todos los elementos comunes de SQL que están distribuidos por todos los productos de base de datos de IBM. Proporciona

límites y normas que pueden servir de ayuda en la preparación de programas portátiles que utilicen bases de datos de IBM. Este manual proporciona una lista de extensiones de SQL e incompatibilidades entre los siguientes estándares y productos: SQL92E, XPG4-SQL, IBM-SQL y los productos de bases de datos relacionales de IBM.

- *American National Standard X3.135-1992, Database Language SQL*
 - Contiene la definición estándar ANSI de SQL.
- *ISO/IEC 9075:1992, Database Language SQL*
 - Contiene la definición de SQL proporcionada por la norma ISO 1992.
- *ISO/IEC 9075-2:2003, Information technology -- Database Languages -- SQL -- Part 2: Foundation (SQL/Foundation)*
 - Contiene una gran parte de la definición de SQL proporcionada por la norma ISO 2003.
- *ISO/IEC 9075-4:2003, Information technology -- Database Languages -- SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
 - Contiene la definición de las sentencias de control de los procedimientos de SQL, tal como aparece en la norma ISO 2003.

Documentación relacionada

Sentencias de SQL

Este tema contiene tablas en las que se listan las sentencias de SQL clasificadas por tipo.

- Sentencias de esquema de SQL (Tabla 1)
- Sentencias de cambio de datos de SQL (Tabla 2 en la página 6)
- Sentencias de datos de SQL (Tabla 3 en la página 7)
- Sentencias de transacciones de SQL (Tabla 4 en la página 7)
- Sentencias de conexión de SQL (Tabla 5 en la página 8)
- Sentencias dinámicas de SQL (Tabla 6 en la página 8)
- Sentencias de sesiones de SQL (Tabla 7 en la página 8)
- Sentencias de lenguaje principal de SQL incorporadas (Tabla 8 en la página 10)
- Sentencias de control de SQL (Tabla 9 en la página 10)

Tabla 1. Sentencias de esquema de SQL

Sentencia de SQL	Finalidad
"ALTER AUDIT POLICY" en la página 29	Modifica la definición de una política de auditoría en el servidor actual.
"ALTER BUFFERPOOL" en la página 33	Cambia la definición de una agrupación de almacenamientos intermedios.
"ALTER DATABASE" en la página 40	Añade nuevas vías de acceso de almacenamiento a la colección de vías de acceso que se utilizan para los espacios de tablas de almacenamiento automático.
"ALTER EVENT MONITOR" en la página 45	Cambia la definición de un supervisor de sucesos de TABLE o UNFORMATTED EVENT TABLE.
"ALTER DATABASE PARTITION GROUP" en la página 36	Cambia la definición de un grupo de particiones de base de datos.
"ALTER FUNCTION" en la página 50	Modifica una función existente cambiando las propiedades de la función.
"ALTER HISTOGRAM TEMPLATE" en la página 54	Modifica la plantilla que describe el tipo de histograma que se puede utilizar para alterar temporalmente uno o varios de los histogramas por omisión de una clase de servicio o de una clase de trabajo.
"ALTER INDEX" en la página 56	Cambia la definición de un índice.
"ALTER MASK" en la página 57	Cambia la definición de una máscara de columna.
"ALTER METHOD" en la página 58	Modifica un método existente cambiando el cuerpo de método que se asocia con el método.
"ALTER MODULE" en la página 60	Cambia la definición de un módulo.
"ALTER NICKNAME" en la página 69	Cambia la definición de un apodo.
"ALTER PACKAGE" en la página 78	Modifica las opciones de vinculación de un paquete en el servidor actual sin necesidad de vincular o volver a vincular el paquete.
"ALTER PERMISSION" en la página 82	Cambia la definición de un permiso de fila.
"ALTER PROCEDURE (externo)" en la página 83	Modifica un procedimiento externo existente cambiando las propiedades del procedimiento.
"ALTER PROCEDURE (derivado)" en la página 86	Modifica un procedimiento derivado existente cambiando el tipo de datos de uno o varios parámetros del procedimiento derivado.

Sentencias de SQL

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Finalidad
"ALTER PROCEDURE (SQL)" en la página 88	Modifica un procedimiento de SQL existente cambiando las propiedades del procedimiento.
"ALTER SCHEMA" en la página 90	Modifica un esquema existente cambiando el atributo de captura de datos del esquema.
"ALTER SECURITY LABEL COMPONENT" en la página 92	Modifica un componente de etiqueta de seguridad.
"ALTER SECURITY POLICY" en la página 95	Modifica una política de seguridad.
"ALTER SEQUENCE" en la página 99	Cambia la definición de una secuencia.
"ALTER SERVER" en la página 103	Cambia la definición de una fuente de datos en un sistema federado.
"ALTER SERVICE CLASS" en la página 106	Cambia la definición de una clase de servicio.
"ALTER STOGROUP" en la página 117	Cambia la definición de un grupo de almacenamiento.
"ALTER TABLE" en la página 122	Cambia la definición de una tabla.
"ALTER TABLESPACE" en la página 194	Cambia la definición de un espacio de tablas.
"ALTER THRESHOLD" en la página 211	Cambia la definición de un umbral.
"ALTER TRIGGER" en la página 225	Cambia la definición de un activador.
"ALTER TRUSTED CONTEXT" en la página 226	Cambia la definición de un contexto fiable en el servidor actual.
"ALTER TYPE (estructurado)" en la página 235	Cambia la definición de un tipo estructurado.
"ALTER USAGE LIST" en la página 242	Cambia la definición de una lista de uso.
"ALTER USER MAPPING" en la página 244	Cambia la definición de una correlación de autorizaciones de usuario.
"ALTER VIEW" en la página 247	Cambia la definición de una vista modificando una columna de tipo de referencia para añadir un ámbito.
"ALTER WORK ACTION SET" en la página 249	Añade, modifica o descarta acciones de trabajo dentro de un conjunto de acciones de trabajo.
"ALTER WORK CLASS SET" en la página 264	Añade, modifica o descarta clases de trabajo dentro de un conjunto de clases de trabajo.
"ALTER WORKLOAD" en la página 270	Cambia una carga de trabajo.
"ALTER WRAPPER" en la página 287	Actualiza las opciones que, junto con un módulo de derivador, se utilizan para acceder a las fuentes de datos de un tipo específico.
"ALTER XSROBJECT" en la página 289	Habilita o inhabilita el soporte de la descomposición para un esquema XML específico.
"AUDIT" en la página 292	Determina la política de auditoría que se debe utilizar para una base de datos o un objeto de base de datos en particular en el servidor actual.
"COMMENT" en la página 312	Sustituye o añade un comentario a la descripción de un objeto.
"CREATE ALIAS" en la página 369	Define un alias para un módulo, apodo, secuencia, tabla, vista u otro alias.
"CREATE AUDIT POLICY" en la página 373	Define una política de auditoría en el servidor actual.
"CREATE BUFFERPOOL" en la página 377	Define una nueva agrupación de almacenamientos intermedios.
"CREATE DATABASE PARTITION GROUP" en la página 381	Define un grupo de particiones de base de datos.
"CREATE EVENT MONITOR" en la página 384	Especifica sucesos de la base de datos que se deben supervisar.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Finalidad
"CREATE EVENT MONITOR (actividades)" en la página 406	Especifica sucesos de actividad de la base de datos que se deben supervisar.
"CREATE EVENT MONITOR (historial de cambios)" en la página 418	Especifica sucesos de historial de cambios de la base de datos que se deben supervisar.
"CREATE EVENT MONITOR (bloqueo)" en la página 425	Especifica sucesos de bloqueo de la base de datos que se deben supervisar.
"Sentencia CREATE EVENT MONITOR (antememoria de paquete)" en la página 431	Especifica sucesos de sentencia de antememoria de paquete de la base de datos que han de supervisarse.
"CREATE EVENT MONITOR (estadísticas)" en la página 438	Especifica sucesos de estadísticas de la base de datos que se deben supervisar.
"CREATE EVENT MONITOR (violaciones de umbral)" en la página 451	Especifica sucesos de violación de umbral de la base de datos que se deben supervisar.
"CREATE EVENT MONITOR (unidad de trabajo)" en la página 464	Especifica sucesos de la unidad o de trabajo de la base de datos que se deben supervisar.
"CREATE FUNCTION" en la página 470	Registra una función definida por el usuario.
"CREATE FUNCTION (escalar externa)" en la página 471	Registra una función escalar externa definida por el usuario.
"CREATE FUNCTION (tabla externa)" en la página 501	Registra una función de tabla externa definida por el usuario.
"CREATE FUNCTION (tabla externa OLE DB)" en la página 523	Registra una función de tabla externa OLE DB definida por el usuario.
"CREATE FUNCTION (derivado o plantilla)" en la página 534	Registra una plantilla de función o una función derivada definida por el usuario.
"CREATE FUNCTION (tabla, fila o escalar de SQL)" en la página 549	Define una función de SQL definida por el usuario.
"CREATE FUNCTION MAPPING" en la página 567	Define una correlación de funciones.
"CREATE GLOBAL TEMPORARY TABLE" en la página 572	Define una tabla temporal creada.
"CREATE HISTOGRAM TEMPLATE" en la página 585	Define una plantilla que describe el tipo de histograma que se puede utilizar para alterar temporalmente uno o varios de los histogramas por omisión de una clase de servicio o de una clase de trabajo.
"CREATE INDEX" en la página 587	Define un índice para una tabla.
"CREATE INDEX EXTENSION" en la página 609	Define un objeto de extensión para su uso con índices sobre tablas con columnas de tipo estructurado o diferenciado.
"CREATE MASK" en la página 616	Define una máscara de columna.
"CREATE METHOD" en la página 623	Define un cuerpo de método para asociarlo con una especificación de método definida previamente.
"CREATE MODULE" en la página 629	Define un módulo.
"CREATE NICKNAME" en la página 631	Define un apodo.
"CREATE PERMISSION" en la página 645	Define un permiso de fila.
"CREATE PROCEDURE" en la página 649	Define un procedimiento.
"CREATE PROCEDURE (externo)" en la página 650	Define un procedimiento externo.

Sentencias de SQL

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Finalidad
"CREATE PROCEDURE (derivado)" en la página 667	Define un procedimiento (procedimiento derivado) que está basado en otro procedimiento (procedimiento fuente). En sistemas federados, se entiende por procedimiento federado un procedimiento derivado cuyo procedimiento fuente se encuentra en una fuente de datos soportada.
"CREATE PROCEDURE (SQL)" en la página 673	Define un procedimiento de SQL.
"CREATE ROLE" en la página 684	Define un rol en el servidor actual.
"CREATE SCHEMA" en la página 685	Define un esquema.
"CREATE SECURITY LABEL COMPONENT" en la página 688	Define un componente que se utilizará como parte de una política de seguridad.
"CREATE SECURITY LABEL" en la página 691	Define una etiqueta de seguridad.
"CREATE SECURITY POLICY" en la página 693	Define una política de seguridad.
"CREATE SEQUENCE" en la página 695	Define una secuencia.
"CREATE SERVER" en la página 713	Define una fuente de datos para una base de datos federada.
"CREATE SERVICE CLASS" en la página 701	Define una clase de servicio.
"CREATE STOGROUP" en la página 717	Define un grupo de almacenamiento nuevo dentro de la base de datos.
"CREATE SYNONYM" en la página 721	Define un sinónimo para un módulo, apodo, secuencia, tabla, vista u otro sinónimo.
"CREATE TABLE" en la página 722	Define una tabla.
"CREATE TABLESPACE" en la página 810	Define un espacio de tablas.
"CREATE THRESHOLD" en la página 827	Define un umbral.
"CREATE TRANSFORM" en la página 845	Define funciones de transformación.
"CREATE TRIGGER" en la página 849	Define un activador.
"CREATE TRUSTED CONTEXT" en la página 865	Define un contexto fiable en el servidor actual.
"CREATE TYPE" en la página 872	Define un tipo de datos definido por el usuario en el servidor actual.
"CREATE TYPE (matriz)" en la página 873	Define un tipo de matriz.
"CREATE TYPE (cursor)" en la página 880	Define un tipo de cursor.
"CREATE TYPE (diferenciado)" en la página 883	Define un tipo de datos diferenciado.
"CREATE TYPE (fila)" en la página 891	Define un tipo de fila.
"CREATE TYPE (estructurado)" en la página 896	Define un tipo de datos estructurado.
"CREATE TYPE MAPPING" en la página 921	Define una correlación entre tipos de datos.
"CREATE USAGE LIST" en la página 928	Define una lista de uso para supervisar todas las secciones exclusivas (sentencias DML) que han hecho referencia a una tabla o índice determinados durante la ejecución.
"CREATE USER MAPPING" en la página 932	Define una correlación entre autorizaciones de usuario.
"CREATE VARIABLE" en la página 935	Define una variable global.
"CREATE VIEW" en la página 945	Define una vista de una o más tablas, vistas o apodos.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Finalidad
"CREATE WORK ACTION SET" en la página 961	Define un conjunto de acciones de trabajo y las acciones de trabajo contenidas en el conjunto de acciones de trabajo.
"CREATE WORK CLASS SET" en la página 971	Define un conjunto de clases de trabajo.
"CREATE WORKLOAD" en la página 977	Define una carga de trabajo.
"CREATE WRAPPER" en la página 996	Registra un derivador.
"DROP" en la página 1042	Suprime objetos de la base de datos.
"GRANT (autorizaciones de bases de datos)" en la página 1117	Otorga autorizaciones sobre toda una base de datos.
"GRANT (exención)" en la página 1123	Otorga una exención sobre una norma de acceso para una política de seguridad LBAC (control de acceso basado en etiquetas) especificada.
"GRANT (privilegios de variable global)" en la página 1126	Otorga uno o más privilegios en una variable global creada.
"GRANT (privilegios de índice)" en la página 1129	Otorga el privilegio CONTROL en índices en la base de datos.
"GRANT (privilegios de módulo)" en la página 1131	Otorga privilegios para un módulo.
"GRANT (privilegios de paquete)" en la página 1133	Otorga privilegios para paquetes de la base de datos.
"GRANT (rol)" en la página 1136	Otorga roles a usuarios, grupos o a otros roles.
"GRANT (privilegios de rutina)" en la página 1139	Otorga privilegios para una rutina (función, método o procedimiento).
"GRANT (privilegios de esquema)" en la página 1144	Otorga privilegios para un esquema.
"GRANT (etiqueta de seguridad)" en la página 1147	Otorga una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC) para un acceso de lectura, acceso de grabación o para ambos tipos de acceso.
"GRANT (privilegios de secuencia)" en la página 1150	Otorga privilegios en una secuencia.
"GRANT (privilegios de servidor)" en la página 1153	Otorga privilegios para consultar una fuente de datos específica.
"GRANT (privilegio SETSESSIONUSER)" en la página 1155	Otorga el privilegio de utilización de la sentencia SET SESSION AUTHORIZATION.
"GRANT (privilegios de espacio de tablas)" en la página 1157	Otorga privilegios para un espacio de tablas.
"GRANT (privilegios de tabla, vista o apodo)" en la página 1159	Otorga privilegios para tablas, vistas y apodos.
"GRANT (privilegios de carga de trabajo)" en la página 1166	Otorga el privilegio USAGE en una carga de trabajo.
"GRANT (privilegios de objeto XSR)" en la página 1168	Otorga el privilegio USAGE sobre un objeto XSR.
"REFRESH TABLE" en la página 1221	Renueva los datos de una tabla de consulta materializada.
"RENAME" en la página 1228	Cambia el nombre de una tabla existente.
"RENAME STOGROUP" en la página 1230	Cambia el nombre de un grupo de almacenamiento existente.
"RENAME TABLESPACE" en la página 1231	Cambia el nombre de un espacio de tablas existente.

Sentencias de SQL

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Finalidad
“REVOKE (autorizaciones de bases de datos)” en la página 1240	Revoca autorizaciones de toda una base de datos.
“REVOKE (exención)” en la página 1244	Revoca la exención sobre una norma de acceso para una política de seguridad LBAC (control de acceso basado en etiquetas).
“REVOKE (privilegios de variable global)” en la página 1246	Revoca uno o más privilegios en una variable global creada.
“REVOKE (privilegios de índice)” en la página 1249	Revoca el privilegio CONTROL en índices determinados.
“REVOKE (privilegios de módulo)” en la página 1251	Revoca los privilegios para un módulo.
“REVOKE (privilegios de paquete)” en la página 1253	Revoca los privilegios de paquetes determinados en la base de datos.
“REVOKE (rol)” en la página 1256	Revoca roles de usuarios, grupos u otros roles.
“REVOKE (privilegios de rutina)” en la página 1259	Revoca privilegios para una rutina (función, método o procedimiento).
“REVOKE (privilegios de esquema)” en la página 1263	Revoca privilegios para un esquema.
“REVOKE (etiqueta de seguridad)” en la página 1265	Revoca una etiqueta de seguridad LBAC (control de acceso basado en etiquetas) para un acceso de lectura, acceso de grabación o ambos tipos de acceso.
“REVOKE (privilegios de secuencia)” en la página 1267	Revoca los privilegios de una secuencia.
“REVOKE (privilegios de servidor)” en la página 1270	Revoca privilegios para consultar una fuente de datos específica.
“REVOKE (privilegio SETSESSIONUSER)” en la página 1272	Revoca el privilegio de utilizar la sentencia SET SESSION AUTHORIZATION.
“REVOKE (privilegios de espacio de tablas)” en la página 1274	Revoca el privilegio de utilización (USE) para un espacio de tablas determinado.
“REVOKE (privilegios de tabla, vista o apodo)” en la página 1276	Revoca privilegios para determinadas tablas, vistas o apodos.
“REVOKE (privilegios de carga de trabajo)” en la página 1282	Revoca el privilegio USAGE en una carga de trabajo.
“REVOKE (privilegios de objeto XSR)” en la página 1284	Revoca el privilegio USAGE en un objeto XSR.
“SET INTEGRITY” en la página 1350	Establece el estado Pendiente de establecimiento de integridad y comprueba los datos para detectar violaciones de restricción.
“TRANSFER OWNERSHIP” en la página 1400	Transfiere la propiedad de un objeto de base de datos.

Tabla 2. Sentencias de cambio de datos de SQL

Sentencia de SQL	Finalidad
“DELETE” en la página 1018	Suprime una o más filas de una tabla.
“INSERT” en la página 1173	Inserta una o más filas en una tabla.
“MERGE” en la página 1193	Actualiza un destino (tabla o vista) utilizando los datos de una fuente (resultado de una referencia de tabla).
“TRUNCATE” en la página 1412	Suprime todas las filas de una tabla.

Tabla 2. Sentencias de cambio de datos de SQL (continuación)

Sentencia de SQL	Finalidad
"UPDATE" en la página 1415	Actualiza los valores de una o varias columnas en una o más filas de una tabla.

Tabla 3. Sentencias de datos de SQL

Sentencia de SQL	Finalidad
"ALLOCATE CURSOR" en la página 27	Asigna un cursor para el conjunto de resultados identificados por la variable del localizador de conjunto de resultados.
"ASSOCIATE LOCATORS" en la página 290	Obtiene el valor del localizador para cada conjunto de resultados devuelto por un procedimiento.
"CLOSE" en la página 310	Cierra un cursor.
"DECLARE CURSOR" en la página 998	Define un cursor SQL.
"FETCH" en la página 1095	Asigna valores de una fila a variables del lenguaje principal.
"FLUSH BUFFERPOOLS" en la página 1100	Graba las páginas sucias de las agrupaciones de almacenamientos intermedios en el disco.
"FLUSH EVENT MONITOR" en la página 1101	Graba el almacenamiento intermedio interno activo de un supervisor de sucesos.
"FLUSH FEDERATED CACHE" en la página 1102	La sentencia FLUSH FEDERATED CACHE vacía la antememoria federada, lo que permite la obtención de nuevos metadatos la próxima vez que se emita una sentencia de SQL para la vista o tabla remota utilizando un nombre de tres partes federado.
"FLUSH OPTIMIZATION PROFILE CACHE" en la página 1104	Elimina los perfiles de optimización colocados en antememoria.
"FLUSH PACKAGE CACHE" en la página 1106	Elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes.
"FREE LOCATOR" en la página 1110	Elimina la asociación entre una variable localizadora y su valor.
"LOCK TABLE" en la página 1189	Impide que los procesos simultáneos cambien una tabla o impide que los procesos simultáneos utilicen una tabla.
"OPEN" en la página 1206	Prepara un cursor que se utilizará para recuperar valores cuando se emita la sentencia FETCH.
"SELECT INTO" en la página 1292	Especifica una tabla de resultados que no tiene más de una fila y asigna los valores a variables del lenguaje principal.
"SET variable" en la página 1385	Asigna valores a variables.
"VALUES INTO" en la página 1434	Especifica una tabla de resultados que no tiene más de una fila y asigna los valores a variables del lenguaje principal.

Tabla 4. Sentencias de transacciones de SQL

Sentencia de SQL	Finalidad
"COMMIT" en la página 323	Finaliza una unidad de trabajo y confirma los cambios que esa unidad de trabajo ha realizado en la base de datos.
"RELEASE SAVEPOINT" en la página 1227	Libera un punto de salvaguarda dentro de una transacción.
"ROLLBACK" en la página 1285	Termina una unidad de trabajo y restituye los cambios realizados por dicha unidad de trabajo.
"SAVEPOINT" en la página 1288	Define un punto de salvaguarda dentro de una transacción.

Sentencias de SQL

Tabla 5. Sentencias de conexión de SQL

Sentencia de SQL	Finalidad
"CONNECT (tipo 1)" en la página 353	Conecta a un servidor de aplicaciones según las normas para una unidad de trabajo remota.
"CONNECT (tipo 2)" en la página 361	Conecta a un servidor de aplicaciones según las normas para la unidad de trabajo distribuida dirigida por aplicación.
"DISCONNECT" en la página 1039	Finaliza una o más conexiones cuando no hay ninguna unidad de trabajo activa.
"RELEASE (conexión)" en la página 1225	Coloca una o más conexiones en el estado pendiente de liberación.
"SET CONNECTION" en la página 1297	Cambia el estado de una conexión de inactivo a actual, haciendo que la ubicación especificada sea el servidor actual.

Tabla 6. Sentencias dinámicas de SQL

Sentencia de SQL	Finalidad
"DESCRIBE" en la página 1029	Obtiene información acerca de un objeto.
"DESCRIBE INPUT" en la página 1030	Obtiene información sobre los marcadores de parámetro de entrada de una sentencia preparada.
"DESCRIBE OUTPUT" en la página 1034	Obtiene información sobre una sentencia preparada o información sobre las columnas de lista de selección de una sentencia SELECT preparada.
"EXECUTE" en la página 1079	Ejecuta una sentencia de SQL preparada.
"EXECUTE IMMEDIATE" en la página 1087	Prepara y ejecuta una sentencia de SQL.
"PREPARE" en la página 1214	Prepara una sentencia de SQL (con parámetros opcionales) para su ejecución.

Tabla 7. Sentencias de sesiones de SQL

Sentencia de SQL	Finalidad
"DECLARE GLOBAL TEMPORARY TABLE" en la página 1004	Define una tabla temporal declarada.
"EXPLAIN" en la página 1090	Captura información acerca del plan de acceso elegido.
"SET COMPILATION ENVIRONMENT" en la página 1296	Cambia el entorno de compilación actual en la conexión para que coincida con los valores incluidos en el entorno de compilación proporcionado por un supervisor de sucesos de punto muerto.
"SET CURRENT DECFLOAT ROUNDING MODE" en la página 1299	Verifica que la modalidad de redondeo especificada sea el valor que está establecido actualmente para el registro especial CURRENT DECFLOAT ROUNDING MODE.
"SET CURRENT DEFAULT TRANSFORM GROUP" en la página 1301	Cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP.
"SET CURRENT DEGREE" en la página 1303	Cambia el valor del registro especial CURRENT DEGREE.
"SET CURRENT EXPLAIN MODE" en la página 1305	Cambia el valor del registro especial CURRENT EXPLAIN MODE.
"SET CURRENT EXPLAIN SNAPSHOT" en la página 1308	Cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT.
"SET CURRENT FEDERATED ASYNCHRONY" en la página 1311	Cambia el valor del registro especial CURRENT FEDERATED ASYNCHRONY.
"SET CURRENT IMPLICIT XMLPARSE OPTION" en la página 1313	Cambia el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION.

Tabla 7. Sentencias de sesiones de SQL (continuación)

Sentencia de SQL	Finalidad
“SET CURRENT ISOLATION” en la página 1314	Cambia el valor del registro especial CURRENT ISOLATION.
“SET CURRENT LOCALE LC_MESSAGES” en la página 1315	Cambia el valor del registro especial CURRENT LOCALE LC_MESSAGES.
“SET CURRENT LOCALE LC_TIME” en la página 1317	Cambia el valor del registro especial CURRENT LOCALE LC_TIME.
“SET CURRENT LOCK TIMEOUT” en la página 1319	Cambia el valor del registro especial CURRENT LOCK TIMEOUT.
“SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION” en la página 1321	Cambia el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.
“SET CURRENT MDC ROLLOUT MODE” en la página 1323	Asigna un valor al registro especial CURRENT MDC ROLLOUT MODE.
“SET CURRENT OPTIMIZATION PROFILE” en la página 1325	Asigna un valor al registro especial CURRENT OPTIMIZATION PROFILE.
“SET CURRENT PACKAGE PATH” en la página 1328	Asigna un valor al registro especial CURRENT PACKAGE PATH.
“SET CURRENT PACKAGESET” en la página 1332	Establece el nombre de esquema para la selección de paquetes.
“SET CURRENT QUERY OPTIMIZATION” en la página 1334	Cambia el valor del registro especial CURRENT QUERY OPTIMIZATION.
“SET CURRENT REFRESH AGE” en la página 1337	Cambia el valor del registro especial CURRENT REFRESH AGE.
“SET CURRENT SQL_CCFLAGS” en la página 1339	Cambia el valor del registro especial CURRENT SQL_CCFLAGS.
“SET CURRENT TEMPORAL BUSINESS_TIME” en la página 1341	Cambia el valor del registro especial CURRENT TEMPORAL BUSINESS_TIME.
“SET CURRENT TEMPORAL SYSTEM_TIME” en la página 1343	Cambia el valor del registro especial CURRENT TEMPORAL SYSTEM_TIME.
“SET ENCRYPTION PASSWORD” en la página 1345	Establece la contraseña para el cifrado.
“SET EVENT MONITOR STATE” en la página 1347	Activa o desactiva un supervisor de sucesos.
“SET PASSTHRU” en la página 1370	Abre una sesión para someter SQL nativo de fuente de datos directamente a la fuente de datos.
“SET PATH” en la página 1372	Cambia el valor del registro especial CURRENT PATH.
“SET ROLE” en la página 1374	Verifica que el ID de autorización de la sesión sea un miembro de un rol específico.
“SET SCHEMA” en la página 1375	Cambia el valor del registro especial CURRENT SCHEMA.
“SET SERVER OPTION” en la página 1377	Establece valores de opciones del servidor.
“SET SESSION AUTHORIZATION” en la página 1379	Cambia el valor del registro especial SESSION USER.
“SET USAGE LIST STATE” en la página 1382	Gestiona el estado de una lista de uso y de los datos y la memoria asociados.

Sentencias de SQL

Tabla 8. Sentencias de lenguaje principal de SQL incorporadas

Sentencia de SQL	Finalidad
"BEGIN DECLARE SECTION" en la página 296	Marca el principio de una sección de declaración de variables del lenguaje principal.
"END DECLARE SECTION" en la página 1078	Marca el final de una sección de declaración de variables del lenguaje principal.
"GET DIAGNOSTICS" en la página 1111	Se utiliza para obtener información acerca de la sentencia de SQL ejecutada previamente.
"INCLUDE" en la página 1171	Inserta código o declaraciones en un programa fuente.
"RESIGNAL" en la página 1235	Se utiliza para retransmitir una condición de error o aviso.
"SIGNAL" en la página 1397	Se utiliza para transmitir una condición de error o aviso.
"WHENEVER" en la página 1437	Define las acciones que se deben tomar sobre la base de los códigos de retorno SQL.

Tabla 9. Sentencias de control de SQL

Sentencia de SQL	Finalidad
"CALL" en la página 298	Llama a un procedimiento.
"CASE" en la página 307	Selecciona una vía de acceso de ejecución basándose en múltiples condiciones.
"SQL compuesto" en la página 325	Incluye las sentencias de SQL entre las palabras clave BEGIN y END.
"SQL compuesto (en línea)" en la página 326	Combina una o más sentencias de SQL diferentes en un bloque dinámico.
"SQL compuesto (incorporado)" en la página 331	Combina una o varias sentencias de SQL para formar un bloque ejecutable.
"SQL compuesto (compilado)" en la página 335	Agrupar otras sentencias en un procedimiento de SQL.
"FOR" en la página 1107	Ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.
"GOTO" en la página 1115	Se utiliza para ramificar a una etiqueta definida por el usuario dentro de un procedimiento de SQL.
"IF" en la página 1169	Selecciona una vía de acceso de ejecución basándose en la evaluación de una condición.
"ITERATE" en la página 1185	Provoca que el flujo de control vuelva al principio de un bucle con etiqueta.
"LEAVE" en la página 1187	Transfiere el control del programa fuera de un bucle o de una sentencia compuesta.
"LOOP" en la página 1191	Repite la ejecución de una sentencia o grupo de sentencias.
"PIPE" en la página 1212	Devuelve una fila de una función de tabla compilada.
"REPEAT" en la página 1233	Ejecuta una sentencia o grupo de sentencias hasta que una condición de búsqueda es verdadera.
"RESIGNAL" en la página 1235	Se utiliza para retransmitir una condición de error o aviso.
"RETURN" en la página 1238	Se utiliza para volver de una rutina.
"SIGNAL" en la página 1397	Se utiliza para transmitir una condición de error o aviso.
"WHILE" en la página 1440	Repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

Cómo se invocan las sentencias de SQL

Las sentencias de SQL se clasifican como ejecutables y no ejecutables.

Una *sentencia ejecutable* puede invocarse de cuatro formas. Puede ser:

- Emitida interactivamente
- Preparada y ejecutada dinámicamente
- Incorporada en un programa de aplicación
- Incorporada en un procedimiento de SQL, activador, SQL compuesto (compilado) o SQL compuesto (en línea) con algunas restricciones:
 - Consulte la sección “sentencia-procedimiento-SQL” en “SQL compuesto (compilado)” en la página 335 para conocer el conjunto de sentencias ejecutables que reciben soporte en los procedimientos de SQL y las sentencias de SQL compuesto (compilado).
 - Consulte la sección “sentencia-SQL” de la sentencia “SQL compuesto (en línea)” en la página 326 para conocer el conjunto de sentencias ejecutables que reciben soporte en las sentencias de SQL compuesto (en línea).
 - Consulte la sección “sentencia-procedimiento-SQL” en “CREATE TRIGGER” en la página 849 para conocer el conjunto de sentencias ejecutables que reciben soporte en un activador.

Se pueden utilizar algunos o todos estos métodos, dependiendo de la sentencia. Las sentencias que se incorporan en REXX se preparan y ejecutan dinámicamente.

Una *sentencia no ejecutable* sólo puede incorporarse en un programa de aplicación.

Otra construcción de sentencia de SQL es la sentencia de selección. Una *sentencia de selección* puede invocarse de tres formas. Puede ser:

- Emitida interactivamente
- Preparada dinámicamente, con referencia en DECLARE CURSOR y ejecutada implícitamente por OPEN, FETCH y CLOSE (invocación dinámica)
- Incluida en DECLARE CURSOR y ejecutada implícitamente por OPEN, FETCH y CLOSE (invocación estática)

Incorporación de una sentencia a un programa de aplicación

Las sentencias de SQL pueden incorporarse en un programa fuente que se someterá a un 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 de las palabras clave EXEC SQL.

Una sentencia ejecutable incorporada a un programa de aplicación se ejecuta cada vez que habría de ejecutarse una sentencia del lenguaje del sistema principal si ésta se hubiera especificado 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 forma puede utilizarse de dos formas. Puede utilizarse:

Incorporación de una sentencia a un programa de aplicación

- Como entrada (el valor actual de la variable del lenguaje principal se utiliza en la ejecución de la sentencia)
- Como salida (como resultado de la ejecución de la sentencia, a la variable se asigna un nuevo valor)

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.

A todas las sentencias ejecutables debe seguir una prueba del código de retorno de SQL. Por otra parte, la sentencia `WHENEVER` (que, en sí, es no ejecutable) puede utilizarse para cambiar el flujo de control inmediatamente después de la ejecución de una sentencia incorporada.

Todos los objetos a los que se hace referencia en las sentencias de lenguaje de manipulación de datos (DML) deben existir al enlazar las sentencias con una base de datos.

Una sentencia no ejecutable incorporada sólo la procesa el precompilador. El precompilador informa de cualquier error encontrado en la sentencia. La sentencia *nunca* se procesa durante la ejecución del programa; por lo tanto, a tales sentencias no debe seguir una prueba del código de retorno de SQL.

Se pueden incorporar sentencias al cuerpo de un procedimiento de SQL correspondiente a la sentencia `CREATE PROCEDURE`. Se dice que las sentencias de este tipo son sentencias incorporadas al procedimiento de SQL. Siempre que una descripción de sentencia de SQL haga referencia a una *variable del lenguaje principal*, podrá utilizarse una *variable de SQL* si la sentencia se ha incorporado a un procedimiento de 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 (que no sea una sentencia de selección) construida puede prepararse para su ejecución por medio de la sentencia `PREPARE` (incorporada) y puede ejecutarse por medio de la sentencia `EXECUTE` (incorporada). Por otra parte, puede utilizarse una sentencia `EXECUTE IMMEDIATE` (incorporada) para preparar y para ejecutar la sentencia en un solo 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 parámetro. (Para obtener información acerca de las normas relacionadas con los marcadores de parámetro, consulte "PREPARE".) Cuando se ejecuta la sentencia preparada, los marcadores de parámetro se sustituyen de manera efectiva por los valores actuales de las variables del lenguaje principal especificadas en la sentencia `EXECUTE`. Una vez preparada, una sentencia puede ejecutarse varias veces con distintos valores para las variables del lenguaje principal. Los marcadores de parámetro no están permitidos en la sentencia `EXECUTE IMMEDIATE`.

La ejecución satisfactoria o no satisfactoria de la sentencia se indica por medio de un código de retorno de SQL en la `SQLCA` tras completarse la sentencia `EXECUTE` (o `EXECUTE IMMEDIATE`). El código de retorno de SQL debe comprobarse, tal

como se describe anteriormente. Para obtener más información, consulte el apartado “Detección y proceso de condiciones de error y de aviso en aplicaciones de lenguaje principal” en la página 14.

Invocación estática de una sentencia de selección

Una sentencia de selección puede incorporarse como parte de la sentencia DECLARE CURSOR (no ejecutable).

Una sentencia de este tipo se ejecuta cada vez que se abre el cursor por medio de la sentencia OPEN (incorporada). Después de haberse abierto el cursor, la tabla de resultados puede recuperarse, una fila cada vez, mediante la realización de ejecuciones sucesivas de la sentencia FETCH.

Cuando se utiliza de esta forma, la sentencia de selección puede contener referencias a variables del lenguaje principal. Estas referencias se sustituyen de forma eficaz por los valores que tienen las variables al ejecutarse la sentencia OPEN.

Invocación dinámica de una sentencia de selección

Un programa de aplicación puede crear dinámicamente una sentencia de selección en forma de una serie de caracteres que se coloca 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 que se construye de esta forma puede prepararse para su ejecución por medio de la sentencia PREPARE (incorporada) y puede hacerse referencia a ésta por medio de una sentencia DECLARE CURSOR (no ejecutable). Entonces, la sentencia se ejecuta cada vez que se abre el cursor por medio de la sentencia OPEN (incorporada). Después de haberse abierto el cursor, la tabla de resultados puede recuperarse, una fila cada vez, mediante la realización de ejecuciones sucesivas de la sentencia FETCH.

Cuando se utiliza de esta forma, la sentencia de selección no debe contener referencias a variables del lenguaje principal. Puede contener marcadores de parámetro en su lugar. Los marcadores de parámetro se sustituyen de manera efectiva por los valores de las variables del lenguaje principal especificadas en la sentencia OPEN.

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 de este tipo debe ser una sentencia ejecutable que no contenga marcadores de parámetro o referencias a las variables del lenguaje principal, pues ello sólo tiene sentido en el contexto de un programa de aplicación.

Utilización de SQL con otros sistemas principales

La sintaxis la de sentencia de SQL muestra pocas variaciones entre los diferentes tipos de sistemas principales ((DB2 para z/OS, DB2 para i, DB2 para Linux, UNIX y Windows).

Utilización de SQL con otros sistemas principales

Con independencia de si las sentencias de SQL de una aplicación son dinámicas o estáticas, es importante - si la aplicación se destina al acceso a distintos sistemas principales de base de datos - asegurarse de que las sentencias de SQL y las opciones de precompilación/vinculación reciben soporte en los sistemas de bases de datos a los que accederá la aplicación.

Se puede encontrar información adicional sobre las sentencias de SQL utilizadas en otros sistemas principales en el manual de consulta de SQL de DB2 para z/OS y DB2 para i.

Detección y proceso de condiciones de error y de aviso en aplicaciones de lenguaje principal

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`. La `SQLCA` incluye una variable entera llamada `SQLCODE` y una variable de serie de caracteres llamada `SQLSTATE`. En REXX, se proporciona automáticamente una `SQLCA`. En otros lenguajes, la `SQLCA` se puede obtener utilizando la sentencia `INCLUDE SQLCA`.
- Si `LANGLEVEL SQL92E` se ha especificado como opción de precompilación, puede declararse una variable denominada `SQLCODE` o `SQLSTATE` en la sección de declaración de SQL del programa. Si no se ha declarado ninguna de estas variables en la sección de declaración de SQL, se da por supuesta la declaración de una variable denominada `SQLCODE` en algún otro lugar del programa. Con `LANGLEVEL SQL92E`, el programa no debe tener una sentencia `INCLUDE SQLCA`.

El gestor de bases de datos establece una variable `SQLCODE` tras la ejecución de 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 de resultados.
- 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, pero se han establecido uno o más indicadores de aviso.
- Si `SQLCODE < 0`, la ejecución no ha sido satisfactoria.

El significado de los valores `SQLCODE` que no sean 0 ni 100 es específico del producto.

El gestor de bases de datos establece una variable `SQLSTATE` tras la ejecución de cada sentencia de SQL. Los programas de aplicación pueden comprobar la ejecución de las sentencias de SQL probando `SQLSTATE` en lugar de `SQLCODE`. `SQLSTATE` proporciona códigos comunes para las condiciones de errores comunes. Los programas de aplicación pueden realizar pruebas para comprobar si existen errores o clases de errores específicos. El esquema de codificación es igual para todos los gestores de bases de datos de IBM® y se basa en el estándar ISO/ANSI SQL92.

Comentarios de SQL

Las sentencias de SQL estático pueden incluir comentarios de SQL o del lenguaje principal. Las sentencias de SQL dinámico pueden incluir comentarios de SQL.

Existen dos tipos de comentarios de SQL:

comentarios simples

Los comentarios simples comienzan con dos guiones consecutivos (--) y finalizan con el final de línea.

comentarios compuestos

Los comentarios compuestos empiezan por /* y finalizan por */.

Las normas siguientes se aplican a la utilización de comentarios simples:

- Los dos guiones deben estar en la misma línea y no deben estar separados por un espacio.
- Los comentarios simples pueden empezar dondequiera que un espacio sea válido (excepto dentro de un símbolo delimitador o entre 'EXEC' y 'SQL').
- Los comentarios simples no pueden continuar en la línea siguiente.
- En COBOL, los guiones deben ir precedidos por un espacio.

Las normas siguientes se aplican a la utilización de comentarios compuestos:

- El /* debe estar en la misma línea y no debe estar separado por un espacio.
- El */ debe estar en la misma línea y no debe estar separado por un espacio.
- Los comentarios compuestos pueden empezar dondequiera que un espacio sea válido (excepto dentro de un símbolo delimitador o entre 'EXEC' y 'SQL').
- Los comentarios compuestos pueden continuar en las líneas posteriores.

Ejemplos

- *Ejemplo 1:* Este ejemplo muestra cómo incluir comentarios simples en una sentencia:

```
CREATE VIEW PRJ_MAXPER      -- PROJECTS WITH MOST SUPPORT PERSONNEL
AS SELECT PROJNO, PROJNAME -- NUMBER AND NAME OF PROJECT
FROM PROJECT
WHERE DEPTNO = 'E21'      -- SYSTEMS SUPPORT DEPT CODE
AND PRSTAFF > 1
```

- *Ejemplo 2:* Este ejemplo muestra cómo incluir comentarios compuestos en una sentencia:

```
CREATE VIEW PRJ_MAXPER      /* PROJECTS WITH MOST SUPPORT
                             PERSONNEL */
AS SELECT PROJNO, PROJNAME /* NUMBER AND NAME OF PROJECT */
FROM PROJECT
WHERE DEPTNO = 'E21'      /* SYSTEMS SUPPORT DEPT CODE */
AND PRSTAFF > 1
```

Compilación condicional en SQL

La compilación condicional permite a SQL incluir directivas de compilador que se utilizan para determinar el SQL real que se compila.

Existen dos tipos de directivas de compilador que pueden utilizarse para la compilación condicional:

Directiva de selección

Sentencia de control de compilador que se utiliza para determinar la

Compilación condicional en SQL

selección de un fragmento de código. La directiva `_IF` puede hacer referencia a directivas de consulta o variables globales que se han definido como una constante.

Directiva de consulta

Referencia a una constante con nombre de compilador que el sistema asigna o que se especifica como una constante con nombre de compilación condicional en `CURRENT SQL_CCFLAGS`. Una directiva de consulta puede utilizarse directamente o en una directiva de selección.

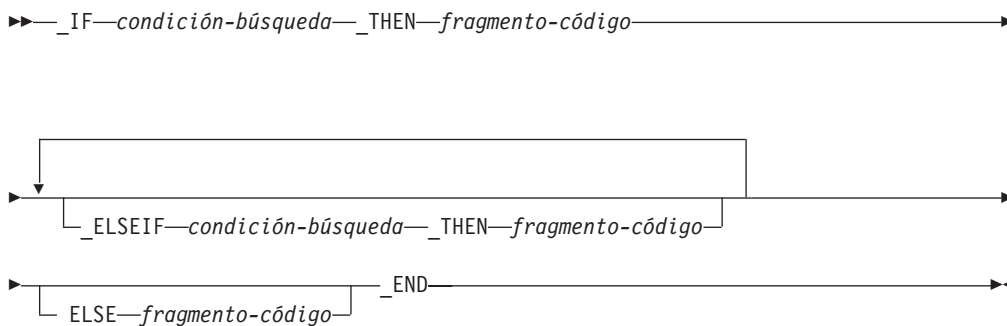
Estas directivas pueden utilizarse en los contextos que se indican a continuación:

- Definiciones de procedimientos de SQL
- Definiciones de funciones de SQL compilado
- Definiciones de activadores compilados
- Definiciones de paquetes PL/SQL de Oracle

Una directiva sólo puede aparecer después del tipo de objeto (`FUNCTION`, `PACKAGE`, `PACKAGE BODY`, `PROCEDURE` o `TRIGGER`) que se ha identificado en la sentencia de lenguaje de definición.

Directiva de selección

La directiva de selección es muy similar a la sentencia `IF`, a excepción de que existen prefijos en las palabras clave para indicar la utilización de compilación condicional y de que la palabra clave de terminación es `_END`.



condición-búsqueda

Especifica la condición que se evalúa para determinar qué *fragmento-código*, si existe, ha de incluirse. Si la condición es desconocida o falsa, la evaluación continúa con la siguiente condición de búsqueda, hasta que una condición es verdadera, se ha llegado a la cláusula `_ELSE` o se ha alcanzado el final de la directiva de selección. La condición de búsqueda sólo puede incluir los elementos siguientes (SQLSTATE 428HV):

- Constantes de tipo `BOOLEAN`, `INTEGER` o `VARCHAR`
- Constantes nulas (`NULL`)
- Directivas de consulta
- Constantes globales, donde el valor de constante definido es un literal simple de tipo `BOOLEAN`, `INTEGER` o `VARCHAR`
- Predicados básicos
- Predicados nulos (`NULL`)

- Predicados que son una constante booleana o una directiva de consulta booleana
- Operadores lógicos (AND, OR y NOT)

fragmento-código

Parte de código SQL que puede incluirse en el contexto de la sentencia de SQL donde aparece la directiva de selección. No debe ser una directiva de selección en *fragmento-código* (SQLSTATE 428HV).

Directiva de consulta

Una directiva de consulta se utiliza para realizar consultas relacionadas con el entorno de compilación. En una sentencia de SQL, una directiva de consulta se especifica como identificador normal precedido de dos caracteres de subrayado. El identificador real puede representar uno de los valores siguientes:

- Un valor de entorno de compilación que el sistema define
- Un valor de compilación que el usuario define en la base de datos o en la sesión individual

La única variable de entorno de compilación que el sistema define es `__SQL_LINE`, que proporciona el número de línea de SQL que está compilándose actualmente.

Una variable de compilación definida por el usuario puede definirse en la base de datos mediante la utilización del parámetro de configuración de base de datos `sql_ccflags` o bien en la sesión mediante la asignación de ésta al registro especial `CURRENT SQL_CCFLAGS`.

Si se hace referencia a una directiva de consulta pero ésta no se ha definido, el proceso continúa y se da por supuesto que el valor de la directiva de consulta es el valor nulo.

Notas

- **Referencias a variables globales definidas como constantes:** una referencia a una variable global (que también puede ser una referencia a una variable de módulo publicada en un módulo) de una directiva de selección sólo se utiliza para proporcionar un valor basado en una constante en tiempo de compilación. La variable global a la que se hace referencia debe satisfacer los requisitos siguientes:

- Debe existir en el servidor actual (SQLSTATE 42704)
- Debe tener un tipo de datos `BOOLEAN`, `INTEGER` o `VARCHAR` (SQLSTATE 428HV)
- Debe haberse definido mediante la utilización de la cláusula `CONSTANT` con un único valor de constante (SQLSTATE 428HV)

Una variable global de este tipo se denomina constante global. Los cambios que posteriormente se realicen en la constante global no tendrán ningún efecto en las sentencias que ya se han compilado.

- **Alternativas de sintaxis:** si el entorno de servidor de datos se ha habilitado para la ejecución de sentencias de PL/SQL:
 - Puede especificarse `ELSIF` en lugar de `ELSEIF`
 - Puede usarse un carácter de dólar (\$) en lugar de un carácter de subrayado (__) como prefijo para las palabras clave para la compilación condicional
 - Pueden usarse dos caracteres de dólar (\$\$) en lugar de dos caracteres de subrayado (__) como prefijo para una directiva de consulta

Compilación condicional en SQL

La única finalidad del prefijo del carácter del dólar es dar soporte a las sentencias de SQL existentes que utilizan directivas de consulta y no se recomienda utilizarlo para escribir nuevas sentencias de SQL.

Ejemplo

Especificar un valor para toda la base de datos para un valor de compilación denominado DBV97 que tenga el valor TRUE.

```
UPDATE DATABASE CONFIGURATION USING SQL_CCFLAGS DB2V97:TRUE
```

El valor está disponible como valor por omisión para cualquier conexión que se establezca posteriormente con la base de datos.

Si una sesión en particular ha necesitado un valor de compilación de número máximo de años para su utilización en la definición de algunas rutinas en la sesión actual, el valor por omisión SQL_CCFLAGS puede ampliarse mediante la utilización de la sentencia SET CURRENT SQL_CCFLAGS.

```
BEGIN
DECLARE CCFLAGS_LIST VARCHAR(1024);
SET CCFLAGS_LIST = CURRENT SQL_CCFLAGS CONCAT ',max_years:50';
SET CURRENT SQL_CCFLAGS = CCFLAGS_LIST;
END
```

La utilización de CURRENT SQL_CCFLAGS en la parte derecha de la asignación a la variable CCFLAGS_LIST conserva los valores SQL_CCFLAGS existentes, mientras que la constante de la serie proporciona los valores de compilación adicionales.

A continuación se muestra un ejemplo de una sentencia CREATE PROCEDURE que utiliza el contenido de CURRENT SQL_CCFLAGS.

```
CREATE PROCEDURE CHECK_YEARS (IN YEARS_WORKED INTEGER)
BEGIN
  IF __DB2V97__ THEN
    IF YEARS_WORKED > __MAX_YEARS THEN
      ...
    END IF;
  END
```

La directiva de consulta __DB2V97__ se utiliza como valor booleano para determinar si el código puede incluirse. El valor de constante 50 sustituye la directiva de consulta __MAX_YEARS durante la compilación.

Acerca de las sentencias de control de SQL

Las sentencias de control de SQL, también denominadas lenguaje de procedimientos SQL (SQL PL), son sentencias de SQL que permiten que el SQL pueda utilizarse de forma similar a la escritura de un programa en un lenguaje de programación estructurado.

Las sentencias de control de SQL ofrecen la posibilidad de controlar el flujo lógico, declarar y establecer variables y manejar avisos y excepciones. Algunas sentencias de control de SQL incluyen otras sentencias de SQL anidadas. Las sentencias de control de SQL pueden utilizarse en el cuerpo de una rutina, un activador o una sentencia compuesta.

Referencias a parámetros de SQL, variables de SQL y variables globales

Puede hacerse referencia a los parámetros SQL, las variables de SQL y a las variables globales en cualquier punto de la sentencia de procedimiento SQL donde pueda especificarse una expresión o una variable.

No pueden especificarse variables del lenguaje principal en las rutinas de SQL, activadores de SQL o sentencias compuestas dinámicas. Puede hacerse referencia a los parámetros de SQL en cualquier punto del cuerpo de la rutina, y pueden calificarse con el nombre de la rutina. Puede hacerse referencia a las variables de SQL en cualquier punto de la sentencia compuesta en la que se han declarado y pueden calificarse con el nombre de etiqueta que se ha especificado al principio de la sentencia compuesta. Si un parámetro o una variable de SQL tiene un tipo de datos de fila, se puede hacer referencia a los campos en cualquier lugar donde se pueda hacer referencia a un parámetro o una variable de SQL. Puede hacerse referencia a las variables globales de cualquier expresión siempre y cuando la expresión no tenga que ser determinante. Los escenarios siguientes requieren expresiones determinantes, que impiden el uso de variables globales:

- Restricciones de comprobación
- Definiciones de columnas generadas
- Renovar MQT inmediatas

Todos los parámetros de SQL, las variables de SQL, los campos de variables de filas y las variables globales se consideran anulables. El nombre de un parámetro de SQL, de una variable de SQL, de un campo de variable de fila o de una variable global de una rutina de SQL puede ser igual al nombre de una columna de una tabla o vista a la que se haga referencia en la rutina. El nombre de una variable de SQL o un campo de variable de fila también puede ser el mismo que el nombre de otra variable de SQL o campo de variable de fila declarado en la misma rutina. Esto puede ocurrir cuando las dos variables de SQL están declaradas en sentencias compuestas diferentes. La sentencia compuesta que contiene la declaración del nombre de una variable de SQL determina el ámbito del nombre de dicha variable. Para obtener más información, consulte “SQL compuesto (procedimiento)”.

El nombre de una variable de SQL o un parámetro de SQL de una rutina de SQL puede ser igual al nombre de un identificador que se ha utilizado en determinadas sentencias de SQL. Si el nombre no se ha calificado, las normas siguientes describen si el nombre hace referencia al identificador o bien al parámetro de SQL o a la variable de SQL:

- En las sentencias SET PATH y SET SCHEMA, se comprueba si el nombre es un parámetro de SQL o una variable de SQL. Si no se encuentra como variable de SQL o parámetro de SQL, se utiliza como identificador.
- En las sentencias CONNECT, DISCONNECT, RELEASE y SET CONNECTION, se utiliza el nombre como identificador.

Los nombres idénticos se deben calificar explícitamente. Si se califica claramente un nombre, ello indica si el nombre se refiere a una columna, una variable de SQL, un parámetro de SQL, un campo de variable de fila o una variable global. Si el nombre no se ha calificado, o se ha calificado pero sigue siendo ambiguo, las normas siguientes describen si el nombre hace referencia a una columna, a una variable de SQL, a un parámetro de SQL o a una variable global:

- Si las tablas y vistas que se han especificado en el cuerpo de la rutina de SQL existen en el momento de crearse la rutina, primero se comprueba si el nombre

Referencias a parámetros de SQL, variables de SQL y variables globales

es un nombre de columna. Si no se encuentra como columna, se comprueba si es como variable de SQL en la sentencia compuesta y, a continuación, si es como parámetro de SQL y, finalmente, se comprueba si es como variable global.

- Si las tablas o vistas a las que se hace referencia no existen en el momento de crearse la rutina, primero se comprueba si el nombre es una variable de SQL de la sentencia compuesta, a continuación como parámetro de SQL y luego como variable global. La variable puede declararse dentro de la sentencia compuesta que contiene referencia o dentro de una sentencia compuesta en la que dicha sentencia compuesta esté anidada. Si dos variables de SQL están dentro del mismo ámbito y tienen el mismo nombre, cosa que puede ocurrir si están declaradas en sentencias compuestas distintas, se utiliza la variable de SQL declarada en la sentencia compuesta más interior. Si no se encuentra, se da por supuesto que es una columna.

Referencias a etiquetas de SQL

Se pueden especificar etiquetas en la mayoría de las sentencias de procedimiento de SQL.

La sentencia compuesta que contiene la sentencia que define una etiqueta determina el ámbito del nombre de dicha etiqueta. El nombre de una etiqueta debe ser exclusivo dentro de la sentencia compuesta en la que está definida, incluidas las etiquetas definidas en sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). La etiqueta no puede ser la misma que la etiqueta especificada en la sentencia compuesta (SQLSTATE 42734) ni la misma que el nombre de la rutina que contiene la sentencia de procedimiento SQL (SQLSTATE 42734).

Sólo se puede hacer referencia al nombre de una etiqueta dentro de la sentencia compuesta en la que está definida, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta. Una etiqueta se puede utilizar para calificar el nombre de una variable de SQL o se puede especificar como destino de una sentencia GOTO, LEAVE, o ITERATE.

Referencias a nombres de condición de SQL

El nombre de una condición de SQL puede ser el mismo que el de otra condición de SQL declarada en la misma rutina.

Esto puede ocurrir cuando las dos condiciones de SQL están declaradas en sentencias compuestas diferentes. La sentencia compuesta que contiene la declaración del nombre de una condición de SQL determina el ámbito del nombre de dicha condición. El nombre de una condición debe ser exclusivo dentro de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una condición dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta. Cuando se hace referencia a un nombre de condición, se utiliza la condición declarada en la sentencia compuesta más interior. Para obtener más información, consulte "SQL compuesto (en línea)".

Referencias a nombres de sentencia de SQL

El nombre de una sentencia de SQL puede ser el mismo que el de otra sentencia de SQL declarada en la misma rutina.

Esto puede ocurrir cuando las dos sentencias de SQL están declaradas en sentencias compuestas diferentes. La sentencia compuesta que contiene la declaración del nombre de una sentencia de SQL determina el ámbito del nombre de dicha sentencia. El nombre de una sentencia debe ser exclusivo dentro de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una sentencia dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta. Cuando se hace referencia a un nombre de sentencia, se utiliza la sentencia declarada en la sentencia compuesta más interior. Para obtener más información, consulte “SQL compuesto (en línea)”.

Referencias a nombre de cursor de SQL

Los nombres de cursor incluyen los nombres de cursores declarados y los nombres de las variables de cursor.

El nombre de un cursor de SQL puede ser el mismo que el de otro cursor de SQL declarado en la misma rutina. Esto puede ocurrir cuando los dos cursores de SQL están declarados en sentencias compuestas diferentes.

La sentencia compuesta que contiene la declaración del nombre de un cursor de SQL determina el ámbito del nombre de dicho cursor. El nombre de un cursor debe ser exclusivo dentro de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de un cursor dentro de la sentencia compuesta en la que está declarado, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta. Cuando se hace referencia a un nombre de cursor, se utiliza el cursor declarado en la sentencia compuesta más interior. Para obtener más información, consulte “SQL compuesto (en línea)”.

Si el constructor de cursor asignado a una variable de cursor contiene una referencia a una variable de SQL local, todas las sentencias OPEN que emplean la variable de cursor deben encontrarse dentro del ámbito en el que se declaró la variable de SQL local.

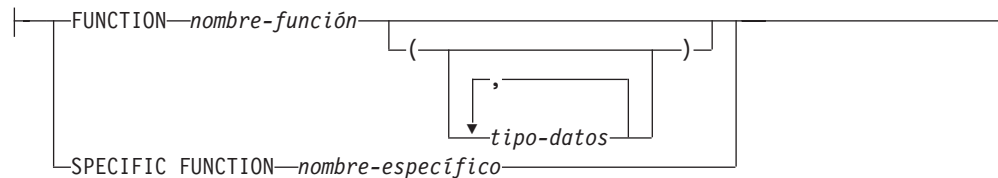
Designadores de función, método y procedimiento

En este tema se describen fragmentos de sintaxis que se utilizan para identificar de forma exclusiva una función, método o procedimiento que no se ha definido en un módulo.

Designador de función

Un designador de función identifica una única función de forma exclusiva. Por lo general, los designadores de función aparecen en sentencias de DDL de las funciones (como DROP o ALTER). Un designador de función no debe identificar una función de módulo (SQLSTATE 42883).

designador-función:



FUNCTION nombre-función

Identifica una función en particular y sólo es válido si existe exactamente una instancia de función con el nombre *nombre-función* en el esquema. Para la función identificada puede definirse cualquier número de parámetros. 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. 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 de la función en el esquema especificado o implícito, se genera un error (SQLSTATE 42725).

FUNCTION nombre-función (tipo-datos,...)

Proporciona la signatura de la función, que identifica la función de forma exclusiva. No se utiliza el algoritmo de resolución de función.

nombre-función

Especifica el nombre de la función. 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.

(*tipo-datos*,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) en la sentencia CREATE FUNCTION. Para identificar la instancia de función específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando 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.

Designadores de función, método y procedimiento

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (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 que se ha 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 en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

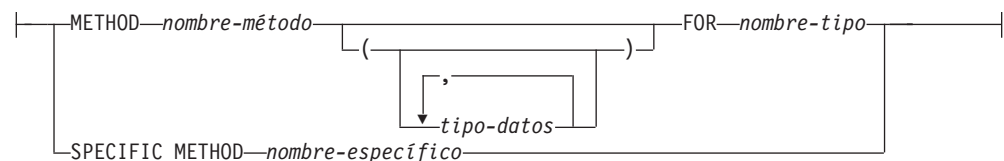
SPECIFIC FUNCTION *nombre-específico*

Identifica una función definida por el usuario en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación de la función. 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. 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).

Designador de método

Un designador de método identifica un único método de forma exclusiva. Por lo general, los designadores de método aparecen en las sentencias de DDL de los métodos (como, por ejemplo, DROP o ALTER).

designador-método:



METHOD *nombre-método*

Identifica un método en particular y sólo es válido si existe exactamente una instancia de método con el nombre *nombre-método* para el tipo *nombre-tipo*. Para el método identificado puede definirse cualquier número de parámetros. Si no existe ningún método con este nombre para el tipo, se generará un error (SQLSTATE 42704). Si existe más de una instancia del método para el tipo, se generará un error (SQLSTATE 42725).

METHOD *nombre-método (tipo-datos, ...)*

Proporciona la signatura del método, que identifica de forma exclusiva al método. No se utiliza el algoritmo de resolución de método.

nombre-método

Especifica el nombre del método para el tipo *nombre-tipo*.

Designadores de función, método y procedimiento

(tipo-datos,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) en la sentencia CREATE TYPE. Para identificar la instancia de método específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando 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.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE).

Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el valor especificado en la sentencia CREATE TYPE.

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha 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 con la signatura especificada para el tipo en el esquema indicado o implícito, se generará un error (SQLSTATE 42883).

FOR *nombre-tipo*

Especifica el nombre del tipo al que va a asociarse el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). 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.

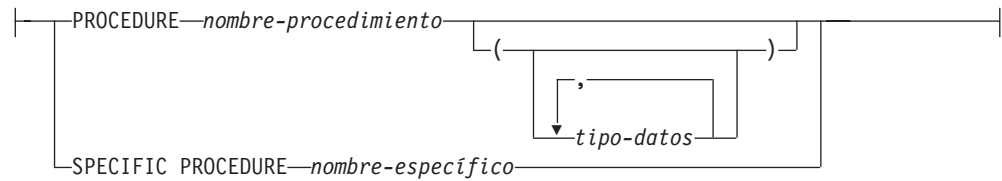
SPECIFIC METHOD *nombre-específico*

Identifica un método en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del método. 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. El *nombre-específico* debe identificar una instancia de método específica en el esquema indicado o implícito; en caso contrario, se genera un error (SQLSTATE 42704).

Designador de procedimiento

Un designador de procedimiento identifica un único procedimiento de forma exclusiva. Por lo general, los designadores de procedimiento aparecen en las sentencias de DDL de los procedimientos (como DROP o ALTER). Un designador de procedimiento no debe identificar un procedimiento de módulo (SQLSTATE 42883).

designador-procedimiento:



PROCEDURE nombre-procedimiento

Identifica un procedimiento en particular y sólo es válido si existe exactamente una instancia de procedimiento con el nombre *nombre-procedimiento* en el esquema. Para el procedimiento identificado puede definirse cualquier número de parámetros. 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. 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 del procedimiento en el esquema especificado o implícito, se genera un error (SQLSTATE 42725).

PROCEDURE nombre-procedimiento (*tipo-datos*,...)

Proporciona la signatura del procedimiento, que identifica el procedimiento de forma exclusiva. No se utiliza el algoritmo de resolución de procedimiento.

nombre-procedimiento

Especifica el nombre del procedimiento. 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.

(*tipo-datos*,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) en la sentencia CREATE PROCEDURE. Para identificar la instancia de procedimiento específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando 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.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE).

Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el valor especificado en la sentencia CREATE PROCEDURE.

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha 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.

Designadores de función, método y procedimiento

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 un procedimiento en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del procedimiento. 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. 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).

ALLOCATE CURSOR

La sentencia `ALLOCATE CURSOR` asigna un cursor para el conjunto de resultados identificado por la variable localizadora de conjuntos de resultados.

Para obtener más información acerca de las variables de localizador de conjuntos de resultados, consulte la descripción de la sentencia `ASSOCIATE LOCATORS`.

Invocación

Esta sentencia sólo puede incorporarse en un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización

No se necesita.

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 normas para declarar variables localizadoras de conjuntos de resultados. Para obtener más información acerca de la declaración de variables de SQL, consulte el apartado “Sentencia de SQL compuesto (procedimiento)”.

La variable localizadora de conjuntos de resultados debe contener un valor válido, tal como lo devuelve la sentencia `ASSOCIATE LOCATORS` de SQL (SQLSTATE 0F001).

Normas

- Son aplicables las normas siguientes cuando se utiliza un cursor asignado:
 - Un cursor asignado no se puede abrir con la sentencia `OPEN` (SQLSTATE 24502).
 - Un cursor asignado no puede utilizarse en una sentencia `UPDATE` o `DELETE` con posición (SQLSTATE 42828).
 - Un cursor asignado se puede cerrar con la sentencia `CLOSE`. El cierre de un cursor asignado cierra el cursor asociado.
 - Sólo se puede asignar un único cursor a cada conjunto de resultados.
- Los cursores asignados permanecen vigentes hasta que se ejecuta una operación de retrotracción, un cierre implícito o un cierre explícito.
- Una operación de confirmación destruye los cursores asignados que no se han definido como `WITH HOLD`.
- La destrucción de un cursor asignado cierra el cursor asociado existente en el procedimiento SQL.

ALLOCATE CURSOR

Ejemplo

Este ejemplo de procedimiento SQL define y asocia el cursor C1 a una variable localizadora de conjuntos de resultados, LOC1, y con el conjunto de resultados asociado devuelto por el procedimiento SQL:

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

ALTER AUDIT POLICY

La sentencia ALTER AUDIT POLICY modifica la definición de una política de auditoría en el servidor actual.

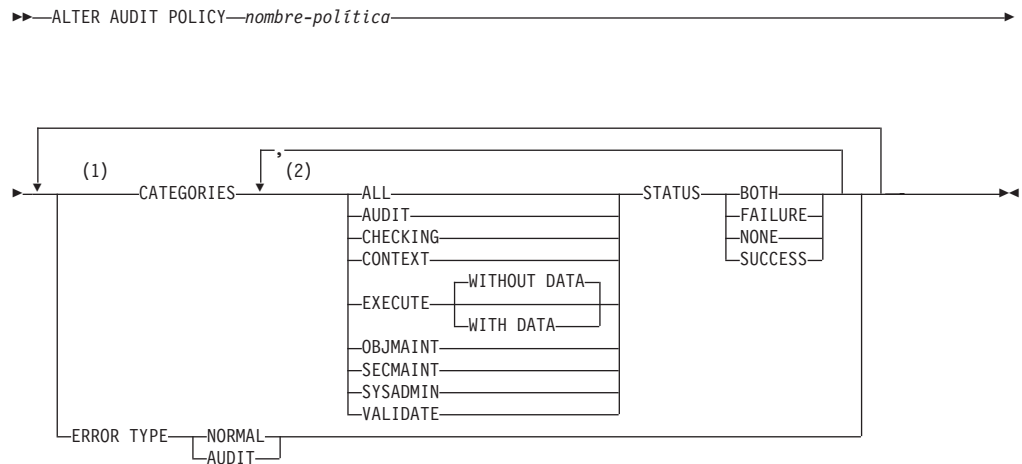
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Notas:

- 1 Cada una de las cláusulas CATEGORIES y ERROR TYPE puede especificarse una vez como máximo (SQLSTATE 42614).
- 2 Cada categoría puede especificarse una vez como máximo (SQLSTATE 42614) y no puede especificarse ninguna otra categoría si se especifica ALL (SQLSTATE 42601).

Descripción

nombre-política

Identifica la política de auditoría que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre debe identificar exclusivamente una política de auditoría en el servidor actual (SQLSTATE 42704).

CATEGORIES

Una lista de una o más categorías de auditoría para las que se especifica un nuevo valor de estado. Si no se especifica ALL, el STATUS de cualquier categoría que no se especifique explícitamente permanece sin modificar.

ALTER AUDIT POLICY

ALL

Establece todas las categorías en el mismo estado. La categoría EXECUTE es WITHOUT DATA.

AUDIT

Genera registros cuando se modifican los valores de auditoría o cuando se accede a las anotaciones cronológicas de auditoría.

CHECKING

Genera registros durante la comprobación de autorización de intentos de acceder o manipular funciones u objetos de base de datos.

CONTEXT

Genera registros para mostrar el contexto de operación cuando se realiza una operación de base de datos.

EXECUTE

Genera registros para mostrar la ejecución de sentencias de SQL.

WITHOUT DATA o WITH DATA

Especifica si se proporcionan valores de datos de entrada para las variables del lenguaje principal y deben registrarse marcadores de parámetro como parte de la categoría EXECUTE.

WITHOUT DATA

Se proporcionan valores de datos de entrada para las variables del lenguaje principal y no se registran marcadores de parámetro como parte de la categoría EXECUTE.

WITH DATA

Se proporcionan valores de datos de entrada para las variables del lenguaje principal y se registran marcadores de parámetro como parte de la categoría EXECUTE. No se registran todos los valores de entrada; específicamente, los parámetros de tipo estructurado, LOB, LONG y XML aparecen como valor nulo. Los campos de fecha, hora e indicación de fecha y hora se registran en formato ISO. Los valores de datos de entrada se convierten a la página de códigos de base de datos antes de registrarse. Si falla la conversión de página de códigos, no se devuelven errores y se registran los datos no convertidos.

OBJMAINT

Genera registros cuando se crean o descartan los objetos de datos.

SECMAINT

Genera registros cuando se otorgan o revocan privilegios de objeto, privilegios de base de datos o la autorización DBADM. También se generan registros cuando se modifican los parámetros de configuración de seguridad del gestor de base de datos **grupo_sysadm**, **grupo_sysctrl** o **grupo_sysmaint**.

SYSADMIN

Genera registros cuando se realizan operaciones que requieren autorización SYSADM, SYSMAINT o SYSCTRL.

VALIDATE

Genera registros cuando se autentifican usuarios o cuando se recupera información de seguridad del sistema relacionada con un usuario.

STATUS

Especifica un estado para la categoría especificada.

BOTH

Se auditarán los sucesos satisfactorios y anómalos.

FAILURE

Sólo se auditarán los sucesos anómalos.

SUCCESS

Sólo se auditarán los sucesos satisfactorios.

NONE

No se auditarán los sucesos de esta categoría.

ERROR TYPE

Especifica si van a devolverse o ignorarse los errores de auditoría.

NORMAL

Los errores generados por la auditoría se ignorarán y sólo se devolverán a la aplicación los SQLCODE para los errores asociados con la operación que se está realizando.

AUDIT

Se devuelven a la aplicación todos los errores, incluyendo los errores que se producen dentro del propio recurso de auditoría.

Normas

- Una sentencia de SQL exclusiva de AUDIT debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de AUDIT son:
 - AUDIT
 - CREATE AUDIT POLICY, ALTER AUDIT POLICY o DROP (AUDIT POLICY)
 - DROP (ROLE) o DROP (TRUSTED CONTEXT) si el rol o contexto fiable está asociado a una política de auditoría.
- Una sentencia de SQL exclusiva de AUDIT no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Sólo se permite una sentencia de SQL exclusiva de AUDIT sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva de AUDIT sin confirmar, las siguientes sentencias de SQL exclusivas de AUDIT esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de AUDIT actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Si la política de auditoría que se está modificando está asociada en la actualidad a un objeto de base de datos, los cambios no surtirán efecto hasta que el cambio afecte a la siguiente unidad de trabajo para la aplicación. Por ejemplo, si se utiliza la política de auditoría para la base de datos, ninguna unidad de trabajo actual verá el cambio para la política hasta después de que se complete una sentencia COMMIT o ROLLBACK para dicha unidad de trabajo.

Ejemplo

Modifique las categorías SECMAINT, CHECKING y VALIDATE de una política de auditoría denominada DBAUDPRF para auditar éxitos y errores.

ALTER AUDIT POLICY

```
ALTER AUDIT POLICY DBAUDPRF  
  CATEGORIES SECMAINT STATUS BOTH,  
             CHECKING STATUS BOTH,  
             VALIDATE STATUS BOTH
```

ALTER BUFFERPOOL

La sentencia ALTER BUFFERPOOL se utiliza para modificar las características o el comportamiento de una agrupación de almacenamientos intermedios.

La sentencia ALTER BUFFERPOOL puede modificar una agrupación de almacenamientos intermedios de las formas siguientes:

- Modificar el tamaño de la agrupación de almacenamientos intermedios en todos los miembros o en un solo miembro
- Habilitar o inhabilitar el redimensionamiento automático de la agrupación de almacenamientos intermedios
- Añadir esta definición de agrupación de almacenamientos intermedios a un nuevo grupo de particiones de base de datos
- Modificar el área de bloque de la agrupación de almacenamientos intermedios para la E/S basada en bloques

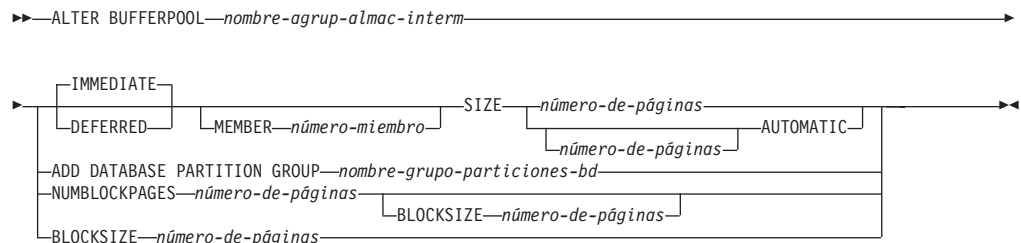
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis



Descripción

nombre-agrup-almac-interm

Indica el nombre de la agrupación de almacenamientos intermedios. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). Debe ser una agrupación de almacenamientos intermedios descrita en el catálogo.

IMMEDIATE o DEFERRED

Indica si el tamaño de la agrupación de almacenamientos intermedios cambiará de forma inmediata.

IMMEDIATE

El tamaño de la agrupación de almacenamientos intermedios cambiará de forma inmediata. Si no existe suficiente espacio reservado en la memoria compartida de base de datos para asignar nuevo espacio (SQLSTATE 01657), la sentencia se ejecutará como DEFERRED.

ALTER BUFFERPOOL

DEFERRED

El tamaño de la agrupación de almacenamientos intermedios cambiará al reactivar la base de datos (todas las aplicaciones deben desconectarse de la base de datos). No es necesario tener espacio de memoria reservado; la base de datos DB2 asignará la memoria necesaria del sistema en el momento de la activación.

MEMBER *número-miembro*

Especifica el miembro en el que se modifica el tamaño de la agrupación de almacenamientos intermedios. Se crea una entrada de excepción en la vista de catálogo SYSCAT.BUFFERPOOLEXCEPTIONS. El miembro debe encontrarse en uno de los grupos de particiones de base de datos para la agrupación de almacenamientos intermedios (SQLSTATE 42729). Si la cláusula no se especifica, el tamaño de la agrupación de almacenamientos intermedios se modifica en todos los miembros con la excepción de aquellas que presentan una entrada de excepción en SYSCAT.BUFFERPOOLEXCEPTIONS.

SIZE

Especifica un tamaño nuevo para la agrupación de almacenamientos intermedios, o habilita o inhabilita el ajuste automático para esta agrupación de almacenamientos intermedios.

número-de-páginas

Número de páginas para el nuevo tamaño de la agrupación de almacenamientos intermedios. Si la agrupación de almacenamientos intermedios ya es una agrupación de almacenamientos intermedios de ajuste automático y se especifica la cláusula *SIZE número-de-páginas*, la operación de modificación inhabilita la función de ajuste automático para esta agrupación de almacenamientos intermedios.

AUTOMATIC

Habilita el ajuste automático para la agrupación de almacenamientos intermedios. El gestor de bases de datos ajusta el tamaño de la agrupación de almacenamientos intermedios como respuesta a los requisitos de carga de trabajo. Si se especifica el número de páginas, el tamaño actual de la agrupación de almacenamientos intermedios se establece con ese valor, a menos que la palabra clave diferida también se especifique, en cuyo caso se omitirá el número de páginas. Tenga en cuenta que el gestor de memoria de ajuste automático (STMM) aplica un tamaño mínimo para las agrupaciones de almacenamientos intermedios automáticas y que cualquier tamaño especificado es un valor de una vez y que cuando se vuelva a activar posteriormente la base de datos, el tamaño de la agrupación de almacenamientos intermedios estará basado en el último valor de ajuste. Para determinar el tamaño actual de las agrupaciones de almacenamientos intermedios que se han habilitado para el autoajuste, utilice el mandato **GET SNAPSHOT** y examine el tamaño actual de las agrupaciones de almacenamientos intermedios (el valor del elemento de supervisor **bp_cur_buffsz**). Si se especifica **AUTOMATIC**, la cláusula **MEMBER** no puede especificarse (SQLSTATE 42601).

ADD DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*

Añade este grupo de particiones de base de datos a la lista de grupos de particiones de base de datos a la que se aplica la definición de agrupación de almacenamientos intermedios. Para cualquier miembro del grupo de particiones de base de datos que todavía no tenga la agrupación de almacenamientos intermedios definida, para crear dicha agrupación en el miembro se utiliza el tamaño por omisión especificado para la agrupación de almacenamientos intermedios. Los espacios de tablas de *nombre-grupo-*

particiones-bd pueden especificar esta agrupación de almacenamientos intermedios. El grupo de particiones de base de datos debe existir actualmente en la base de datos (SQLSTATE 42704).

NUMBLOCKPAGES *número-de-páginas*

Especifica el número de páginas que deben existir en el área basada en bloques. El número de páginas no debe superar el 98 por ciento del número de páginas para la agrupación de almacenamientos intermedios (SQLSTATE 54052). Si se especifica el valor 0, se inhabilita la E/S de bloque. El valor real de NUMBLOCKPAGES utilizado será un múltiplo de BLOCKSIZE.

No se da soporte a NUMBLOCKPAGES en un entorno DB2 pureScale (SQLSTATE 56038).

BLOCKSIZE *número-de-páginas*

Especifica el número de páginas de un bloque. El tamaño de bloque debe ser un valor comprendido entre el 2 y el 256 (SQLSTATE 54053). El valor por omisión es 32.

BLOCKSIZE no se soporta en un entorno DB2 pureScale (SQLSTATE 56038).

Notas

- Sólo el tamaño de agrupación de almacenamientos intermedios puede cambiarse dinámicamente (inmediatamente). Todos los demás cambios se aplican de forma diferida y sólo entrarán en vigor tras la reactivación de la base de datos.
- Si la sentencia se ejecuta como sentencia diferida, aunque la definición de agrupación de almacenamientos intermedios sea transaccional y los cambios realizados en la definición de agrupación de almacenamientos intermedios aparezcan reflejados en las tablas de catálogo durante la confirmación, ningún cambio realizado en la agrupación de almacenamientos intermedios real entrará en vigor 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 que se han creado en los espacios de tablas de los nuevos grupos de particiones de base de datos utilizarán la agrupación de almacenamientos intermedios por omisión. Por omisión, la sentencia es IMMEDIATE cuando se aplica dicha palabra clave.
- 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.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP

ALTER DATABASE PARTITION GROUP

La sentencia ALTER DATABASE PARTITION GROUP se utiliza para añadir una o varias particiones de base de datos a un grupo de particiones de base de datos o para descartar una o varias particiones de base de datos de un grupo de particiones de base de datos.

Invocación

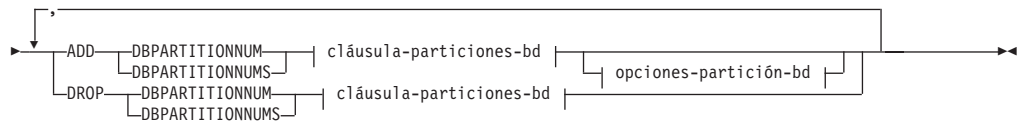
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

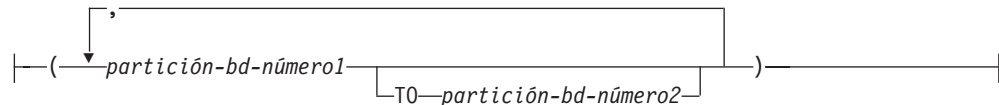
El ID de autorización de la sentencia debe tener autorización SYSCTRL o SYSADM.

Sintaxis

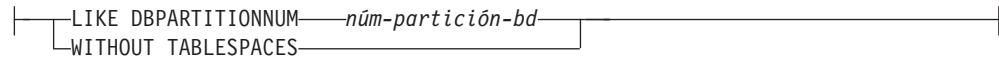
→ ALTER DATABASE PARTITION GROUP *nombre-partición-base-datos* →



cláusula-particiones-bd:



opciones-partición-bd:



Descripción

nombre-partición-base-datos

Indica el nombre del grupo de particiones de base de datos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). Debe ser un grupo de particiones de base de datos descrito en el catálogo. No se puede especificar IBMCATGROUP ni IBMTEMPGROUP (SQLSTATE 42832).

ADD DBPARTITIONNUM

Especifica las particiones de base de datos específicas que deben añadirse al grupo de particiones de base de datos. DBPARTITIONNUMS es sinónimo de DBPARTITIONNUM. Las particiones de base de datos especificadas no deben estar definidas en el grupo de particiones de base de datos (SQLSTATE 42728).

DROP DBPARTITIONNUM

Especifica las particiones de base de datos específicas que deben descartarse del grupo de particiones de base de datos. DBPARTITIONNUMS es sinónimo de DBPARTITIONNUM. Las particiones de base de datos especificadas deben estar definidas en el grupo de particiones de base de datos (SQLSTATE 42729).

cláusula-particiones-bd

Especifica las particiones de base de datos que deben añadirse o descartarse.

partición-bd-núm-1

Especifica un número de partición de base de datos específica.

TO *partición-bd-número2*

Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9).

opciones-partición-bd

LIKE DBPARTITIONNUM *núm-partición-bd*

Especifica que los contenedores de los espacios de tablas existentes del grupo de particiones de base de datos serán iguales a los contenedores del *número-partición-base-de-datos* especificado. La partición de base de datos especificada debe ser una partición que ya exista en el grupo de particiones de base de datos antes de esta sentencia, y que no esté incluida en una cláusula DROP DBPARTITIONNUM de la misma sentencia.

Para los espacios de tablas que están definidos para utilizar el almacenamiento automático (es decir, espacios de tablas creados con la cláusula MANAGED BY AUTOMATIC STORAGE de la sentencia CREATE TABLESPACE, o para las que no se ha especificado ninguna cláusula MANAGED BY), los contenedores no coincidirán necesariamente con los de la partición especificada. En lugar de ello, el gestor de bases de datos asignará automáticamente los contenedores en función de las vías de acceso de almacenamiento asociadas con la base de datos, y esto puede dar lugar o no al uso de los mismos contenedores. El tamaño de cada espacio de tablas se basa en el tamaño inicial especificado al crear la tabla, y es posible que no coincida con el tamaño actual del espacio de tablas de la partición especificada.

WITHOUT TABLESPACES

Especifica que los contenedores para los espacios de tablas existentes en el grupo de particiones de base de datos no se crean en las particiones de base de datos añadidas recientemente. La sentencia ALTER TABLESPACE que utiliza la *cláusula-particiones-bd* o MANAGED BY AUTOMATIC STORAGE debe utilizarse para definir los contenedores que deben utilizarse con los espacios de tablas que se han definido en este grupo de particiones de base de datos. Si no se especifica esta opción, los contenedores por omisión se especifican en las particiones de base de datos añadidas recientemente para cada espacio de tablas definido en el grupo de particiones de base de datos.

Esta opción se pasa por alto para los espacios de tablas que se han definido para utilizar el almacenamiento automático (es decir, los espacios de tablas que se han creado con la cláusula MANAGED BY AUTOMATIC STORAGE de la sentencia CREATE TABLESPACE, o para los que no se ha especificado ninguna cláusula MANAGED BY). No existe ninguna forma de diferir la creación de los contenedores para estos espacios de tablas. El gestor de bases de datos asignará los contenedores automáticamente en función de las vías de acceso de almacenamiento asociadas con la base de

ALTER DATABASE PARTITION GROUP

datos. El tamaño de cada espacio de tablas se basará en el tamaño inicial especificado al crear el espacio de tablas.

Normas

- Cada partición de base de datos especificada por número debe estar definida en el archivo `db2nodes.cfg` (SQLSTATE 42729).
- Cada *núm-partición-bd* listado en la *cláusula-particiones-bd* debe ser para una partición de base de datos exclusiva (SQLSTATE 42728).
- Un número de partición de base de datos válido oscila entre 0 y 999, ambos inclusive (SQLSTATE 42729).
- Una partición de base de datos no puede aparecer en la cláusula ADD y en la cláusula DROP al mismo tiempo (SQLSTATE 42728).
- Debe haber como mínimo una partición de base de datos que permanezca en el grupo de particiones de base de datos. La última partición de base de datos de un grupo de particiones de base de datos no puede descartarse (SQLSTATE 428C0).
- Si no se especifica la cláusula LIKE DBPARTITIONNUM ni la cláusula WITHOUT TABLESPACES al añadir una partición de base de datos, por omisión se utiliza el número de partición de base de datos más bajo de las particiones de base de datos existentes en el grupo de particiones de base de datos (es decir, el 2) y se procede como si se hubiera especificado LIKE DBPARTITIONNUM 2. Para que una partición de base de datos existente se utilice como partición por omisión, debe tener contenedores definidos para todos los espacios de tablas del grupo de particiones de base de datos (la columna IN_USE de SYSCAT.DBPARTITIONGROUPDEF no es 'T').
- La sentencia ALTER DATABASE PARTITION GROUP puede resultar anómala (SQLSTATE 55071) si una petición para añadir un servidor de particiones de base de datos está pendiente o en curso. Esta sentencia puede también resultar anómala (SQLSTATE 55077) si se añade en línea un servidor de particiones de base de datos nuevo a la instancia y no todas las aplicaciones saben de la existencia del servidor de particiones de base de datos nuevo.

Notas

- Cuando se añade una partición de base de datos a un grupo de particiones de base de datos, se crea una entrada de catálogo para la partición de base de datos (consulte SYSCAT.DBPARTITIONGROUPDEF). La correlación de distribución cambia inmediatamente para incluir la nueva partición de base de datos, junto con un indicador (IN_USE) que indica que la partición de base de datos se encuentra en la correlación de distribución si se cumple una de las condiciones siguientes:

- No se ha definido ningún espacio de tablas para el grupo de particiones de base de datos, o bien
- No se ha definido ninguna tabla en los espacios de tablas definidos en el grupo de particiones de base de datos y no se ha especificado la cláusula WITHOUT TABLESPACES.

La correlación de distribución no cambia y el indicador (IN_USE) se establece para indicar que la partición de base de datos no se incluye en la correlación de distribución si se cumple una de las condiciones siguientes:

- Existen tablas en los espacios de tablas en el grupo de particiones de base de datos, o bien
- Existen espacios de tablas en el grupo de particiones de base de datos y se ha especificado la cláusula WITHOUT TABLESPACES (a menos que todos los

espacios de tablas estén definidos para utilizar el almacenamiento automático, en cuyo caso la cláusula WITHOUT TABLESPACES se pasa por alto)

Para cambiar la correlación de distribución, debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP. Este mandato redistribuye los datos, cambia la correlación de distribución y cambia el indicador. Si se especifica la cláusula WITHOUT TABLESPACES, antes de intentar redistribuir los datos deben añadirse contenedores de espacios de tablas.

- Cuando se descarta una partición de base de datos de un grupo de particiones de base de datos, la entrada de catálogo para la partición de base de datos (consulte SYSCAT.DBPARTITIONGROUPDEF) se actualiza. Si no hay tablas definidas en los espacios de tablas definidos en el grupo de particiones de base de datos, la correlación de distribución cambia de forma inmediata para excluir la partición de base de datos descartada, y se descarta la entrada para la partición de base de datos del grupo de particiones de base de datos. Si existen tablas, la correlación de distribución no cambia y el indicador (IN_USE) se establece para indicar que la partición de base de datos está a la espera de ser descartada. Para redistribuir los datos y descartar la entrada para la partición de base de datos del grupo de particiones de base de datos, debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODE puede especificarse en lugar de DBPARTITIONNUM
 - NODES puede especificarse en lugar de DBPARTITIONNUMS
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP

Ejemplo

Supongamos que tiene una base de datos de seis particiones con las particiones siguientes: 0, 1, 2, 5, 7 y 8. Se añaden dos particiones de base de datos (3 y 6) al sistema.

- *Ejemplo 1:* supongamos que desea añadir las particiones de base de datos 3 y 6 a un grupo de particiones de base de datos denominado MAXGROUP, y que tiene contenedores de espacios de tablas como los de la partición de base de datos 2. La sentencia sería la siguiente:

```
ALTER DATABASE PARTITION GROUP MAXGROUP
ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- *Ejemplo 2:* supongamos que desea descartar la partición de base de datos 1 y añadir la partición de base de datos 6 al grupo de particiones de base de datos denominado MEDGROUP. Definirá los contenedores de espacios de tablas por separado para la partición de base de datos 6 utilizando ALTER TABLESPACE. La sentencia es la siguiente:

```
ALTER DATABASE PARTITION GROUP MEDGROUP
ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
DROP DBPARTITIONNUM(1)
```

ALTER DATABASE

La sentencia ALTER DATABASE añade nuevas vías de acceso de almacenamiento a la colección de vías de acceso que se utilizan para los espacios de tablas de almacenamiento automático, o elimina vías de acceso de almacenamiento existentes de dicha colección.

Un espacio de tablas de almacenamiento automático es un espacio de tablas que se ha creado utilizando almacenamiento automático; es decir, se ha especificado la cláusula MANAGED BY AUTOMATIC STORAGE en la sentencia CREATE TABLESPACE o no se ha especificado ninguna cláusula MANAGED BY. Si una base de datos está habilitada para el almacenamiento automático, las características de los contenedores y de gestión del espacio correspondientes a sus espacios de tablas pueden determinarse completamente a través del gestor de bases de datos. Si la base de datos no está habilitada actualmente para almacenamiento automático, el acto de añadir vías de acceso de almacenamiento la habilitará.

Importante: Esta sentencia parámetro ha quedado en desuso y puede dejar de mantenerse en un release futuro. En su lugar, utilice las sentencias CREATE STOGROUP o ALTER STOGROUP.

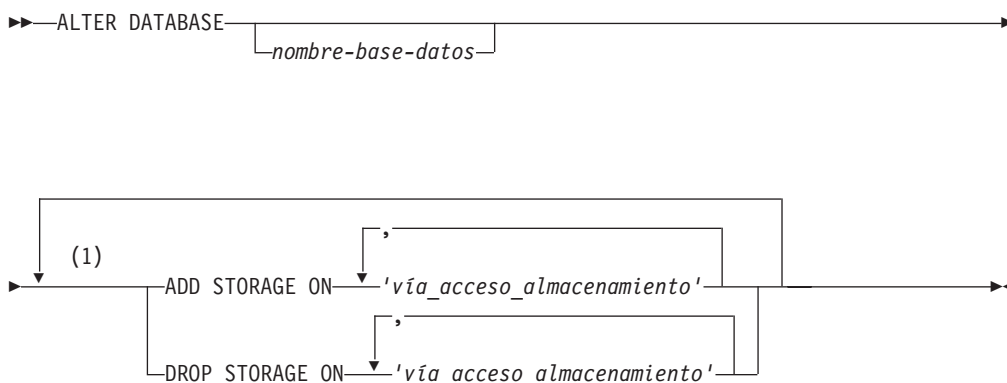
Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener autorización SYSADM o SYSCTRL.

Sintaxis



Notas:

- 1 Cada cláusula sólo puede especificarse una vez.

Descripción

nombre-base-datos

Un valor opcional que especifica el nombre de la base de datos que debe

modificarse. Si se especifica, el valor debe coincidir con el nombre de la base de datos a la que está conectada actualmente la aplicación (no el alias que el cliente puede tener catalogado); de lo contrario, se devolverá un error (SQLSTATE 42961).

ADD STORAGE ON

Especifica que una o más vías de acceso de almacenamiento nuevas se añadirán a la colección de vías de acceso de almacenamiento que se utilizan para los espacios de tablas de almacenamiento automático.

'vía_acceso_almacenamiento'

Constante de tipo serie que especifica una vía de acceso absoluta o la letra de una unidad (sólo en los sistemas operativos Windows) donde se crearán los contenedores para los espacios de tablas de almacenamiento automático.

DROP STORAGE ON

Especifica que una o más vías de acceso de almacenamiento se eliminarán de la colección de vías de acceso de almacenamiento que se utilizan para los espacios de tablas de almacenamiento automático. Si los espacios de tablas están utilizando de forma activa una vía de acceso de almacenamiento que se va a descartar, el estado de la vía de acceso de almacenamiento cambia de "En uso" a "Descarte pendiente" y se impedirá que se utilice en el futuro la vía de acceso de almacenamiento.

'vía_acceso_almacenamiento'

Constante de serie que especifica una vía de acceso absoluta o la letra de una unidad (sólo en los sistemas operativos Windows).

Normas

- Para una base de datos que se ejecuta en Versión 10.1 o posterior, las operaciones de esta sentencia se aplican al grupo de almacenamiento por omisión para la base de datos. Si no se define ningún grupo de almacenamiento para la base de datos, se utiliza el nombre IBMSTOGROUP.
- La vía de acceso de almacenamiento que se está añadiendo debe ser válida según las normas de denominación y debe ser accesible (SQLSTATE 57019). De igual modo, en un entorno de base de datos particionada, la vía de acceso de almacenamiento debe existir y ser accesible en cada partición de base de datos (SQLSTATE 57019).
- La vía de acceso de almacenamiento que se está descartando debe existir actualmente en la base de datos (SQLSTATE 57019) y no puede tener ya el estado "Descarte pendiente" (SQLSTATE 55073).
- La base de datos habilitada para el almacenamiento automático debe tener una vía de acceso de almacenamiento como mínimo. No se pueden descartar todas las vías de acceso de almacenamiento de la base de datos (SQLSTATE 428HH).
- No se puede ejecutar la sentencia ALTER DATABASE mientras se está añadiendo un servidor de particiones de base de datos (SQLSTATE 55071).
- DROP STORAGE ON no se puede especificar en un entorno DB2 pureScale (SQLSTATE 56038).

Notas

- Cuando se añaden nuevas vías de acceso de almacenamiento:
 - Los espacios de tablas normales y grandes existentes que utilicen el almacenamiento automático no utilizarán estas nuevas vías de acceso inicialmente. El gestor de bases de datos puede elegir crear nuevos

contenedores de espacio de tablas en estas vías de acceso únicamente si se produce una condición de falta de espacio.

- Los espacios de tablas temporales existentes gestionados por almacenamiento automático no utilizan automáticamente las vías de acceso de almacenamiento nuevas. Se debe detener con normalidad la base de datos y reiniciarla para que los contenedores de estos espacios de tablas utilicen las nuevas vías de acceso de almacenamiento. Como alternativa, se pueden descartar y recrear los espacios de tablas temporales. Al crearlos, estos espacios de tablas utilizan automáticamente todas las vías de acceso de almacenamiento que disponen de espacio libre suficiente.
- Añadir vías de acceso de almacenamiento a la base de datos para habilitar el almacenamiento automático no hará que la base de datos convierta espacios de tablas no habilitados para el almacenamiento automático para utilizarlo.
- Aunque las operaciones ADD STORAGE y DROP STORAGE son operaciones anotadas cronológicamente, si se rehacen o no durante una operación de recuperación en avance dependerá de cómo se haya restaurado la base de datos. Si la operación de restauración no redefine las vías de acceso de almacenamiento asociadas a la base de datos, el registro de anotaciones cronológicas que contiene el cambio de las vías de acceso de almacenamiento se rehace y las vías de acceso de almacenamiento descritas en el registro de anotaciones cronológicas se añaden o descartan durante la operación de recuperación en avance. No obstante, si las vías de acceso de almacenamiento *se redefinen* durante la operación de restauración, la operación de recuperación en avance no rehará los registros de anotaciones cronológicas ADD STORAGE o DROP STORAGE, porque se supone que ya se han configurado las vías de acceso de almacenamiento.
- Cuando se calcula el espacio libre para una vía de acceso de almacenamiento en una partición de base de datos, el gestor de bases de datos comprueba la existencia de los siguientes directorios o puntos de montaje en la vía de acceso de almacenamiento, y utiliza la primera que se encuentre.

```
<vía acceso almacenamiento>/<nombre instancia>/NODE####/<nombre base datos>
<vía acceso almacenamiento>/<nombre instancia>/NODE####
<vía acceso almacenamiento>/<nombre instancia>
<vía acceso almacenamiento>
```

Donde:

- <vía acceso almacenamiento> es una vía de acceso de almacenamiento asociada con la base de datos
- <nombre instancia> es la instancia bajo la que reside la base de datos
- NODE#### corresponde al número de la partición de base de datos (por ejemplo, NODE0000 o NODE0001)
- <nombre base de datos> es el nombre de la base de datos

Los sistemas de archivos se pueden montar en un punto bajo la vía de acceso de almacenamiento, y el gestor de bases de datos reconocerá que la cantidad real de espacio libre disponible para los contenedores del espacio de tablas puede no ser la misma que la asociada con el directorio de la vía de acceso de almacenamiento.

Considere un ejemplo donde existan dos particiones de bases de datos lógicas en una máquina física, y haya una única vía de acceso de almacenamiento (/db2data). Cada partición de base de datos utilizará esta vía de acceso de almacenamiento, pero es posible que se desee aislar los datos de cada partición en su propio sistema de archivos. En este caso, se puede crear un sistema de archivos diferente para cada partición y se puede montar en /db2data/<instancia>/NODE####. Al crear contenedores en la vía de acceso de almacenamiento y determinar el espacio libre, el gestor de bases de datos no

recuperará la información sobre el espacio libre para /db2data, sino que la recuperará para el directorio /db2data/<instancia>/NODE#### correspondiente.

- En general, deben utilizarse las mismas vías de acceso de almacenamiento para cada partición en un entorno de base de datos particionada. Una excepción a esta regla se da en caso de que se utilicen expresiones de partición de base de datos en la vía de acceso de almacenamiento. Esto permite que el número de partición de base de datos quede reflejado en la vía de acceso de almacenamiento, de modo que el nombre de vía de acceso resultante es diferente para cada partición.
- Al descartar una vía de acceso de almacenamiento que están utilizando uno o más espacios de tablas, el estado de la vía de acceso cambia de “En uso” a “Descarte pendiente”. La vía de acceso no crecerá más. Antes de que se pueda eliminar completamente la vía de acceso de la base de datos, es necesario reequilibrar todos los espacios de tablas afectados (con la cláusula REBALANCE o la sentencia ALTER TABLESPACE), para que los datos de sus contenedores se retiren de la vía de acceso. El reequilibrio sólo se puede aplicar a espacios de tablas de tamaño normal y grande. Los espacios de tablas temporales deben descartarse y volverse a crear para que se retiren sus contenedores de la vía de acceso descartada. Cuando ningún espacio de tablas esté utilizando ya la vía de acceso, se eliminará físicamente de la base de datos.

Para las bases de datos particionadas, la vía de acceso se mantiene de forma independiente en cada partición. Cuando ya no se esté utilizando una vía de acceso en una partición de base de datos dada, se eliminará físicamente de dicha partición. Es posible que otras particiones muestren todavía la vía de acceso con estado “Descarte pendiente”.

Se puede obtener la lista de los espacios de tablas de almacenamiento automático que utilizan vías de acceso de almacenamiento con estado de descarte pendiente emitiendo la sentencia de SQL siguiente:

```
SELECT DISTINCT A.TBSP_NAME, A.TBSP_ID, A.TBSP_CONTENT_TYPE
FROM SYSIBMADM.SNAPTbsp A, SYSIBMADM.SNAPTbsp PART B
WHERE A.TBSP_ID = B.TBSP_ID AND B.TBSP_PATHS_DROPPED = 1
```

- Al descartar una vía de acceso de almacenamiento que se especificó inicialmente mediante una expresión de partición de base de datos, se debe utilizar en el descarte la misma serie de vía de acceso de almacenamiento, incluida la expresión de partición de base de datos. Si se especificó una expresión de partición de base de datos, se puede encontrar esta serie de vía de acceso en el elemento “Vía acceso con expresión partición bd” (db_storage_path_with_dpe) de una instantánea de base de datos. Este elemento no se muestra si no se incluyó una expresión de partición de base de datos en la vía de acceso original especificada.
- Se puede añadir varias veces una vía de acceso de almacenamiento dada a una base de datos. Al utilizar la cláusula DROP STORAGE ON, si se especifica esa vía de acceso concreta una vez se descartarán de la base de datos *todas* las instancias de la vía de acceso.

Ejemplos

- *Ejemplo 1:* Añadir dos vías de acceso bajo el directorio /db2 (/db2/filesystem1 y /db2/filesystem2) y una tercera vía de acceso llamada /filesystem3 a la ubicación para los espacios de tablas de almacenamiento automático asociados con la base de datos conectada actualmente.

```
ALTER DATABASE ADD STORAGE ON '/db2/filesystem1', '/db2/filesystem2',
'/filesystem3'
```

- *Ejemplo 2:* Añadir las unidades D y E al espacio para los espacios de tablas de almacenamiento automático que está asociado con la base de datos SAMPLE.

ALTER DATABASE

```
ALTER DATABASE SAMPLE ADD STORAGE ON 'D:', 'E:\'
```

- *Ejemplo 3:* Añadir el directorio F:\DB2DATA y la unidad G al espacio para los espacios de tablas de almacenamiento automático que está asociado con la base de datos conectada actualmente.

```
ALTER DATABASE ADD STORAGE ON 'F:\DB2DATA', 'G:'
```

- *Ejemplo 4:* Añadir una vía de acceso de almacenamiento que utilice una expresión de partición de base de datos para diferenciar las vías de acceso de almacenamiento en cada una de las particiones de base de datos.

```
ALTER DATABASE ADD STORAGE ON '/dataForPartition $N'
```

La vía de acceso de almacenamiento que se utilizaría en la partición 0 de la base de datos sería /dataForPartition0; en la partición 1 de la base de datos sería /dataForPartition1; y así sucesivamente.

- *Ejemplo 5:* Añadir vías de acceso de almacenamiento a una base de datos que no esté habilitada para el almacenamiento automático, con el objetivo de habilitar el almacenamiento automático para esa base de datos.

```
CREATE DATABASE MYDB AUTOMATIC STORAGE NO  
CONNECT TO MYDB  
ALTER DATABASE ADD STORAGE ON '/db2/filesystem1', '/db2/filesystem2'
```

La base de datos MYDB está ahora habilitada para el almacenamiento automático.

- *Ejemplo 6:* Eliminar las vías de acceso /db2/filesystem1 y /db2/filesystem2 de la base de datos conectada actualmente.

```
ALTER DATABASE DROP STORAGE ON '/db2/filesystem1', '/db2/filesystem2'
```

Una vez descartado satisfactoriamente el almacenamiento, se utiliza la sentencia ALTER TABLESPACE con la cláusula REBALANCE para cada espacio de tablas que utilizaba estas vías de acceso de almacenamiento para reequilibrar el espacio de tablas.

- *Ejemplo 7:* Eliminar una vía de acceso de almacenamiento con una expresión de partición de base de datos (/dataForPartition \$N) que se había añadido previamente.

```
ALTER DATABASE DROP STORAGE ON '/dataForPartition $N'
```

Una vez descartado satisfactoriamente el almacenamiento, se utiliza la sentencia ALTER TABLESPACE con la cláusula REBALANCE para cada espacio de tablas que utilizaba estas vías de acceso de almacenamiento para reequilibrar el espacio de tablas.

ALTER EVENT MONITOR

La sentencia ALTER EVENT MONITOR modifica la definición de un supervisor de sucesos que tiene un destino para los datos del supervisor de sucesos de TABLE.

Invocación

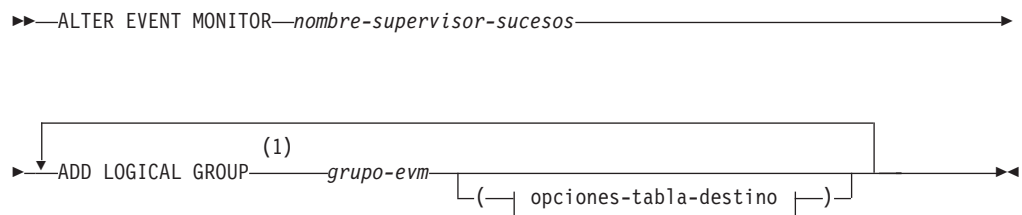
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

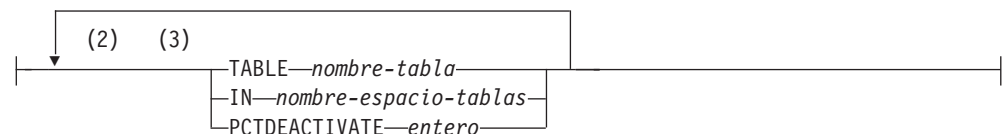
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM

Sintaxis



opciones-tabla-destino:



Notas:

- 1 Se puede añadir un grupo lógico a los supervisores de sucesos de TABLE (no supervisores de sucesos de UNFORMATTED EVENT TABLE).
- 2 Cada cláusula sólo puede especificarse una vez.
- 3 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que existe en el servidor actual y que tiene un destino para la tabla del supervisor de sucesos de TABLE.

ADD LOGICAL GROUP

Añade un grupo lógico al supervisor de sucesos que tiene un destino para los datos de TABLE.

ALTER EVENT MONITOR

grupo-vm

Identifica el grupo de datos lógicos para el que está añadiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tabla 10. Valores para grupo-vm basados en el tipo de supervisor de sucesos

Tipo de supervisor de sucesos	Valor grupo-vm
Base de datos	<ul style="list-style-type: none"> • DB • CONTROL¹ • DBMEMUSE
Tablas	<ul style="list-style-type: none"> • TABLE • CONTROL¹
Puntos muertos	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN • CONTROL¹
Puntos muertos con detalles	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • CONTROL¹
Puntos muertos con detalles e histórico	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • STMTHIST • CONTROL¹
Puntos muertos con detalles, histórico y valores	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • STMTHIST • STMTVALS • CONTROL¹
Espacios de tablas	<ul style="list-style-type: none"> • TABLESPACE • CONTROL¹
Agrupaciones de almacenamientos intermedios	<ul style="list-style-type: none"> • BUFFERPOOL • CONTROL¹
Conexiones	<ul style="list-style-type: none"> • CONNHEADER • CONN • CONTROL¹ • CONNMEMUSE

Tabla 10. Valores para grupo-evm basados en el tipo de supervisor de sucesos (continuación)

Tipo de supervisor de sucesos	Valor grupo-evm
Sentencias	<ul style="list-style-type: none"> • CONNHEADER • STMT • SUBSECTION⁴ • CONTROL¹
Transacciones	<ul style="list-style-type: none"> • CONNHEADER • XACT • CONTROL¹
Actividades	<ul style="list-style-type: none"> • ACTIVITY • ACTIVITYMETRICS • ACTIVITYSTMT • ACTIVITYVALS • CONTROL¹
Estadísticas	<ul style="list-style-type: none"> • QSTATS • SCSTATS • SCMETRICS • WCSTATS • WLSTATS • WLMETRICS • HISTOGRAMBIN • CONTROL¹
Violaciones de umbral	<ul style="list-style-type: none"> • THRESHOLDVIOLATIONS • CONTROL¹
Bloqueo ⁵	<ul style="list-style-type: none"> • LOCK • LOCK_PARTICIPANTS • LOCK_PARTICIPANT_ACTIVITIES • LOCK_ACTIVITY_VALUES • CONTROL¹
Antememoria de paquetes ⁵	<ul style="list-style-type: none"> • PKGCACHE • PKGCACHE_METRICS • CONTROL¹
Unidad de trabajo ⁵	<ul style="list-style-type: none"> • UOW • UOW_METRICS • UOW_PACKGE_LIST • UOW_EXECUTABLE_LIST • CONTROL¹

ALTER EVENT MONITOR

Tabla 10. Valores para grupo-evm basados en el tipo de supervisor de sucesos (continuación)

Tipo de supervisor de sucesos	Valor grupo-evm
Historial de cambios	<ul style="list-style-type: none"> • CHANGESUMMARY • EVMONSTART • TXNCOMPLETION • DDLSTMTEXEC • DBDBMCFG • REGVAR • UTILSTART • UTILSTOP • UTILPHASE • UTILLOCATION • CONTROL¹

¹ La cabecera de base de datos de los grupos de datos lógicos (sólo el elemento conn_time), el inicio y el desbordamiento se graban en el grupo CONTROL. El desbordamiento se graba si el supervisor de sucesos está en estado no bloqueado y se han descartado los sucesos.

² Corresponde al suceso DETAILED_DLCONN.

³ Corresponde a los grupos de datos lógicos LOCK que se producen dentro de cada suceso DETAILED_DLCONN.

⁴ Sólo se crea para los entornos de bases de datos particionadas.

⁵ Se refiere a la versión de la tabla de sucesos con formato de este tipo de supervisor de sucesos.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'
CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando la sentencia CREATE TABLE.

Al especificar un nombre de espacio de tablas para actividades, bloqueos, paquetes de antememoria o unidades del supervisor de sucesos de trabajo, el tamaño de la página del espacio de tablas afecta a las longitudes de INLINE LOB utilizadas. Por lo tanto, considere la posibilidad de especificar un espacio de tablas con el tamaño de página más grande posible para mejorar el rendimiento de INSERT del supervisor de sucesos.

PCTDEACTIVATE *entero*

Si está creándose una tabla en un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de

tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100. El valor por omisión es 100 (significa que el supervisor de sucesos se desactivará cuando el espacio de tablas se haya llenado por completo). Esta opción se ignora para espacios de tablas SMS. Cuando un espacio de tablas de destino tiene el redimensionamiento automático habilitado, se recomienda que PCTDEACTIVATE se establezca en 100.

Notas

- **Cuándo entran en vigor los cambios en el catálogo del sistema:** los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirman y se reactiva el supervisor de sucesos.

Ejemplo

Al supervisor de sucesos ACT le falta el grupo ACTIVITYMETRICS. Modifique el supervisor de sucesos para añadir este grupo y dar a la tabla el nombre "ACTMETRICS".

```
ALTER EVENT MONITOR ACT
  ADD LOGICAL GROUP ACTIVITYMETRICS TABLE ACTMETRICS
```

ALTER FUNCTION

La sentencia ALTER FUNCTION modifica las propiedades de una función existente.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema de la función
- Propietario de la función, como se ha registrado en la columna OWNER de la vista de catálogo SYSCAT.ROUTINES
- Autorización DBADM

Para modificar el EXTERNAL NAME de una función, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos
- Autorización DBADM

Para modificar una función con el fin establecerla como no delimitada, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

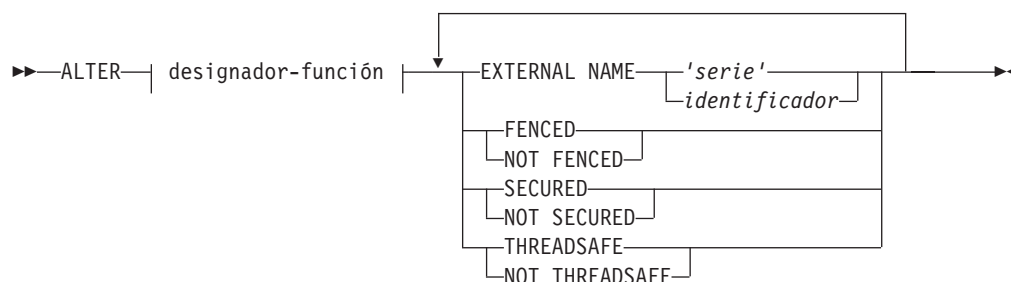
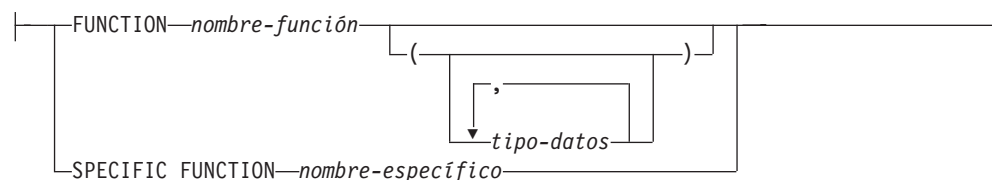
Para modificar una función con el fin de que esté delimitada, no se necesita ninguna autorización ni privilegio adicional.

Para modificar una función con el fin de establecerla como SECURED o NOT SECURED, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización SECADM
- Autorización CREATE_SECURE_OBJECT

Si no se especifica ninguna otra cláusula, no se necesita ningún otro privilegio para procesar la sentencia.

Sintaxis

**designador-función:****Descripción***designador-función*

Identifica de forma exclusiva la función que va a modificarse. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

EXTERNAL NAME 'serie' o identificador

Identifica el nombre del código escrito por el usuario que implementa la función. Esta opción sólo puede especificarse cuando se alteran funciones externas (SQLSTATE 42849).

FENCED o NOT FENCED

Especifica si se considera que la función es segura para ejecutarse en el espacio de dirección o en el proceso del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED). La mayoría de las funciones tienen la opción de ejecutarse como FENCED o como NOT FENCED.

Si una función se altera para que sea FENCED, el gestor de bases de datos aísla sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:

El uso de NOT FENCED en funciones que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 disponen de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no pueden garantizar la integridad completa cuando se utilizan funciones NOT FENCED definidas por el usuario.

Una función declarada como NOT THREADSAFE no puede modificarse para que sea NOT FENCED (SQLSTATE 42613).

Si una función tiene algún parámetro cuya definición sea AS LOCATOR y se ha definido con la opción NO SQL, la función no puede modificarse para que sea FENCED (SQLSTATE 42613).

ALTER FUNCTION

Esta opción no se puede modificar para las funciones LANGUAGE OLE, OLEDB ni CLR (SQLSTATE 42849).

SECURED o NOT SECURED

Especifica si la función se considera segura para el control de acceso a filas y columnas.

NOT SECURED

Indica que la función no se considera segura. Cuando se invoca la función, los argumentos de la función no deben hacer referencia a una columna para la que haya habilitada una máscara de columna y el control de acceso de nivel de columna esté activado para su tabla (SQLSTATE 428HA). Esta norma se aplica a las funciones definidas por el usuario no seguras que se invocan en algún punto de la sentencia.

SECURED

Indica que la función se considera segura.

La función debe ser segura cuando se hace referencia a ella en un permiso de fila o una máscara de columna (SQLSTATE 428H8).

La función debe ser segura cuando se hace referencia a ella en una tabla de consulta materializada y ésta hace referencia a una tabla que tiene el control de acceso de nivel de fila o columna activado (SQLSTATE 428H8).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que la función es segura para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si la función se ha definido con un LANGUAGE distinto de OLE y OLEDB:

- Si la función se ha definido como THREADSAFE, el gestor de bases de datos puede invocar la función en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, una función no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Las funciones FENCED y NOT FENCED pueden ser THREADSAFE.
- Si la función se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará al mismo tiempo la función en el mismo proceso que otra rutina. Sólo una función FENCED puede ser NOT THREADSAFE (SQLSTATE 42613).

Esta opción no puede modificarse para las funciones LANGUAGE OLE u OLEDB (SQLSTATE 42849).

Notas

- No es posible modificar una función que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).
- Las funciones declaradas como LANGUAGE SQL, las funciones derivadas o las funciones de plantilla no pueden modificarse (SQLSTATE 42917).
- *Alteración de una función de NOT SECURED a SECURED:* Por lo general los usuarios con autorización SECADM no tienen privilegios para alterar objetos de base de datos como funciones definidas por el usuario y activadores. Suelen examinar las acciones que lleva a cabo una función, garantizar que es segura y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya tenga los privilegios obligatorios para alterar la función definida por el usuario para convertirla en segura. Tras alterar la función, revocan la autorización CREATE_SECURE_OBJECT al usuario al que se le ha otorgado.

La función se considera segura. El atributo SECURED se considera una aseercción que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. El gestor de bases de datos presupone que ese procedimiento de auditoría de control está en vigor para todas las sentencias ALTER FUNCTION posteriores o cambios en los paquetes externos.

Los paquetes y las sentencias de SQL colocadas dinámicamente en antememoria que dependen de la función podrían quedar invalidadas porque el atributo seguro afecta a la selección de la vía de acceso de las sentencias que implican tablas para las que se ha activado el control de acceso de nivel de fila o columna y la función que se va a sustituir.

- **Alteración de una función de SECURED a NOT SECURED:** la función se considera no segura. Los paquetes y las sentencias de SQL colocadas dinámicamente en antememoria que dependen de la función podrían quedar invalidadas porque el atributo seguro afecta a la selección de la vía de acceso de las sentencias que implican tablas para las que se ha activado el control de acceso de nivel de fila o columna.
- **Invocación de otras funciones definidas por el usuario en una función segura:** cuando se hace referencia a una función segura definida por el usuario en una sentencia de manipulación de datos en la que se hace referencia a una tabla obligatoria de control de acceso a filas o columnas, si la función segura definida por el usuario invoca otras funciones definidas por el usuario, el gestor de bases de datos no valida si esas funciones anidadas definidas por el usuario son seguras. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.

Ejemplo

La función MAIL() se ha probado minuciosamente. Para mejorar su rendimiento, altere la función para que sea NOT FENCED.

```
ALTER FUNCTION MAIL() NOT FENCED
```

ALTER HISTOGRAM TEMPLATE

La sentencia ALTER HISTOGRAM TEMPLATE se utiliza para modificar la plantilla que describe el tipo de histograma que puede utilizarse para alterar temporalmente uno o más histogramas por omisión de una clase de servicio o de una clase de trabajo.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Sintaxis

```
▶—ALTER HISTOGRAM TEMPLATE—nombre-plantilla—————▶  
▶—HIGH BIN VALUE—constante-bigint—————▶▶
```

Descripción

nombre-plantilla

Da nombre a la plantilla del histograma. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre debe identificar una plantilla de histograma existente en el servidor actual (SQLSTATE 42704). El nombre de plantilla puede ser la plantilla de histograma del sistema por omisión SYSDEFAULTHISTOGRAM.

HIGH BIN VALUE *constante-bigint*

Especifica el valor superior desde el segundo binario hasta el último (el último binario tiene un valor superior no vinculado). Las unidades dependen del modo en que se utiliza el histograma. El valor máximo es 268 435 456.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)

- GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplo

Cambie el valor binario superior de una plantilla de histograma denominada LIFETIMETEMP.

```
ALTER HISTOGRAM TEMPLATE LIFETIMETEMP  
HIGH BIN VALUE 90000
```

ALTER INDEX

La sentencia ALTER INDEX modifica la definición de un índice.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- El privilegio ALTERIN sobre el esquema del índice
- El privilegio ALTER sobre la tabla en la que está definido el índice
- El privilegio CONTROL sobre el índice
- Autorización DBADM

Sintaxis

```
▶▶ ALTER INDEX nombre-índice COMPRESS  NO  YES ▶▶
```

Descripción

INDEX *nombre-índice*

Identifica el índice que se va a modificar. El nombre debe identificar un índice existente en el servidor actual (SQLSTATE 42704).

COMPRESS

Especifica si hay que habilitar o inhabilitar la compresión de índice. El índice no debe ser un índice de bloque MDC o ITC, un índice de catálogo, un índice de vía de acceso XML, una especificación de índice o un índice en una tabla temporal creada o declarada (SQLSTATE 42995).

NO Especifica que la compresión de índice está inhabilitada. Un índice comprimido permanecerá comprimido hasta que se vuelva a crear mediante la reorganización o recreación.

YES

Especifica que la compresión de índice está habilitada. Un índice sin comprimir permanecerá sin comprimir hasta que se vuelva a crear mediante la reorganización o recreación.

Ejemplo

Modificar el índice JOB_BY_DPT para que sea comprimido.

```
ALTER INDEX JOB_BY_DPT
COMPRESS YES
```

ALTER MASK

La sentencia ALTER MASK modifica una máscara de columna que existe en el servidor actual.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES se especifica implícita o explícitamente.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

```
▶▶ ALTER MASK nombre-máscara [ENABLE | DISABLE] ▶▶
```

Descripción

nombre-máscara

Identifica la máscara de columna que se va a modificar. El nombre debe identificar una máscara existente en el servidor actual (SQLSTATE 42704).

ENABLE

Habilita la máscara de columna. Si el control de acceso de nivel de columna no está activado actualmente en la tabla, el permiso de columna será efectivo cuando se active el control de acceso de nivel de columna en la tabla. Si el control de acceso de nivel de columna está activado actualmente para la tabla, la máscara de columna será efectiva inmediatamente y todos los paquetes y las sentencias colocadas dinámicamente en antememoria que hacen referencia a la tabla se invalidarán.

La sentencia ENABLE se ignorará si la máscara de columna ya está habilitada.

DISABLE

Inhabilita la máscara de columna. Si el control de acceso de nivel de columna no está activado actualmente en la tabla, el permiso de columna seguirá sin ser efectivo cuando se active el control de acceso de nivel de columna en la tabla. Si el control de acceso de nivel de columna está activado actualmente para la tabla, la máscara de columna dejará de ser efectiva inmediatamente y todos los paquetes y las sentencias colocados dinámicamente en antememoria que hacen referencia a la tabla se invalidarán.

La sentencia DISABLE se ignorará si la máscara de columna ya está inhabilitada.

Ejemplos

- *Ejemplo 1:* habilitar máscara de columna M1.
ALTER MASK M1 ENABLE
- *Ejemplo 2:* inhabilitar máscara de columna M1.
ALTER MASK M1 DISABLE

ALTER METHOD

La sentencia ALTER METHOD modifica un método existente cambiando el cuerpo de método que se asocia al método.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
 - Privilegio ALTERIN para el esquema del tipo
 - Propietario del tipo, como se ha registrado en la columna OWNER de la vista de catálogo SYSCAT.DATATYPES
- Autorización DBADM

Sintaxis

```

▶▶ALTER designador-método [EXTERNAL NAME 'serie' | identificador]

```

designador-método:

```

METHOD nombre-método ( [ tipo-datos ] ) FOR nombre-tipo
[ SPECIFIC METHOD nombre-específico ]

```

Descripción

designador-método

Identifica de forma exclusiva el método que va a modificarse. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

EXTERNAL NAME 'serie' o identificador

Identifica el nombre del código escrito por el usuario que implementa el método. Esta opción sólo puede especificarse cuando se alteran métodos externos (SQLSTATE 42849).

Notas

- No es posible modificar un método que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).
- Los métodos declarados como LANGUAGE SQL no pueden modificarse (SQLSTATE 42917).

- Los métodos declarados como LANGUAGE CLR no se pueden modificar (SQLSTATE 42849).
- El método especificado debe tener un cuerpo antes de que pueda modificarse (SQLSTATE 42704).

Ejemplo

Altere el método DISTANCE() del tipo estructurado ADDRESS_T para que utilice la biblioteca newaddresslib.

```
ALTER METHOD DISTANCE()  
FOR ADDRESS_T  
EXTERNAL NAME 'newaddresslib!distance2'
```

ALTER MODULE

La sentencia ALTER MODULE modifica la definición de un módulo.

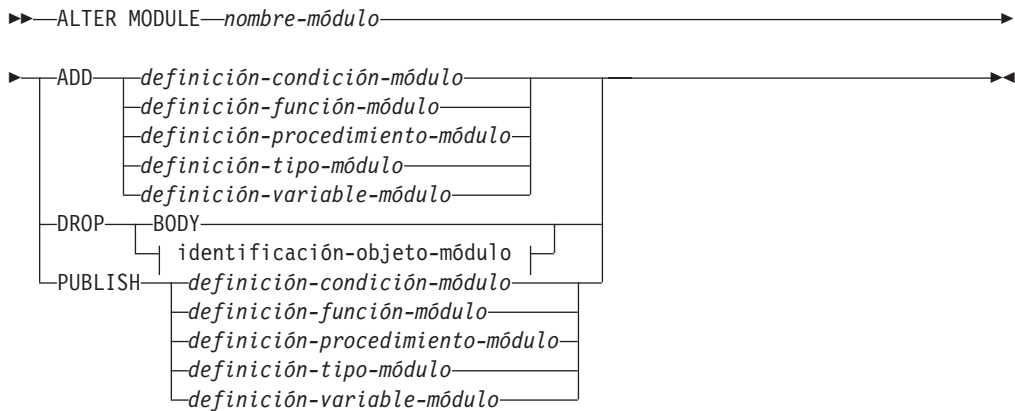
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

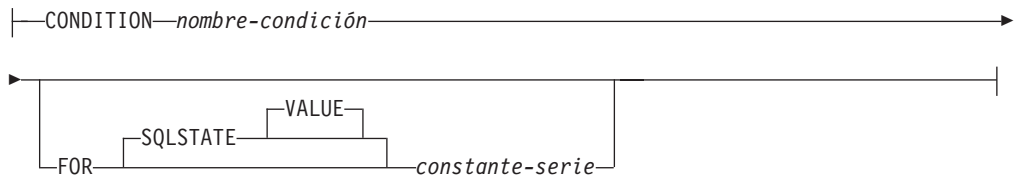
Autorización

Los privilegios que mantiene el ID de autorización de la sentencia deben incluir todos los privilegios necesarios para invocar las sentencias de SQL especificadas en la sentencia ALTER MODULE.

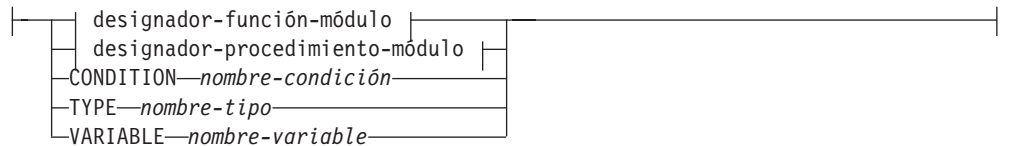
Sintaxis



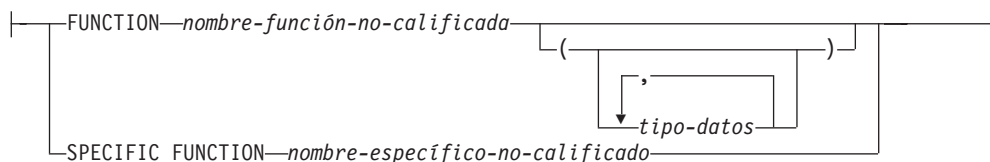
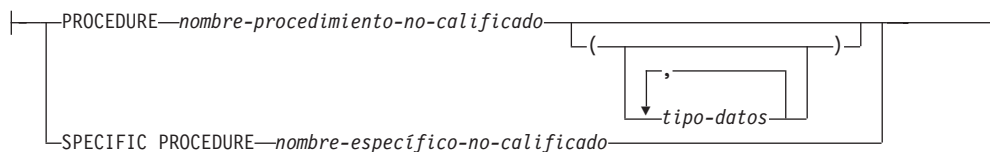
definición-condición-módulo:



identificación-objeto-módulo:



designador-función-módulo:

**designador-procedimiento-módulo:****Descripción***module-name*

Identifica el módulo que se va a modificar. El nombre-módulo debe identificar un módulo que exista en el servidor actual (SQLSTATE 42704).

ADD

Añade un objeto al módulo o añade el cuerpo a una definición de rutina que ya existe en el módulo sin un cuerpo. Si se añade un tipo definido por el usuario o una variable global, el objeto no debe identificar un tipo definido por el usuario o una variable global que ya exista en el módulo. Si el tipo definido por el usuario o la variable global no existía, se añade al módulo para utilizarse solamente en ese módulo.

Si se añade una rutina y la rutina especificada no existe, se añade la rutina. Si se añade una rutina y la rutina especificada existe, la definición de rutina existente no debe incluir un cuerpo de rutina (SQLSTATE 42723). Este prototipo de rutina se sustituye completamente por la nueva definición de rutina, incluidos los atributos de rutina y el cuerpo de rutina, excepto que se retiene el atributo publicado. Se considera que la rutina especificada existe si una de las condiciones siguientes es verdadera:

- Existe una rutina en el módulo con el mismo nombre específico y el mismo nombre de rutina.
- La rutina especificada es un procedimiento y existe un procedimiento en el módulo que tiene el mismo nombre y el mismo número de parámetros. No es necesario que los nombres y tipos de datos de los parámetros coincidan.
- La rutina especificada es una función y existe una función en el módulo que tiene el mismo nombre y el mismo número de parámetros con tipos de datos coincidentes. La longitud, la precisión y la escala de los tipos de datos de parámetro no se comparan y pueden ser distintos al determinar si la rutina especificada existe. No es necesario que los nombres de los parámetros coincidan.

definición-condición-módulo

Añade una condición de módulo.

nombre-condición

Nombre de la condición. El nombre no debe identificar una condición existente en el módulo. El nombre-condición debe especificarse sin ninguna calificación (SQLSTATE 42601). El nombre de la condición debe ser exclusivo dentro del módulo.

FOR SQLSTATE *constante-serie*

Especifica el SQLSTATE que está asociado a la condición. La constante de serie debe especificarse como cinco caracteres entre comillas simples, y la clase SQLSTATE (los dos primeros caracteres) no deben ser '00'. Esta es una cláusula opcional.

definición-función-módulo

La sintaxis para añadir una función es la misma que la de la sentencia CREATE FUNCTION excepto la palabra clave CREATE, y tanto el nombre-función como el nombre-específico deben especificarse sin ninguna calificación (SQLSTATE 42601). Si la función es exclusiva dentro del módulo, se añade una nueva función. Si la función coincide con una función existente que no incluye un cuerpo (cuerpo-rutina-SQL o cláusula EXTERNAL NAME), este prototipo de función se sustituye por la nueva definición excepto que se retiene el atributo publicado. Todas las funciones de SQL añadidas al módulo se procesan como si se utilizase una sentencia de SQL compuesto (compilado).

La definición de función de módulo solo puede especificar la cláusula RETURNS TABLE cuando el cuerpo de rutina de SQL es una sentencia de SQL compuesto (compilado) que especifica NOT ATOMIC. La definición de función de módulo no debe especificar la cláusula SOURCE, la cláusula TEMPLATE o la opción LANGUAGE OLEDEB (SQLSTATE 42613).

definición-procedimiento-módulo

La sintaxis para definir el procedimiento es la misma que la de la sentencia CREATE PROCEDURE excepto la palabra clave CREATE, y tanto el nombre-procedimiento como el nombre-específico deben especificarse sin ninguna calificación (SQLSTATE 42601). Si la signatura de procedimiento es exclusiva dentro del módulo, se añade un nuevo procedimiento. Si el procedimiento coincide con un procedimiento existente que no incluye un cuerpo (cuerpo-rutina-SQL o cláusula EXTERNAL NAME), este prototipo de procedimiento se sustituye por la nueva definición excepto que se retiene el atributo publicado. El nombre del procedimiento puede empezar por "SYS_" solamente para añadir el procedimiento de inicialización de módulo denominado SYS_INIT. Consulte las Notas para obtener información detallada.

definición-tipo-módulo

La sintaxis para definir el tipo definido por el usuario es la misma que la de la sentencia CREATE TYPE excepto la palabra clave CREATE, y el nombre-tipo debe especificarse sin ninguna calificación (SQLSTATE 42601). El nombre del tipo definido por el usuario debe ser exclusivo dentro del módulo. Un tipo estructurado no puede definirse en un módulo. Cualquier función generada necesaria para soportar la definición de tipo también está definida en el módulo. Si se publica el tipo de módulo definido por el usuario también se publican las funciones generadas.

definición-variable-módulo

La sintaxis para definir la variable es la misma que la de la sentencia CREATE VARIABLE excepto la palabra clave CREATE, y el nombre-variable debe especificarse sin ninguna calificación (SQLSTATE 42601). El nombre de la variable debe ser exclusivo dentro del módulo.

DROP

Descarta una parte especificada de un módulo. La sintaxis identificación-objeto-módulo se utiliza para identificar el objeto para descartar a menos que se esté descartando el cuerpo del módulo.

BODY

Descarta el cuerpo del módulo, que incluye:

- todos los objetos que no están publicados.
- el cuerpo de rutina de cualquier rutina de SQL publicada.
- la referencia EXTERNAL para todas las rutinas externas publicadas.

PUBLISH

Añade un objeto nuevo al módulo y hace que esté disponible para utilizarlo fuera del módulo. En el caso de rutinas, puede especificarse un prototipo de rutina que no incluya el cuerpo ejecutable de la rutina.

definición-condición-módulo

Añade una condición de módulo que está disponible para utilizarla fuera del módulo.

nombre-condición

Nombre de la condición. El nombre no debe identificar una condición existente en el módulo. El nombre-condición debe especificarse sin ninguna calificación (SQLSTATE 42601). El nombre de la condición debe ser exclusivo dentro del módulo.

FOR SQLSTATE *constante-serie*

Especifica el SQLSTATE que está asociado a la condición. La constante de serie debe especificarse como cinco caracteres entre comillas simples, y la clase SQLSTATE (los dos primeros caracteres) no deben ser '00'. Ésta es una cláusula opcional.

definición-función-módulo

La sintaxis para definir la función es la misma que la de la sentencia CREATE FUNCTION excepto la palabra clave CREATE, y tanto el nombre-función como el nombre-específico deben especificarse sin ninguna calificación (SQLSTATE 42601). La definición de la función debe incluir el nombre de función, la especificación completa de cualquier parámetro y la cláusula RETURNS. Los tipos de datos de módulo definidos por el usuario que no están publicados no son candidatos para los tipos de datos de parámetro o el tipo de datos de la cláusula RETURNS. Las variables de módulo que no están publicadas no son candidatas para el objeto de anclaje en una cláusula ANCHOR de un tipo de datos de parámetro o un tipo de datos RETURNS. Un prototipo de función puede especificarse omitiendo la cláusula LANGUAGE (o especificando LANGUAGE SQL) y cuerpo-rutina-SQL. La signatura de función debe ser exclusiva dentro del módulo. El nombre de la función no debe empezar por "SYS_" (SQLSTATE 42939). Todas las funciones de SQL añadidas al módulo se procesan como si se utilizase una sentencia de SQL compuesto (compilado).

La definición de función de módulo solo puede especificar la cláusula RETURNS TABLE cuando el cuerpo de rutina de SQL es una sentencia de SQL compuesto (compilado) que especifica NOT ATOMIC. La definición de función de módulo no debe especificar la cláusula SOURCE, la cláusula TEMPLATE o la opción LANGUAGE OLEDEB (SQLSTATE 42613).

definición-procedimiento-módulo

La sintaxis para definir el procedimiento es la misma que la de la sentencia CREATE PROCEDURE excepto la palabra clave CREATE, y tanto el nombre-procedimiento como el nombre-específico deben especificarse sin ninguna calificación (SQLSTATE 42601). La definición del procedimiento debe incluir el nombre del procedimiento y la especificación completa de cualquier parámetro. Los tipos de datos de módulo definidos por el usuario que no están publicados no son candidatos para los tipos de datos

de parámetro. Las variables de módulo que no están publicadas no son candidatas para el objeto de anclaje en una cláusula ANCHOR de una definición de parámetro. Un prototipo de función puede especificarse omitiendo la cláusula LANGUAGE (o especificando LANGUAGE SQL) y cuerpo-rutina-SQL. La signatura de procedimiento debe ser exclusiva dentro del módulo. El nombre del procedimiento no debe empezar por "SYS_" (SQLSTATE 42939).

definición-tipo-módulo

La sintaxis para definir el tipo definido por el usuario es la misma que la de la sentencia CREATE TYPE excepto la palabra clave CREATE, y el nombre-tipo debe especificarse sin ninguna calificación (SQLSTATE 42601). Los tipos de datos de módulo definidos por el usuario que no están publicados no son candidatos para ningún tipo de datos al que se haga referencia en la definición de tipo de datos de módulo definida por el usuario. Las variables de módulo que no están publicadas no son candidatas para el objeto de anclaje en una cláusula ANCHOR. El nombre del tipo definido por el usuario no debe empezar por "SYS_" (SQLSTATE 42939) y debe ser exclusivo dentro del módulo. Un tipo estructurado no puede definirse en un módulo. Cualquier función generada necesaria para soportar la definición de tipo también está definida en el módulo como función publicada.

definición-variable-módulo

La sintaxis para definir la variable es la misma que la de la sentencia CREATE VARIABLE excepto la palabra clave CREATE, y el nombre-variable debe especificarse sin ninguna calificación (SQLSTATE 42601). Los tipos de datos de módulo definidos por el usuario que no están publicados no son candidatos para ningún tipo de datos al que se haga referencia en la definición de variable. Las variables de módulo que no están publicadas no son candidatas para el objeto de anclaje en una cláusula ANCHOR. El nombre de la variable no debe empezar por "SYS_" (SQLSTATE 42939) y debe ser exclusivo dentro del módulo.

identificación-objeto-módulo

Identifica un único objeto de módulo.

designador-función-módulo

Identifica una única función de módulo de forma exclusiva.

FUNCTION *nombre-función-no-calificada*

Identifica una función en particular y sólo es válido si existe exactamente una instancia de función con el nombre nombre-función-no-calificada en el módulo. Para la función identificada puede definirse cualquier número de parámetros. Si no existe ninguna función con este nombre en el módulo, se generará un error (SQLSTATE 42704). Si existe más de una instancia de la función en el módulo, se generará un error (SQLSTATE 42725).

FUNCTION *nombre-función-no-calificada (tipo de datos,...)*

Proporciona la signatura de la función, que identifica la función de forma exclusiva. No se utiliza el algoritmo de resolución de función.

nombre-función-no-calificada

Especifica el nombre de la función.

(tipo-datos,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) cuando la función se ha

definido originalmente. Para identificar la instancia de función específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando 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.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos. FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE). Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el valor especificado cuando se definió la función.

No es necesario que un tipo de FLOAT(n) coincida con el valor que se ha 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 en el módulo no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-específico-no-calificado*

Identifica una función definida por el usuario en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión en el momento de la definición de la función. El nombre-específico-no-calificado debe identificar una instancia de función específica en el módulo; de lo contrario, se devuelve un error (SQLSTATE 42704).

designador-procedimiento-módulo

Identifica un único procedimiento de módulo de forma exclusiva.

PROCEDURE *nombre-procedimiento-no-calificado*

Identifica un procedimiento en particular y sólo es válido si existe exactamente una instancia de procedimiento con el nombre nombre-procedimiento-no-calificado en el módulo. Para el procedimiento identificado puede definirse cualquier número de parámetros. Si no existe ningún procedimiento con este nombre en el módulo, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia del procedimiento en el módulo, se devuelve un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento-no-calificado (tipo-datos,...)*

Proporciona la signatura del procedimiento, que identifica el procedimiento de forma exclusiva. No se utiliza el algoritmo de resolución de procedimiento.

nombre-procedimiento-no-calificado

Especifica el nombre del procedimiento.

(tipo-datos,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) cuando el procedimiento se ha definido originalmente. Para identificar la instancia de procedimiento específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

ALTER MODULE

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando 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.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE). Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el valor especificado cuando se definió el procedimiento.

No es necesario que un tipo de FLOAT(n) coincida con el valor que se ha 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 procedimiento con la signature especificada en el módulo, se devuelve un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico-no-calificado*

Identifica un procedimiento en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión en el momento de la definición del procedimiento. El nombre-específico-no-calificado debe identificar una instancia de procedimiento específica en el módulo; de lo contrario, se devuelve un error (SQLSTATE 42704).

TYPE *nombre-tipo*

Identifica un tipo definido por el usuario del módulo. El nombre-tipo debe especificarse sin ninguna calificación (SQLSTATE 42601) y debe identificar un tipo definido por el usuario que exista en el módulo (SQLSTATE 42704).

VARIABLE *nombre-variable*

Identifica una variable global del módulo. El nombre-variable debe especificarse sin ninguna calificación (SQLSTATE 42601) y debe identificar una variable global que exista en el módulo (SQLSTATE 42704).

CONDITION *nombre-condición*

Identifica una condición del módulo. El nombre-condición debe especificarse sin ninguna calificación y debe identificar una condición que exista en el módulo (SQLSTATE 42737).

Normas

- Los nombres de objetos en el módulo no pueden empezar por "SYS_", a excepción del nombre de procedimiento específicamente designado SYS_INIT (SQLSTATE 42939).
- **ALTER MODULE DROP FUNCTION:** Si se hace referencia a la función en la definición de un permiso de fila o una máscara de columna, no se puede descartar la función (SQLSTATE 42893).

- **ALTER MODULE DROP VARIABLE:** Si se hace referencia a la variable en la definición de un permiso de fila o una máscara de columna, no se puede descartar la variable (SQLSTATE 42893).
- **ALTER MODULE DROP BODY:** Si se hace referencia al módulo en la definición de un permiso de fila o una máscara de columna, no se puede descartar el módulo (SQLSTATE 42893).

Notas

- **Inicialización de módulo:** un módulo puede tener un procedimiento de inicialización especial denominado SYS_INIT que se ejecuta de forma implícita cuando se hace la primera referencia a una rutina de módulo o variable global de módulo. El procedimiento SYS_INIT debe implementarse sin parámetros, no puede devolver conjuntos de resultados y puede ser un procedimiento externo o de SQL que no puede publicarse (SQLSTATE 428HP). Si falla el procedimiento SYS_INIT, se devolverá un error para la sentencia que causó la inicialización de módulo (SQLSTATE 56098).
- **Uso de condiciones de módulo:** una condición de módulo solamente puede utilizarse con una sentencia SIGNAL, una sentencia RESIGNAL o una declaración de manejador que se encuentre dentro de una sentencia de SQL compuesto (compilado).
- **Invalidación:** si un prototipo de rutina se sustituye usando la acción ADD, se invalidarán todos los objetos que dependen de la rutina de módulo publicada. Si se emite DROP BODY, se invalidarán todos los objetos que dependen de rutinas de módulo publicadas.
- **Ofuscación:** las sentencias ALTER MODULE ADD FUNCTION, ALTER MODULE ADD PROCEDURE, ALTER MODULE PUBLISH FUNCTION y ALTER MODULE PUBLISH PROCEDURE se pueden enviar en formato ofuscado. En una sentencia ofuscada, solamente son legibles el nombre de la rutina y sus parámetros. El resto de la sentencia se codifica de tal manera que no es legible, pero que el servidor de bases de datos puede descodificar. Las sentencias ofuscadas se pueden generar mediante una llamada a la función DBMS_DDL.WRAP.

Ejemplo

Las sentencias siguientes crean un módulo denominado INVENTORY que contiene un tipo de matriz asociativa, una variable de ese tipo de datos, un procedimiento que añade elementos a la matriz y una función que extrae elementos de la matriz. Solamente se puede hacer referencia a la función y al procedimiento desde fuera del módulo en función de la palabra clave PUBLISH en las sentencias ALTER MODULE correspondientes. Sólo pueden hacer referencia al tipo de datos y a la variable otros objetos en el módulo.

```
CREATE MODULE INVENTORY

ALTER MODULE INVENTORY ADD
TYPE ITEMLIST AS INTEGER ARRAY[VARCHAR(100)]

ALTER MODULE INVENTORY ADD
VARIABLE ITEMS ITEMLIST

ALTER MODULE INVENTORY PUBLISH
PROCEDURE UPDATE_ITEM(NAME VARCHAR(100), QUANTITY INTEGER)
BEGIN
SET ITEMS[NAME] = QUANTITY;
END
```

ALTER MODULE

```
ALTER MODULE INVENTORY PUBLISH  
FUNCTION CHECK_ITEM(NAME VARCHAR(100)) RETURNS INTEGER  
RETURN ITEMS[NAME]
```

ALTER NICKNAME

La sentencia ALTER NICKNAME modifica la información de apodo asociada a un objeto de fuente de datos (por ejemplo, una tabla, una vista o un archivo).

Esta sentencia modifica la información que se almacena en la base de datos federada de las formas siguientes:

- La modificación de los nombres de columna locales para las columnas del objeto de fuente de datos
- La modificación de los tipos de datos locales para las columnas del objeto de fuente de datos
- La adición, el establecimiento o el descarte de opciones de apodo y columna
- La adición o el descarte de una clave primaria
- La adición o el descarte de una o varias restricciones de comprobación, de referencia o exclusivas
- La modificación de uno o varios atributos de restricción de comprobación o de referencia
- Modificar el almacenamiento en antememoria de datos en un servidor federado

Invocación

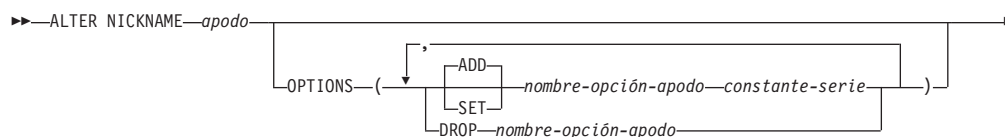
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

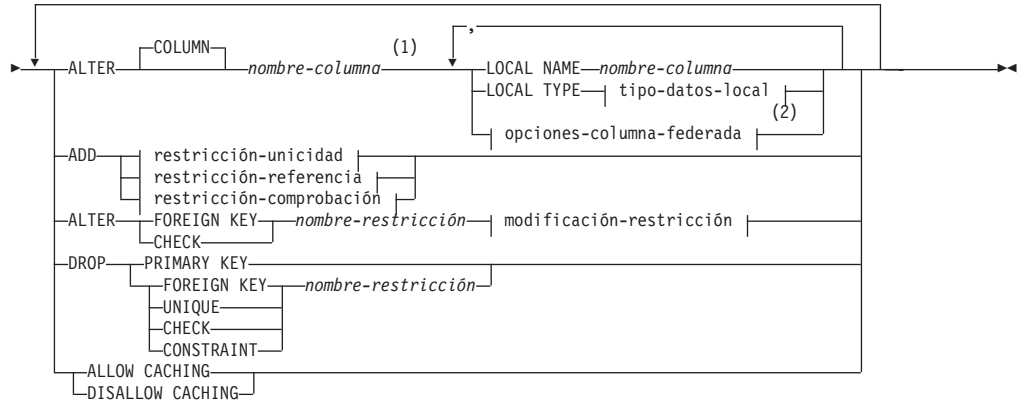
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- 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
- Propietario del apodo, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.TABLES
- Autorización DBADM

Sintaxis



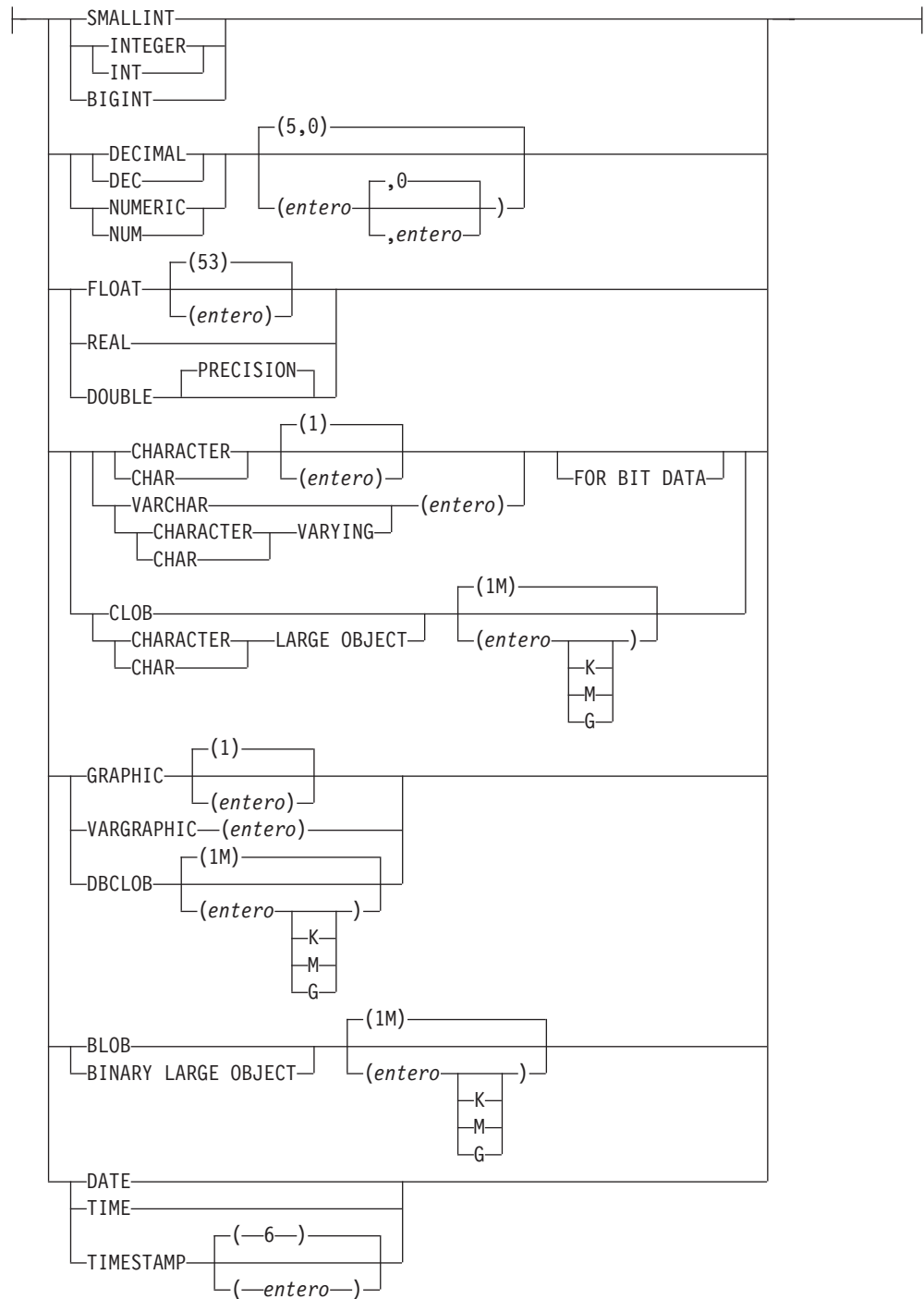
ALTER NICKNAME



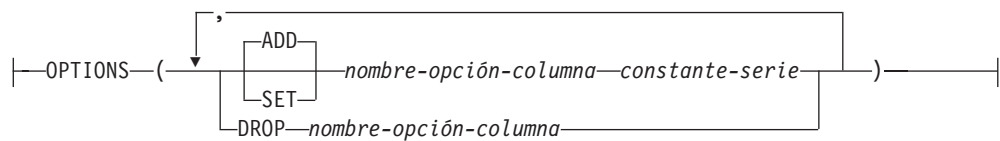
tipo-datos-locales:



tipo-incorporado:

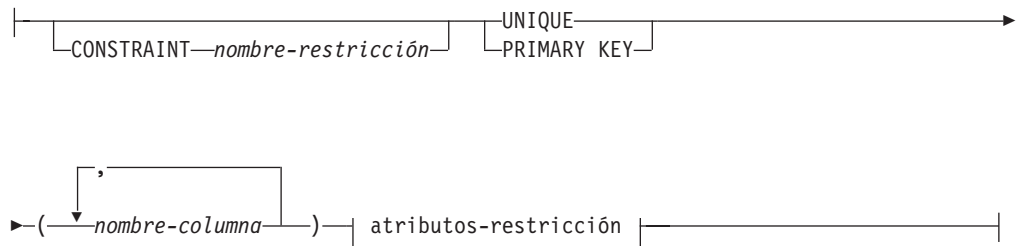


opciones-columna-federada:

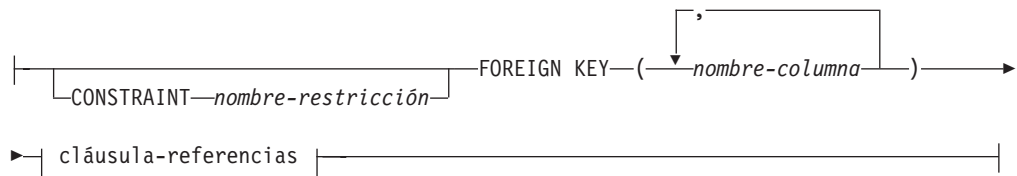


ALTER NICKNAME

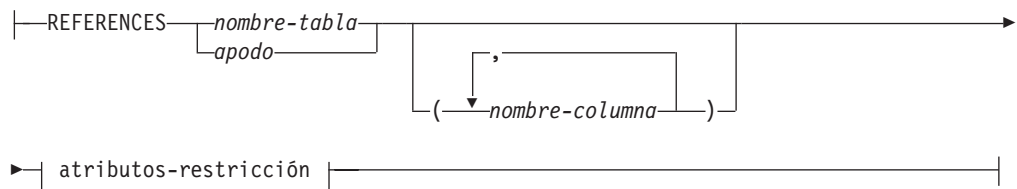
restricción-unicidad:



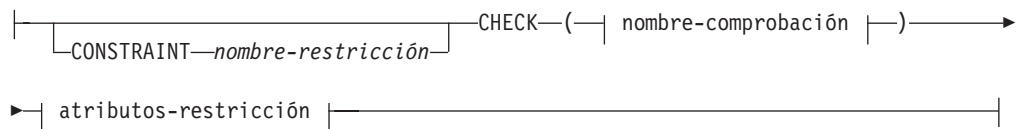
restricción-referencia:



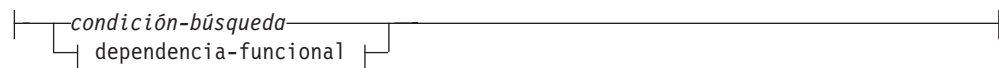
cláusula-referencias:



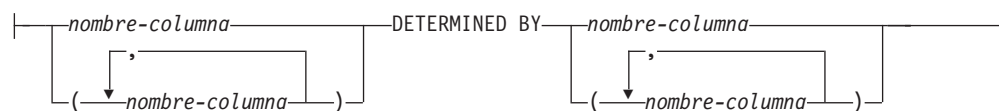
restricción-comprobación:



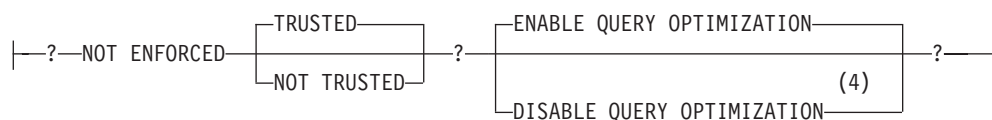
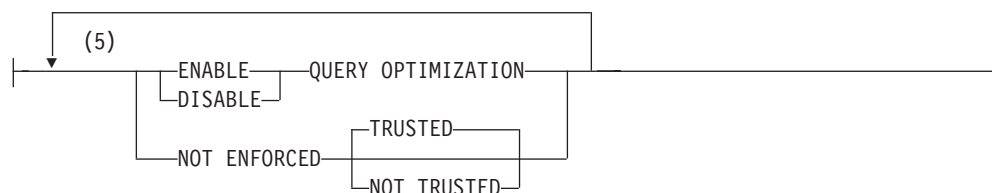
condición-error:



dependencia-funcional:



atributos-restricción:

**modificación-restricción:****Notas:**

- 1 No se puede especificar la cláusula ALTER COLUMN y una cláusula de restricción informativa ADD, ALTER o DROP en la misma sentencia ALTER NICKNAME.
- 2 Si debe 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.
- 3 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2).
- 4 No se da soporte a DISABLE QUERY OPTIMIZATION para una restricción de clave primaria o exclusiva.
- 5 Una misma cláusula no se debe especificar más de una vez.

Descripción*apodo*

Identifica el apodo para el objeto de fuente de datos (por ejemplo una tabla, una vista o un archivo) que contiene la columna que se está modificando. Debe ser un apodo descrito en el catálogo.

OPTIONS

Indica las opciones de apodo que se añaden, se establecen o se descartan cuando se modifica el apodo.

ADD

Añade una opción de apodo.

SET

Cambia el valor de una opción de apodo.

nombre-opción-apodo

Nombra una opción de apodo que se debe añadir o establecer.

constante-serie

Especifica el valor para *nombre-opción-apodo* como una constante de serie de caracteres.

DROP *nombre-opción-apodo*

Descarta una opción de apodo.

ALTER NICKNAME

ALTER COLUMN *nombre-columna*

Los nombres de la columna que se deben 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 del apodo (SQLSTATE 42703). No puede hacer referencia al mismo nombre de columna varias veces en la misma sentencia ALTER NICKNAME (SQLSTATE 42711).

LOCAL NAME *nombre-columna*

Especifica un nuevo nombre, *nombre-columna*, por el que el servidor federado debe hacer referencia a la columna que se va a modificar. El nuevo nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna del apodo (SQLSTATE 42711).

LOCAL TYPE *tipo-datos-local*

Especifica un nuevo tipo de datos locales con el que correlacionará el tipo de datos de la columna que se va a modificar. El nuevo tipo se indica mediante *tipo-datos-local*.

Algunos derivadores sólo dan soporte a un subconjunto de tipos de datos SQL. Para ver descripciones de tipos de datos específicos, consulte la descripción de la sentencia "CREATE TABLE".

OPTIONS

Indica las opciones de columna que se deben añadir, establecer o descartar para la columna especificada después de la palabra clave COLUMN.

ADD

Añade una opción de columna.

SET

Cambia el valor de una opción de columna.

nombre-opción-columna

Nombra una opción de columna que se debe añadir o establecer.

constante-serie

Especifica el valor para *nombre-opción-columna* como una constante de serie de caracteres.

DROP *nombre-opción-columna*

Descarta una opción de columna.

ADD *restricción-unicidad*

Define una restricción exclusiva. Consulte la descripción de la sentencia "CREATE NICKNAME".

ADD *restricción-referencial*

Define una restricción de referencia. Consulte la descripción de la sentencia "CREATE NICKNAME".

ADD *restricción-comprobación*

Define una restricción de comprobación. Consulte la descripción de la sentencia "CREATE NICKNAME".

ALTER FOREIGN KEY *nombre-restricción*

Altera los atributos de restricción de la restricción de referencia *nombre-restricción*. Para obtener una descripción de los atributos de restricción, consulte la sentencia "CREATE NICKNAME". El *nombre-restricción* debe identificar una restricción de referencia existente (SQLSTATE 42704).

ALTER CHECK *nombre-restricción*

Altera los atributos de restricción de la restricción de comprobación

nombre-restricción. El *nombre-restricción* debe identificar una restricción de comprobación existente (SQLSTATE 42704).

modificación-restricción

Ofrece opciones para cambiar los atributos asociados a restricciones de referencia o de comprobación.

ENABLE QUERY OPTIMIZATION

La restricción puede utilizarse para la optimización de la consulta cuando se dan las circunstancias adecuadas.

DISABLE QUERY OPTIMIZATION

No se puede utilizar la restricción para la optimización de consulta.

NOT ENFORCED

Especifica que el gestor de bases de datos no implanta la restricción durante operaciones normales como, por ejemplo, insertar, actualizar o suprimir.

TRUSTED

Se puede confiar en los datos para que se ajusten a la restricción.

TRUSTED sólo debe utilizarse si se reconoce de forma independiente que los datos de la tabla se ajustan a la restricción. Los resultados de la consulta pueden ser imprevisibles si los datos no se ajustan realmente a la restricción. Es la opción por omisión.

NOT TRUSTED

No se puede confiar en los datos para que se ajusten a la restricción.

NOT TRUSTED está pensado para casos en los que los datos se ajustan a la restricción para la mayor parte de las filas, pero no se reconoce de forma independiente que todas las filas o las futuras adiciones se ajustarán a la restricción. Si una restricción es NOT TRUSTED y se ha habilitado para la optimización de consultas, no se utilizará para realizar optimizaciones que dependan de los datos que se ajusten por completo a la restricción. Sólo se puede especificar NOT TRUSTED para las restricciones de integridad referenciales (SQLSTATE 42613).

DROP PRIMARY KEY

Descarta la definición de la clave primaria y todas las restricciones de referencia que dependen de esta clave primaria. El apodo debe tener una clave primaria.

DROP FOREIGN KEY *nombre-restricción*

Descarta la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia existente, definida en el apodo.

DROP UNIQUE *nombre-restricción*

Descarta la definición de la restricción exclusiva *nombre-restricción* y todas las restricciones de referencia que dependen de esta restricción exclusiva. El *nombre-restricción* debe identificar una restricción exclusiva existente.

DROP CHECK *nombre-restricción*

Descarta la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente, definida en el apodo.

DROP CONSTRAINT *nombre-restricción*

Descarta la restricción *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación, una restricción de referencia, una clave primaria o una restricción exclusiva definida en el apodo.

ALTER NICKNAME

ALLOW CACHING o DISALLOW CACHING

Especifica si el apodo se puede referenciar en una consulta que defina una tabla de consulta materializada, que podría utilizarse para guardar datos en la antememoria desde una fuente de datos en el servidor federado.

ALLOW CACHING

Especifica que el apodo se puede referenciar en una consulta que defina una tabla de consulta materializada, que permite guardar datos desde la fuente de datos en la antememoria en la tabla de consulta materializada en el servidor federado. Las opciones que pueden renovarse definidas para la tabla de consulta materializada especifican cómo se mantienen los datos guardados en la antememoria en la tabla de consulta materializada.

DISALLOW CACHING

Especifica que el apodo no se puede referenciar en una consulta que defina una tabla de consulta materializada. DISALLOW CACHING no se puede especificar para un apodo al que se hace referencia en la selección completa de una definición de tabla de consulta materializada (SQLSTATE 42917).

Normas

- Si se utiliza un apodo en una vista, método SQL o función SQL, o se definen sobre la misma restricciones informativas, la sentencia ALTER NICKNAME no se puede utilizar para cambiar los nombres locales o los tipos de datos para las columnas del apodo (SQLSTATE 42893). Sin embargo, la sentencia se puede utilizar para añadir, establecer o descartar opciones de columna, opciones de apodo o restricciones informativas.
- Si una definición de tabla de consulta materializada hace referencia a un apodo, la sentencia ALTER NICKNAME no se puede utilizar para cambiar los nombres locales, los tipos de datos, las opciones de columna ni las opciones de apodo (SQLSTATE 42893). Además, la sentencia no se puede utilizar para inhabilitar el almacenamiento en antememoria (SQLSTATE 42917). Sin embargo, la sentencia se puede utilizar para añadir, modificar o descartar restricciones informativas.
- 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 descarta una opción de columna, no afecta a ninguna otra opción de columna que se esté utilizando.
- Para apodos relacionales, la sentencia ALTER NICKNAME de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
 - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
 - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Para apodos no relacionales, la sentencia ALTER NICKNAME de una unidad de trabajo (UOW) determinada no puede procesarse bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
 - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
 - Una sentencia SELECT de la misma UOW ya hace referencia a un apodo al que se hace referencia en esta sentencia
 - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia

Notas

- Si se utiliza la sentencia ALTER NICKNAME para cambiar el nombre local de una columna de un apodo, las consultas deben hacer referencia a esa columna por su nuevo nombre.
- 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.
- *Almacenamiento en antememoria y objetos protegidos:* Para apodos cuyo objeto de la fuente de datos está protegido, especifique DISALLOW CACHING. Esto garantiza que cada vez que se utiliza el apodo, se devuelven los datos del ID de autorización apropiados desde la fuente de datos en el momento de la ejecución de la consulta. Esto se realiza limitando el uso del apodo en la definición de una tabla de consulta materializada en el servidor federado, que podría utilizarse para guardar en la antememoria los datos de apodo.

Ejemplos

- *Ejemplo 1:* El apodo NICK1 hace referencia a una tabla de DB2 para i denominada T1. También, COL1 es el nombre local que hace referencia a la primera columna de esta tabla, C1. Cambie el nombre local COL1 de C1 por NEWCOL.

```
ALTER NICKNAME NICK1
ALTER COLUMN COL1
LOCAL NAME NEWCOL
```

- *Ejemplo 2:* El apodo EMPLOYEE hace referencia a una tabla de DB2 para z/OS denominada EMP. Además, 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:* Indica que en una tabla Oracle, una columna con el tipo de datos 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')
```

- *Ejemplo 4:* Modifique la vía de acceso completamente calificada para el archivo de tabla estructurada, drugdata1.txt, para el apodo DRUGDATA1. Utilice la opción de apodo FILE_PATH y cambie el valor actual de la vía de acceso '/user/pat/drugdata1.txt' por '/usr/kelly/data/drugdata1.txt'.

```
ALTER NICKNAME DRUGDATA1
OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

ALTER PACKAGE

La sentencia ALTER PACKAGE modifica las opciones de vinculación de un paquete en el servidor actual sin necesidad de vincular o volver a vincular el paquete.

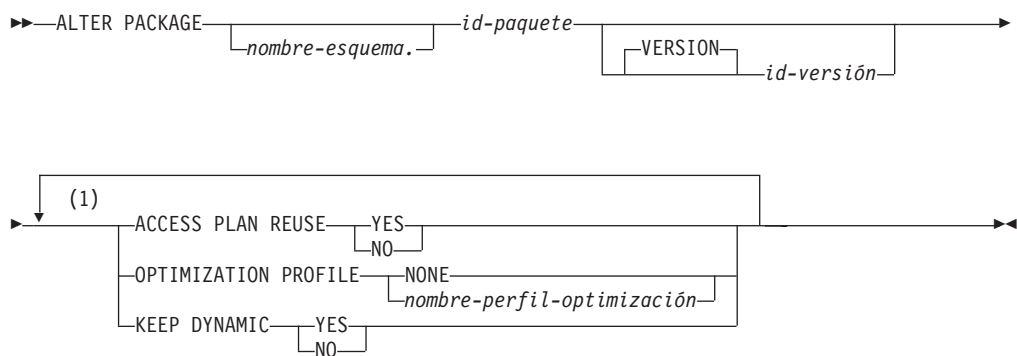
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN sobre el esquema
- Privilegio BIND sobre el paquete
- Autorización DBADM



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.

Descripción

nombre-esquema.id-paquete

Identifica el paquete que se debe modificar. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. El nombre de esquema y el ID de paquete, junto con el ID de versión especificado de forma implícita o explícita, deben identificar un paquete que exista en el servidor actual (SQLSTATE 42704).

VERSION *id-versión*

Identifica qué versión del paquete va a descartarse. Si no se especifica un valor, la versión tomará por omisión una serie de caracteres vacía. Si existen varios paquetes con el mismo nombre de paquete pero con distinta versión, sólo puede modificarse una versión del paquete en una invocación de la sentencia ALTER. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito

- Contiene minúsculas o mayúsculas y minúsculas

Si la sentencia se invoca desde un indicador de mandatos del sistema operativo, preceda cada delimitador de comillas dobles con una barra inclinada invertida para asegurarse de que el sistema operativo no divide los delimitadores.

ACCESS PLAN REUSE

Indica si el compilador de consultas debería intentar reutilizar los planes de acceso para las sentencias estáticas del paquete durante las revinculaciones implícitas y explícitas.

NO Especifica que no se reutilicen los planes de acceso.

YES

Especifica que se intenten reutilizar los planes de acceso.

OPTIMIZATION PROFILE

Indica qué perfil de optimización se asociará con el paquete, si debe asociarse alguno.

NONE

No asocia ningún perfil de optimización con el paquete. Si ya hay un perfil de optimización asociado con el paquete, se elimina la asociación.

nombre-perfil-optimización

Asocia el perfil de optimización *nombre-perfil-optimización* con el paquete. El perfil de optimización es un nombre en dos partes. Si el *nombre-perfil-optimización* especificado no está calificado, se utilizará el valor del registro especial CURRENT DEFAULT SCHEMA como calificador implícito. Si ya hay un perfil de optimización asociado con el paquete, la asociación se sustituye por el *nombre-perfil-optimización*.

Aunque la sentencia ALTER PACKAGE elimina la copia actual del paquete de la antememoria de paquetes de DB2, no invalida el paquete y no provoca que tenga lugar una revinculación implícita. Esto significa que, aunque al SQL dinámico le afectan los cambios que se efectúan en la sentencia, los planes de ejecución de consultas para las sentencias estáticas no se ven afectados por ello hasta la siguiente revinculación implícita o explícita.

KEEP DYNAMIC

A partir de DB2 para Linux, UNIX y Windows Versión 9.8 Fixpack 2, puede modificar el valor de la opción de vinculación KEEP DYNAMIC para un paquete sin que sea necesaria una nueva operación de vinculación, con lo que se evita una recompilación innecesaria hasta que tiene lugar la siguiente operación de vinculación. Esta opción controla durante cuánto tiempo el texto y la sección de la sentencia asociados a una sentencia preparada se conservan en el contexto de SQL, y se especifica si se conservan sentencias de SQL dinámico después de una operación COMMIT o ROLLBACK. Entra en vigor después de que todas las aplicaciones que están utilizando el paquete hayan efectuado las transacciones que estuvieran en ejecución cuando se ha ejecutado la sentencia **ALTER PACKAGE**.

YES

Indica al contexto SQL que conserve de forma indefinida el texto y la sección de la sentencia asociados a sentencias preparadas. Las sentencias de SQL dinámico se conservan de una transacción a otra. Todos los paquetes vinculados con KEEP DYNAMIC YES son, por omisión, compatibles con el comportamiento de antememoria de paquete existente.

NO

Indica al contexto SQL que elimine el texto y la sección de la sentencia asociados a sentencias preparadas al final de cada unidad de trabajo. Las versiones ejecutables de las sentencias preparadas y el texto de las sentencias en los paquetes vinculados con la opción KEEP DYNAMIC NO se eliminan del contexto SQL en los límites de transacciones. El cliente, el controlador o la aplicación tiene que preparar cualquier sentencia de SQL dinámico que desee reutilizar en una nueva unidad de trabajo.

En el caso de aplicaciones remotas que utilizan una API no incorporada de IBM, una vez que se haya asegurado de que las sentencias se prepararán en transacciones nuevas, puede utilizar esta opción para que WLB no se rechace exclusivamente en función del comportamiento de KEEP DYNAMIC. No obstante, incluso con esta opción, WLB puede rechazarse por otros motivos.

Las sentencias SELECT emitidas con cursores con la opción WITH HOLD quedan desasociadas del contexto de SQL en el siguiente límite de transacción en el que se cierre el cursor. Como resultado de ello, el equilibrio de carga de trabajo se permite siempre que no haya ninguna versión ejecutable de sentencias preparadas asociadas con la aplicación en el contexto SQL.

Nota: El equilibrio de carga de trabajo no se limita a las aplicaciones de SQL dinámico que utilizan API no incorporadas de IBM, como JDBC, .NET o CLI/ODBC, para ejecutar SQL dentro de los paquetes de cliente comunes. Estas interfaces vuelven a preparar de forma implícita sentencias SQL antes de ejecutarlas en transacciones donde su conexión pueden haberse movido a una versión nueva ejecutable de las sentencias preparadas.

Notas

- **Los valores de vista de catálogo podrían no reflejar la configuración que estaba vigente para el paquete:** dado que esta sentencia no activa una revinculación del paquete, es posible que la configuración de un paquete, tal y como se muestra en la vista de catálogo SYSCAT.PACKAGES, no refleje los valores que estaban vigentes realmente durante la última operación BIND o REBIND. Podría producirse la situación anterior si ALTER_TIME es mayor que LAST_BIND_TIME.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con los mandatos BIND y REBIND. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse APREUSE en lugar de ACCESS PLAN REUSE.
 - Puede especificarse OPTPROFILE en lugar de OPTIMIZATION PROFILE.
 - Puede especificarse KEEP DYNAMIC en lugar de KEEP DYNAMIC.

Ejemplos

Ejemplo 1: habilitar la reutilización de planes de acceso para el paquete TRUUVERT.EMPADMIN.

```
ALTER PACKAGE TRUUVERT.EMPADMIN ACCESS PLAN REUSE YES
```

Ejemplo 2: Se asume que se ha habilitado la reutilización de los planes de acceso para el paquete TRUUVERT.EMPADMIN. Se asume también que el perfil de optimización AYYANG.INDEXHINTS contiene un perfil de sentencia para una

sentencia específica del paquete. Asociar el perfil de optimización con este paquete para que omita la reutilización del plan de acceso para la sentencia.

```
ALTER PACKAGE TRUUVERT.EMPADMIN OPTIMIZATION PROFILE AYYANG.INDEXHINTS
```

Las sentencias dinámicas resultarán afectadas tras la confirmación de la sentencia; las sentencias estáticas resultarán afectadas con la próxima revinculación. Cuando el paquete se revincule, el compilador de consultas intentará reutilizar los planes de acceso para todas las sentencias estáticas en el paquete, salvo la sentencia identificada por el perfil de optimización. Al volver a compilar dicha sentencia, el compilador de consultas intentará por el contrario aplicar el perfil de la sentencia.

Ejemplo 3: La siguiente sentencia dará lugar a que no se asocie ningún perfil de optimización al paquete TRUUVERT.EMPADMIN.

```
ALTER PACKAGE TRUUVERT.EMPADMIN OPTIMIZATION PROFILE NONE
```

ALTER PERMISSION

La sentencia ALTER PERMISSION altera un permiso de fila en el servidor actual.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES se especifica implícita o explícitamente.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

```
▶▶ ALTER PERMISSION nombre-permiso { ENABLE | DISABLE } ▶▶
```

Descripción

nombre-permiso

Especifica el nombre del permiso de fila que se va a modificar. El nombre debe identificar un permiso de fila que ya existe en el servidor actual (SQLSTATE 42704). El nombre no debe identificar un permiso de fila por omisión que se ha creado de forma implícita mediante el gestor de bases de datos (SQLSTATE 428H9).

ENABLE

Habilita el permiso de fila. Si el control de acceso de nivel de fila no está activado actualmente en la tabla, el permiso de fila será efectivo cuando se active el control de acceso de nivel de fila en la tabla. Si el control de acceso de nivel de fila está activado actualmente para la tabla, el permiso de fila será efectivo inmediatamente y todos los paquetes y las sentencias en memoria caché dinámicas que hacen referencia a la tabla se invalidarán.

ENABLE se ignorará si el permiso de fila ya está definido como habilitado.

DISABLE

Inhabilita el permiso de fila. Si el control de acceso de nivel de fila no está activado actualmente en la tabla, el permiso de fila no será efectivo cuando se active el control de acceso de nivel de fila en la tabla. Si el control de acceso de nivel de fila está activado actualmente para la tabla, el permiso de fila dejará de ser efectivo inmediatamente y todos los paquetes y las sentencias en memoria caché dinámicas que hacen referencia a la tabla se invalidarán.

DISABLE se ignorará si el permiso de fila ya está definido como inhabilitado.

Ejemplos

- *Ejemplo 1:* habilitar permiso P1.
ALTER PERMISSION P1 ENABLE
- *Ejemplo 2:* inhabilitar permiso P1.
ALTER PERMISSION P1 DISABLE

ALTER PROCEDURE (externo)

La sentencia ALTER PROCEDURE (externo) modifica un procedimiento externo existente cambiando las propiedades del procedimiento.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema del procedimiento
- Propietario del procedimiento, como está registrado en la columna OWNER de la vista de catálogo SYSCAT.ROUTINES
- Autorización DBADM

Para modificar el EXTERNAL NAME de un procedimiento, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos
- Autorización DBADM

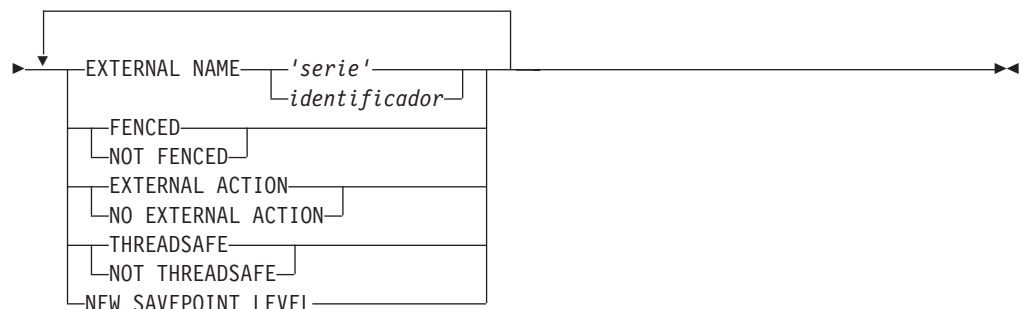
Para modificar un procedimiento con el fin de establecerlo como no delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

Para modificar un procedimiento con el fin de que esté delimitado, no se necesita ninguna autorización ni privilegio adicional.

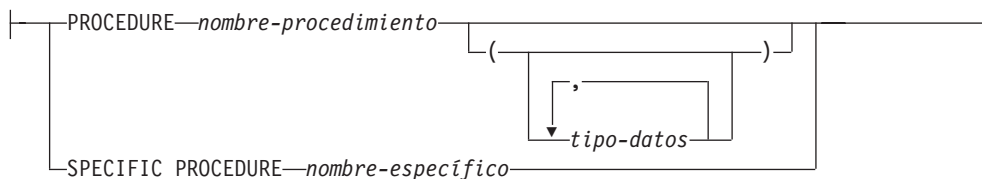
Sintaxis

►► ALTER | designador-procedimiento |



ALTER PROCEDURE (externo)

designador-procedimiento:



Descripción

designador-procedimiento

Identifica el procedimiento a modificar. El *designador-procedimiento* debe identificar un procedimiento que exista en el servidor actual. Se conservan el propietario del procedimiento y todos los privilegios en el procedimiento. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

EXTERNAL NAME 'serie' o identificador

Identifica el nombre del código escrito por el usuario que implementa el procedimiento.

FENCED o NOT FENCED

Especifica si se considera que el procedimiento es seguro para ejecutarse en el espacio de dirección o en el proceso del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED). La mayoría de los procedimientos tienen la opción de ejecutarse como FENCED o como NOT FENCED.

Si un procedimiento se altera para que sea FENCED, el gestor de bases de datos aísla sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que el procedimiento no pueda acceder a ellos. En general, un procedimiento que se ejecute como FENCED no funcionará tan bien como otro de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:

El uso de NOT FENCED en procedimientos que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 disponen de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no pueden garantizar la integridad completa cuando se utilizan procedimientos almacenados NOT FENCED.

Un procedimiento declarado como NOT THREADSAFE no puede modificarse para que sea NOT FENCED (SQLSTATE 42613).

Si un procedimiento tiene algún parámetro cuya definición sea AS LOCATOR y se ha definido con la opción NO SQL, el procedimiento no puede modificarse para que sea FENCED (SQLSTATE 42613).

Esta opción no se puede modificar para los procedimientos LANGUAGE OLE ni CLR (SQLSTATE 42849).

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si el procedimiento realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones que suponen que el procedimiento no tiene ningún impacto externo.

THREADSAFE o NOT THREADSAFE

Especifica si se considera que el procedimiento es seguro para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si se procedimiento se ha definido con un LANGUAGE distinto de OLE:

- Si el procedimiento se ha definido como THREADSAFE, el gestor de bases de datos puede invocar el procedimiento en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, un procedimiento no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Los procedimientos FENCED y NOT FENCED pueden ser THREADSAFE.
- Si el procedimiento se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará el procedimiento en el mismo proceso que otra rutina. Sólo un procedimiento FENCED puede ser NOT THREADSAFE (SQLSTATE 42613).

Esta opción no puede modificarse para los procedimientos LANGUAGE OLE (SQLSTATE 42849).

NEW SAVEPOINT LEVEL

Especifica que se debe crear un nuevo nivel de punto de salvaguarda para el procedimiento. Un nivel de punto de salvaguarda hace referencia al ámbito de referencia para cualquier sentencia relacionada con el punto de salvaguarda, así como al espacio de nombres utilizado para la comparación y la referencia de cualquier nombre de punto de salvaguarda.

El nivel de punto de salvaguarda para un procedimiento sólo se puede modificar por NEW SAVEPOINT LEVEL.

Normas

- No es posible modificar un procedimiento que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

Ejemplo

Altere el procedimiento PARTS_ON_HAND() para que sea NOT FENCED.

```
ALTER PROCEDURE PARTS_ON_HAND() NOT FENCED
```

ALTER PROCEDURE (derivado)

La sentencia ALTER PROCEDURE (derivado) modifica un procedimiento derivado existente cambiando el tipo de datos de uno o varios parámetros del procedimiento derivado.

Invocación

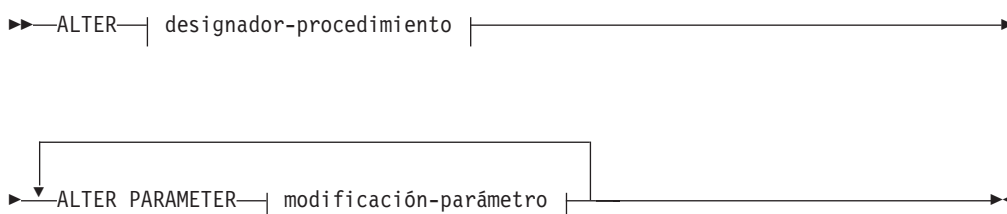
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

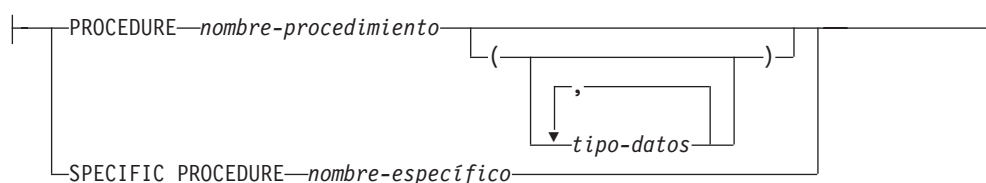
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema del procedimiento
- Propietario del procedimiento, como está registrado en la columna OWNER de la vista de catálogo SYSCAT.ROUTINES
- Autorización DBADM

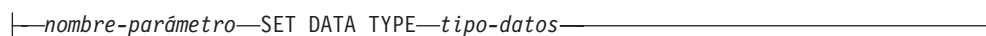
Sintaxis



designador-procedimiento:



modificación-parámetro:



Descripción

designador-procedimiento

Identifica de forma exclusiva el procedimiento que va a modificarse. El procedimiento identificado debe ser un procedimiento derivado (SQLSTATE 42849). Para obtener más información, consulte el apartado "Designadores de función, método y procedimiento" en la página 22.

nombre-parámetro

Identifica el parámetro que se debe modificar. El *nombre-parámetro* debe identificar un parámetro existente del procedimiento (SQLSTATE 42703). El nombre no debe identificar un parámetro que de algún otro modo se esté modificando en la misma sentencia ALTER PROCEDURE (SQLSTATE 42713).

tipo-datos

Especifica el nuevo tipo de datos local del parámetro. Se pueden especificar abreviaturas y especificaciones de tipo de datos SQL que sean válidas para la definición de *tipo-datos* de una sentencia CREATE TABLE. Los tipos BLOB, CLOB, DBCLOB, DECFLOAT, XML, REFERENCE y definidos por el usuario no están soportados (SQLSTATE 42815).

Ejemplo

Suponga que se ha creado el procedimiento federado FEDEMPLOYEE para un procedimiento Oracle remoto denominado 'EMPLOYEE'. El tipo de datos de un parámetro de entrada denominado SALARY se correlaciona con DOUBLE(8) en DB2. Modifique el tipo de datos de este parámetro a DECIMAL(5,2).

```
ALTER PROCEDURE FEDEMPLOYEE
ALTER PARAMETER SALARY
SET DATA TYPE DECIMAL(5,2)
```

ALTER PROCEDURE (SQL)

La sentencia ALTER PROCEDURE (SQL) modifica un procedimiento SQL existente cambiando las propiedades del procedimiento.

Invocación

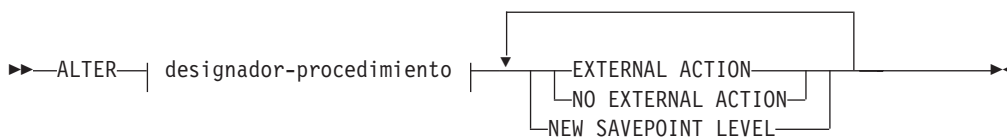
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

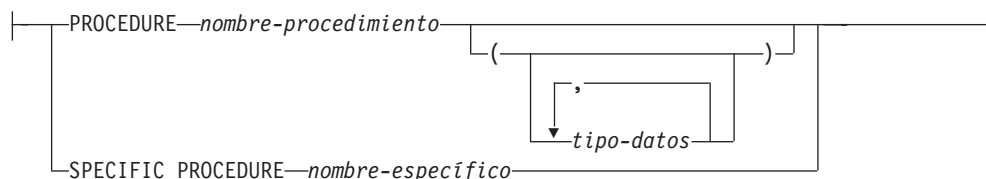
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema del procedimiento
- Propietario del procedimiento, como está registrado en la columna OWNER de la vista de catálogo SYSCAT.ROUTINES
- Autorización DBADM

Sintaxis



designador-procedimiento:



Descripción

designador-procedimiento

Identifica el procedimiento a modificar. El *designador-procedimiento* debe identificar un procedimiento que exista en el servidor actual. Se conservan el propietario del procedimiento y todos los privilegios en el procedimiento. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si el procedimiento realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones que suponen que el procedimiento no tiene ningún impacto externo.

NEW SAVEPOINT LEVEL

Especifica que se debe crear un nuevo nivel de punto de salvaguarda para el procedimiento. Un nivel de punto de salvaguarda hace referencia al ámbito de referencia para cualquier sentencia relacionada con el punto de salvaguarda, así como al espacio de nombres utilizado para la comparación y la referencia de cualquier nombre de punto de salvaguarda.

El nivel de punto de salvaguarda para un procedimiento sólo se puede modificar por NEW SAVEPOINT LEVEL.

Normas

- No es posible modificar un procedimiento que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

Ejemplo

Modifique el procedimiento MEDIAN_RESULT_SET para indicar que no tiene ninguna acción externa.

```
ALTER PROCEDURE MEDIAN_RESULT_SET(DOUBLE)
NO EXTERNAL ACTION
```

ALTER SCHEMA

La sentencia ALTER SCHEMA modifica el atributo de captura de datos de un esquema existente.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Propietario del esquema, como se ha registrado en la columna OWNER de la vista de catálogo SYSCAT.SCHEMATA
- Autorización DBADM

Sintaxis

```
▶▶ ALTER SCHEMA nombre-esquema DATA CAPTURE { NONE | CHANGES } ▶▶
```

Descripción

nombre-esquema

Identifica el esquema que se debe alterar. El *nombre-esquema* debe identificar un esquema que exista en el servidor actual (SQLSTATE 42704).

DATA CAPTURE

Indica si debe grabarse información adicional para la duplicación de datos en las anotaciones cronológicas.

NONE

Indica que no se va a anotar cronológicamente ninguna información adicional de duplicación de datos.

CHANGES

Indica que en el archivo de anotaciones cronológicas se registrará información adicional referente a los cambios de SQL efectuados en este esquema. Esta opción es necesaria para duplicar este esquema y cuando se utiliza el programa de captura de duplicación para capturar los cambios contenidos en el archivo de anotaciones cronológicas para este esquema.

Notas

- Modificar el atributo DATA CAPTURE en el nivel de esquema hace que las tablas recién creadas hereden el atributo DATA CAPTURE del esquema si no se especifica uno en el nivel de tabla. Modificar el atributo DATA CAPTURE en el nivel de esquema no afecta al atributo DATA CAPTURE de las tablas existentes dentro de ese esquema. Si el atributo DATA CAPTURE cambia y las tablas existentes no coinciden con el nuevo atributo, se devuelve un aviso (SQLSTATE 01696).
- Para buscar la lista de tablas que tienen el atributo DATA CAPTURE establecido en CHANGES, emita la consulta siguiente:


```
SELECT TABNAME, TABSCHEMA FROM SYSCAT.TABLES  
WHERE TYPE IN ('T','S','L')  
AND DATACAPTURE <> 'N'
```

- Para buscar la lista de tablas que tienen el atributo DATA CAPTURE establecido en NONE, emita la consulta siguiente:

```
SELECT TABNAME, TABSCHEMA FROM SYSCAT.TABLES  
WHERE TYPE IN ('T','S','L')  
AND DATACAPTURE = 'N'
```

ALTER SECURITY LABEL COMPONENT

La sentencia ALTER SECURITY LABEL COMPONENT modifica un componente de etiqueta de seguridad.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

```
▶▶ALTER SECURITY LABEL COMPONENT—nombre-componente—| cláusula-añadir-elemento |▶▶
```

cláusula-añadir-elemento:

```
|—ADD ELEMENT—constante-serie—|  
|  
|—| cláusula-elemento-matriz |  
|—| cláusula-elemento-árbol |—|
```

cláusula-elemento-matriz:

```
|—| BEFORE |  
|—| AFTER |—| constante-serie —|
```

cláusula-elemento-árbol:

```
|—| ROOT |  
|—| UNDER—| constante-serie —|  
|  
|—| OVER—| constante-serie —|
```

Descripción

nombre-componente

Especifica el nombre del componente de etiqueta de seguridad que se va a modificar. El componente con nombre debe existir en el servidor actual (SQLSTATE 42704).

ADD ELEMENT

Especifica el elemento que se va a añadir al componente de etiqueta de seguridad. Si no se especifican la *cláusula-elemento-matriz* y la *cláusula-elemento-árbol*, el elemento se añade a un componente de conjunto.

constante-serie

Valor de constante de serie que se va a añadir al conjunto de valores válidos del componente de etiqueta de seguridad. El valor no puede ser el

mismo que el de otro valor del conjunto de valores válidos del componente de etiqueta de seguridad (SQLSTATE 42713).

BEFORE o AFTER

Para un componente de matriz, especifica dónde se va a añadir el elemento en el conjunto ordenado de valores de elementos para el componente de etiqueta de seguridad.

BEFORE

El elemento que se va a añadir se coloca en orden inmediatamente antes del elemento existente identificado.

AFTER

El elemento que se va a añadir se coloca en orden inmediatamente después del elemento existente identificado.

constante-serie

Especifica un valor de constante de serie de un elemento existente en el componente de matriz (SQLSTATE 42704).

ROOT o UNDER

Para un componente de árbol, especifica dónde se va a añadir el elemento en la estructura de árbol de valores de elementos del nodo para el componente de etiqueta de seguridad.

ROOT

El elemento que se va a añadir se considera el nodo raíz del árbol.

UNDER *constante-serie*

El elemento que se va a añadir es un hijo inmediato del elemento identificado por la *constante-serie*. El valor *constante-serie* debe ser un elemento existente en el componente de árbol (SQLSTATE 42704).

OVER *constante-serie,...*

El elemento que se va a añadir es un hijo inmediato de cada elemento identificado por la lista de valores *constante-serie*. Cada valor *constante-serie* debe ser un elemento existente en el componente de árbol (SQLSTATE 42704).

Normas

- Los nombres de elementos no pueden contener ninguno de estos caracteres (SQLSTATE 42601):
 - Paréntesis de apertura - (
 - Paréntesis de cierre -)
 - Coma - ,
 - Dos puntos - :
- Un nombre de elemento no puede tener más de 32 bytes (SQLSTATE 42622).
- Si un componente de etiqueta de seguridad es un conjunto o un árbol, no podrán formar parte de este componente más de 64 elementos.
- Si el componente es una matriz, puede que sea posible o no llegar a una matriz cuyo número total de elementos coincida con el número total de elementos que se podían especificar al crear un componente de etiqueta de seguridad de tipo de matriz (65.535). DB2 asigna un valor codificado al nuevo elemento desde dentro del intervalo en el que se añade el nuevo elemento. Según el patrón que se siga al añadir elementos a un componente de matriz, el número de valores posibles que se pueden asignar desde un intervalo determinado puede agotarse rápidamente si se insertan varios elementos en dicho intervalo.

ALTER SECURITY LABEL COMPONENT

- BEFORE y AFTER sólo deben especificarse para un componente de etiqueta de seguridad que sea una matriz (SQLSTATE 42613).
- ROOT y UNDER sólo deben especificarse para un componente de etiqueta de seguridad que sea un árbol (SQLSTATE 42613).

Notas

- Para un componente de conjunto, no existe ningún orden en los elementos del conjunto.

Ejemplos

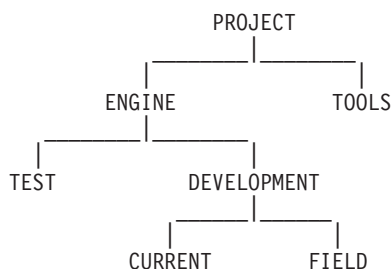
- *Ejemplo 1:* añade el elemento 'High classified' al componente de matriz de etiqueta de seguridad LEVEL entre los elementos 'Secret' y 'Classified'.

```
ALTER SECURITY LABEL COMPONENT LEVEL
ADD ELEMENT 'High classified' BEFORE 'Classified'
```

- *Ejemplo 2:* añade el elemento 'Funding' al componente de conjunto de etiqueta de seguridad COMPARTMENTS.

```
ALTER SECURITY LABEL COMPONENT COMPARTMENTS
ADD ELEMENT 'Funding'
```

- *Ejemplo 3:* añade los elementos 'ENGINE' y 'TOOLS' al componente de matriz de etiqueta de seguridad GROUPS. El diagrama siguiente muestra dónde se van a colocar estos nuevos elementos.



```
ALTER SECURITY LABEL COMPONENT GROUPS
ADD ELEMENT 'TOOLS' UNDER 'PROJECT'
```

```
ALTER SECURITY LABEL COMPONENT GROUPS
ADD ELEMENT 'ENGINE' UNDER 'PROJECT'
OVER 'TEST', 'DEVELOPMENT'
```

ALTER SECURITY POLICY

La sentencia ALTER SECURITY POLICY modifica una política de seguridad.

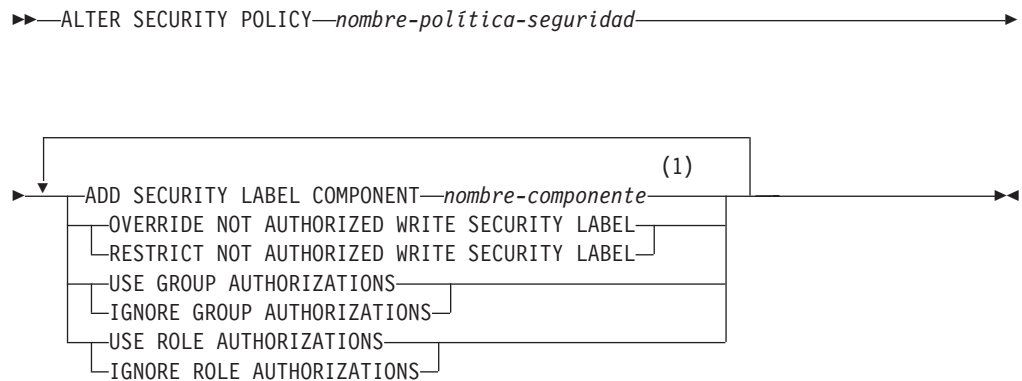
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Notas:

- 1 Sólo la cláusula ADD SECURITY LABEL COMPONENT puede especificarse más de una vez.

Descripción

nombre-política-seguridad

Especifica el nombre de la política de seguridad que se va a modificar. El nombre debe identificar una política de seguridad existente en el servidor actual (SQLSTATE 42710).

ADD SECURITY LABEL COMPONENT *nombre-componente*

Añade un componente de etiqueta de seguridad a la política de seguridad. No se debe especificar el mismo componente de seguridad más de una vez para la política de seguridad (SQLSTATE 42713). Una tabla no puede estar utilizando actualmente la política de seguridad (SQLSTATE 42893).

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL o **RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL**

Especifica la acción que se lleva a cabo cuando un usuario no está autorizado para grabar la etiqueta de seguridad explícitamente especificada que se proporciona en la sentencia INSERT o UPDATE emitida sobre una tabla que está protegida con esta política de seguridad. Una etiqueta de seguridad del usuario y las credenciales de exención determinan la autorización del usuario para grabar una etiqueta de seguridad proporcionada explícitamente.

ALTER SECURITY POLICY

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

Indica que se utilizará el valor de la etiqueta de seguridad del usuario para el acceso de grabación durante una operación de inserción o actualización, en lugar de la etiqueta de seguridad explícitamente especificada.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

Indica que la operación de inserción o actualización no será satisfactoria si el usuario no está autorizado a grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia INSERT o UPDATE (SQLSTATE 42519).

USE GROUP AUTHORIZATION o IGNORE GROUP AUTHORIZATION

Especifica si se van a tener en cuenta o no las etiquetas de seguridad y las exenciones otorgadas a grupos, directa o indirectamente, para cualquier intento de acceso.

USE GROUP AUTHORIZATION

Indica que se tendrá en cuenta cualquier etiqueta de seguridad o exención otorgada a grupos, directa o indirectamente.

IGNORE GROUP AUTHORIZATION

Indica que no se tienen en cuenta las etiquetas de seguridad o exenciones otorgadas a grupos.

USE ROLE AUTHORIZATION o IGNORE ROLE AUTHORIZATION

Especifica si se van a tener en cuenta o no las etiquetas de seguridad y exenciones otorgadas a roles, directa o indirectamente.

USE ROLE AUTHORIZATION

Indica que se tendrá en cuenta cualquier etiqueta de seguridad o exención otorgada a roles, directa o indirectamente.

IGNORE ROLE AUTHORIZATION

Indica que no se tienen en cuenta las etiquetas de seguridad o exenciones otorgadas a roles.

Normas

- Si un usuario no dispone directamente de una etiqueta de seguridad para el acceso de grabación, se devuelve un error en las situaciones siguientes (SQLSTATE 42519):
 - Un valor de la columna de etiqueta de seguridad de fila no se proporciona explícitamente como parte de la sentencia de SQL
 - La opción **OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL** está en vigor para la política de seguridad y no se permite que el usuario grave un objeto de datos con la etiqueta de seguridad proporcionada

Notas

- Los nuevos componentes se añaden lógicamente al final de la definición de la etiqueta de seguridad existente contenida en la política modificada. Las etiquetas de seguridad existentes definidas para esta política de seguridad se modifican para que contengan el nuevo componente como parte de su definición sin ningún elemento en su valor para dicho componente.
- *Invalidación de antememoria al cambiar NOT AUTHORIZED WRITE SECURITY LABEL*: el cambio de NOT AUTHORIZED WRITE SECURITY LABEL a un nuevo valor provoca la invalidación de cualquier sentencia de SQL dinámico o estático de la antememoria que dependa de cualquier tabla que esté protegida por la política de seguridad que se está modificando.

- Dado que el ID de autorización de la sesión es el ID de autorización central para el control de acceso basado en etiquetas, se pueden tener en cuenta las etiquetas de seguridad otorgadas a grupos o roles accesibles a través para todos los tipos de sentencias de SQL, incluido el SQL estático.
- Si hay más de una etiqueta de seguridad o exención disponible para un usuario con grupos o roles asociados en el momento del intento de acceso de grabación o de lectura, se evaluarán dichas etiquetas de seguridad y exenciones para ver si pueden elegirse en función de las normas siguientes:
 - Si la política de seguridad sólo permite que se tengan en cuenta las autorizaciones de rol, se tendrán en cuenta todas las etiquetas de seguridad y exenciones otorgadas a roles de las que el ID de autorización de usuario sea un miembro directo o indirecto. No se tendrán en cuenta las etiquetas de seguridad y exenciones otorgadas a roles a los que sólo se puede acceder como miembro a través de grupos asociados con el ID de autorización de usuario.
 - Si la política de seguridad sólo permite que se tengan en cuenta autorizaciones de grupo, se tendrán en cuenta todas las etiquetas de seguridad y exenciones otorgadas a grupos asociados con el ID de autorización de usuario. No se tendrán en cuenta las etiquetas de seguridad y exenciones otorgadas a roles a los que sólo se puede acceder como miembro a través de grupos asociados con el ID de autorización de usuario.
 - Si la política de seguridad permite que se tengan en cuenta tanto las autorizaciones de grupo como las de rol, se tendrá en cuenta cualquier etiqueta de seguridad y exención otorgada a los roles accesibles al usuario indirectamente a través de grupos asociados con el ID de autorización de usuario.
 - En ningún momento se tendrán en cuenta las autorizaciones de rol que sean accesibles al usuario sólo a través de PUBLIC.
- Si hay más de una etiqueta de seguridad que se pueda tener en cuenta durante un intento de acceso, los valores proporcionados para cada etiqueta de seguridad se fusionan en el nivel de componente individual para formar una etiqueta de seguridad que refleje la combinación de todos los valores disponibles en cada parte del componente de la política de seguridad. Ésta es el valor de etiqueta de seguridad que se utilizará para el intento de acceso.

Los mecanismos para combinar etiquetas de seguridad varían en función del tipo de componente. Los componentes de la etiqueta de seguridad resultante son los siguientes:

- Los componentes de conjunto contienen la unión de todos los valores exclusivos encontrados en las etiquetas de seguridad que pueden elegirse
- Los componentes de matriz contienen el elemento de orden más elevado encontrado en las etiquetas de seguridad que pueden elegirse
- Los componentes de árbol contienen la unión de todos los valores exclusivos encontrados en las etiquetas de seguridad que pueden elegirse
- Si hay más de una exención que se pueda tener en cuenta durante un intento de acceso, todas las exenciones encontradas se aplican en el intento de acceso.

Ejemplos

- *Ejemplo 1:* modifique una política de seguridad de nombre DATA_ACCESS para añadir un nuevo componente llamado REGION.

```
ALTER SECURITY POLICY DATA_ACCESS
ADD COMPONENT REGION
```

- *Ejemplo 2:* modifique una política de seguridad de nombre DATA_ACCESS para permitir el acceso a través de etiquetas de seguridad otorgadas a roles.

ALTER SECURITY POLICY

ALTER SECURITY POLICY DATA_ACCESS USE ROLE AUTHORIZATIONS

- *Ejemplo 3:* muestra las etiquetas de seguridad que se podrían tener en cuenta según los valores de las autorizaciones de grupo o rol en una política de seguridad. La política de seguridad SECUR_POL tiene un componente de matriz y un componente de conjunto que consta de los elementos siguientes:

Matriz = {TS, S, C, U}

Conjunto = {A, B, X, Y}

Las siguientes etiquetas de seguridad están definidas para SECUR_POL:

Etiqueta de seguridad L1 = C:A

Etiqueta de seguridad L2 = S:B

Etiqueta de seguridad L3 = TS:X

Etiqueta de seguridad L4 = U:Y

El usuario Paul es miembro del rol R1 y del grupo G1. El grupo G1 es miembro del rol R2. Se otorga la etiqueta de seguridad L1 a Paul. Se otorga la etiqueta de seguridad L2 al rol R1. Se otorga la etiqueta de seguridad L3 al grupo G1. Se otorga la etiqueta de seguridad L4 al rol R2. La tabla siguiente muestra las etiquetas de seguridad que se tendrían en cuenta para cualquier intento de acceso de Paul, en función de los distintos valores posibles de la política de seguridad SECUR_POL.

Tabla 11. Etiquetas de seguridad consideradas como función de los valores de la política de seguridad

	Roles habilitados	Roles inhabilitados
Grupos habilitados	L1, L2, L3, L4	L1, L3
Grupos inhabilitados	L1, L2	L1

La tabla siguiente muestra el valor de la etiqueta de seguridad combinada para cualquier intento de acceso de Paul, en función de los diferentes valores de la política de seguridad SECUR_POL.

Tabla 12. Etiquetas de seguridad combinadas como función de los valores de la política de seguridad

	Roles habilitados	Roles inhabilitados
Grupos habilitados	TS:(A, B, X, Y)	TS:(A, X)
Grupos inhabilitados	S:(A, B)	C:A

ALTER SEQUENCE

La sentencia ALTER SEQUENCE puede utilizarse para cambiar una secuencia.

Una secuencia puede cambiarse de las formas siguientes:

- 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 se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

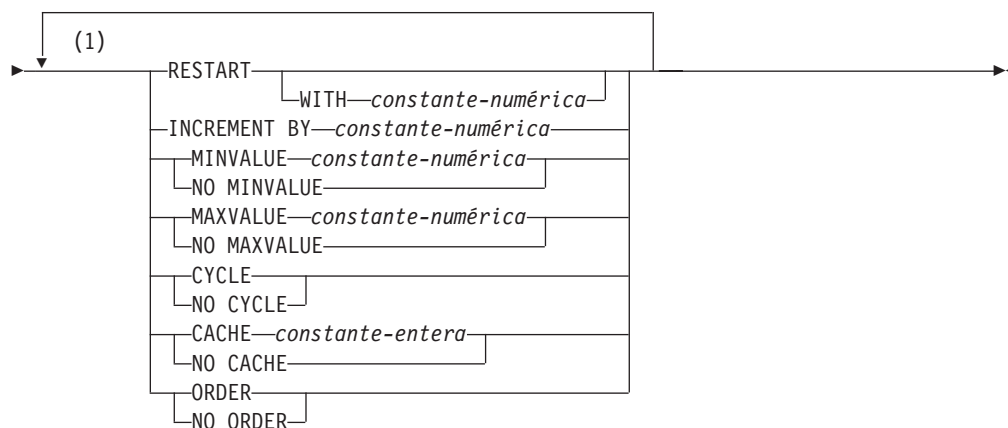
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTER para la secuencia que se va a modificar
- Privilegio ALTERIN para el esquema especificado implícita o explícitamente
- Autorización DBADM

Sintaxis

►► ALTER SEQUENCE *nombre-secuencia* ►►



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.

Descripción

nombre-secuencia

Identifica la secuencia que va a cambiarse. El nombre (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva una secuencia existente en el servidor actual. Si en el esquema especificado implícita o explícitamente no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704). El *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 a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

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 a la secuencia (SQLSTATE 42815). El valor no debe superar el valor de una constante de enteros grande (SQLSTATE 42820) ni contener dígitos que no sean 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 positivo, es una secuencia ascendente tras la sentencia ALTER.

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 a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser 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.

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 a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser 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 reducen 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).

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.

ORDER o NO ORDER

Especifica si los números de secuencia deben generarse según el orden de petición.

ALTER SEQUENCE

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.

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, descarte la secuencia y vuelva a crearla especificando el tipo de datos necesario para la nueva secuencia.
- Todos los valores almacenados en antememoria se pierden cuando se modifica una secuencia.
- 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.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede utilizarse una coma para separar varias opciones en una secuencia.
 - Se puede especificar NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER en lugar de NO MINVALUE, NO MAXVALUE, NO CYCLE, NO CACHE y NO ORDER, respectivamente

Ejemplo

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 NEXT VALUE FOR ORG_SEQ, ORG.* FROM ORG
```

ALTER SERVER

La sentencia ALTER SERVER se utiliza para modificar la definición o configuración de una fuente de datos.

Esta sentencia puede utilizarse para realizar los cambios siguientes:

- 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 cambios en las fuentes que persistirán en varias conexiones con la base de datos federada.

En esta sentencia, la palabra SERVER y los nombres de parámetro que terminan por *-servidor* sólo hacen referencia a las fuentes de datos de un sistema federado. No hacen referencia al servidor federado de ese sistema ni a los servidores de aplicaciones DRDA.

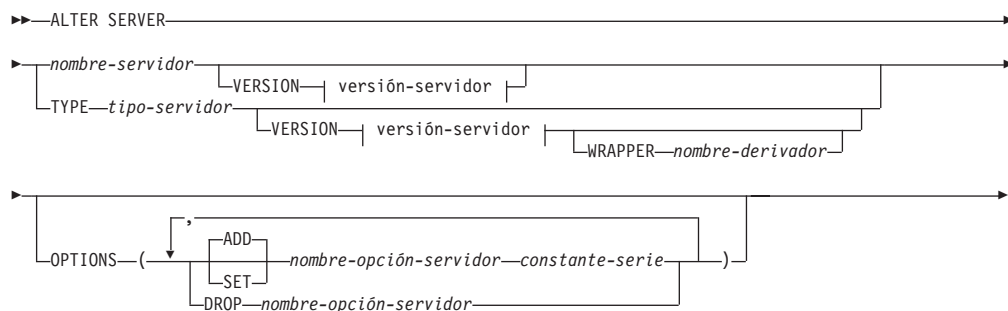
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

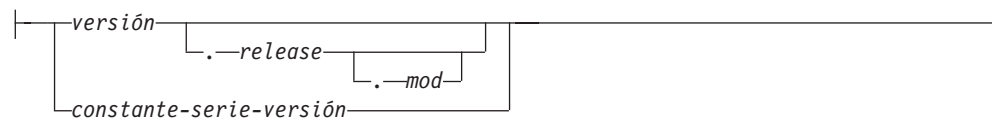
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

Sintaxis



versión-servidor:



Descripción

nombre-servidor

Identifica el nombre del servidor federado para la fuente de datos a la que se deben aplicar los cambios que se están solicitando. La fuente de datos debe ser una descrita en el catálogo.

ALTER SERVER

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. El valor debe ser un entero.

release

Especifica el número de release de la versión indicada por *versión*. El valor debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. El valor 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 estar formada por 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 deben aplicar los cambios que se están solicitando.

VERSION

Después de *tipo-servidor*, VERSION y su parámetro especifican la versión de las fuentes de datos para las cuales se deben habilitar, restaurar o descartar las opciones de servidor.

WRAPPER *nombre-derivador*

Especifica el nombre del derivador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*. El derivador debe aparecer en la lista del catálogo.

OPTIONS

Indica las opciones de servidor que se deben habilitar, restaurar o descartar 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.

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 debe habilitar o restaurar.

constante-serie

Especifica un valor para *nombre-opción-servidor* como una constante de serie de caracteres.

DROP *nombre-opción-servidor*

Descarta una opción de servidor.

Notas

- 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 descarta 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 (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
 - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o 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 uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
 - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- Si la opción de servidor se establece en un valor para un tipo de fuente de datos y en otro valor para una instancia de este tipo, el segundo valor altera temporalmente el primero para la instancia. Por ejemplo, supongamos que PLAN_HINTS se establece en 'Y' para el tipo de servidor ORACLE y en 'N' para una fuente de datos Oracle denominada DELPHI. Esta configuración provoca que se habiliten las indicaciones de planes en todas las fuentes de datos Oracle excepto en DELPHI.
- Sólo se puede ejecutar ALTER SET o ALTER DROP en opciones de servidor de una categoría de fuentes de datos que se ha habilitado antes por una operación de opción de servidor ALTER ADD (SQLSTATE 42704).
- Al modificar la versión del servidor, DB2 no comprueba que la versión del servidor especificado coincida con la versión del servidor remoto. Si se especifica una versión de servidor incorrecta se pueden provocar errores de SQL cuando se accede a apodos que pertenecen a la definición del servidor DB2. Esto puede producirse con mucha probabilidad cuando se especifica una versión de servidor posterior a la versión del servidor remoto. En ese caso, al acceder a apodos que pertenecen a la definición del servidor, puede que DB2 envíe SQL que el servidor remoto no reconoce.

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. Suponga también que la CPU del servidor federado local es el doble de rápida que la CPU de fuente de datos. 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 la fuente de datos Documentum denominada DCTM_SVR_ASIA se ha cambiado a la Versión 4.

```
ALTER SERVER DCTM_SVR_ASIA
VERSION 4
```

ALTER SERVICE CLASS

La sentencia ALTER SERVICE CLASS modifica la definición de una clase de servicio.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

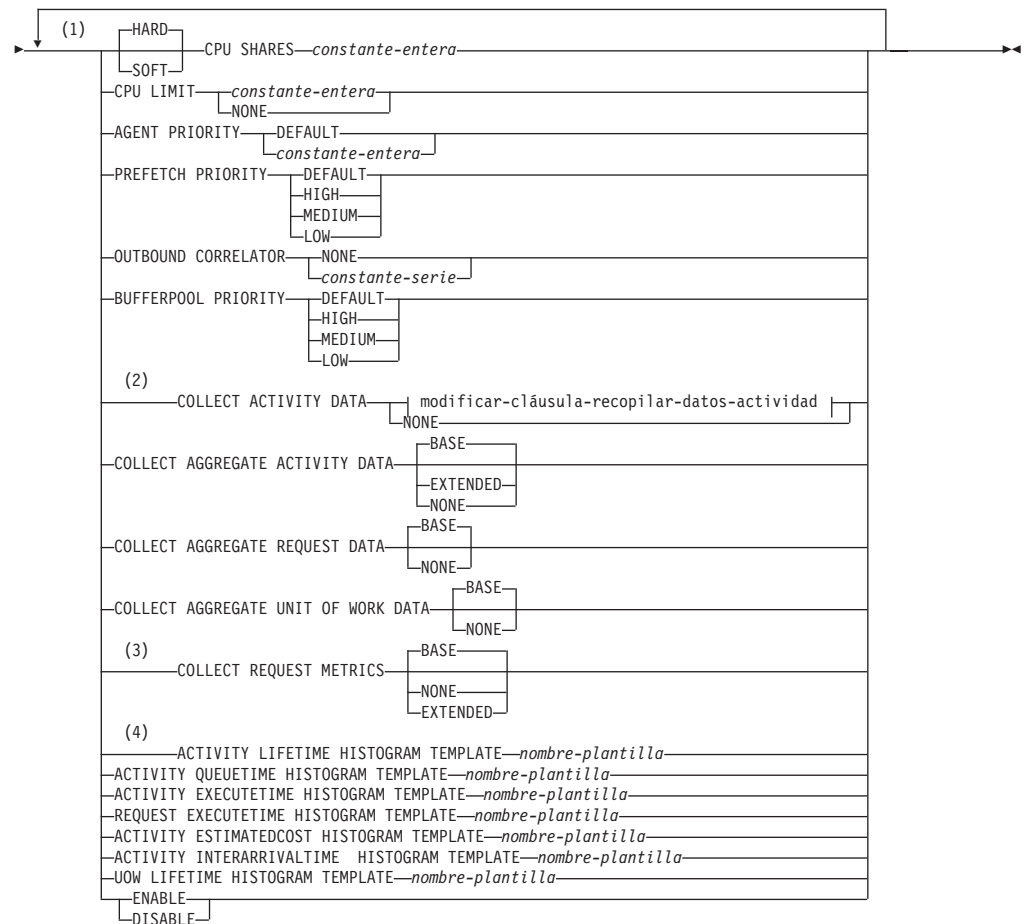
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

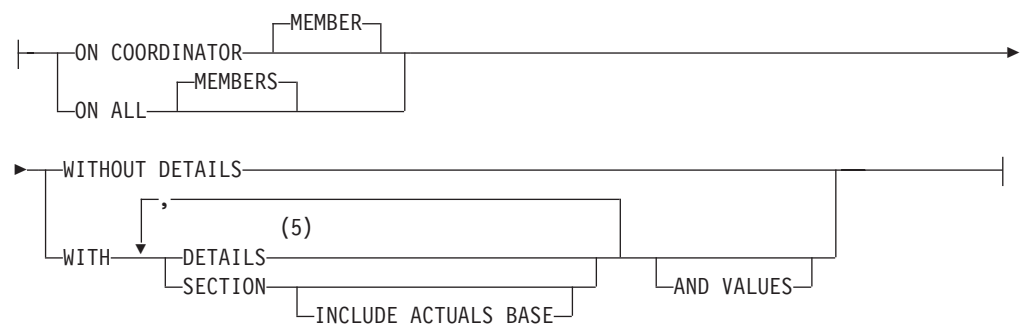
- Autoridad SQLADM, únicamente si todas las cláusulas de modificación son cláusulas COLLECT.
- Autorización WLMADM
- Autorización DBADM

Sintaxis

▶▶ALTER SERVICE CLASS—*nombre-clase-servicio*—[UNDER—*nombre-superclase-servicio*]—▶▶



modificar-cláusula-recopilar-datos-actividad:



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.
- 2 Todas las cláusulas COLLECT excepto COLLECT REQUEST METRICS sólo son válidas para una subclase de servicio.
- 3 La cláusula COLLECT REQUEST METRICS sólo es válida para una superclase de servicio.
- 4 Las cláusulas HISTOGRAM TEMPLATE sólo son válidas para una subclase de servicio.

ALTER SERVICE CLASS

- 5 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-clase-servicio

Identifica la clase de servicio que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-clase-servicio* debe identificar una clase de servicio que exista en la base de datos (SQLSTATE 42704). Para modificar una subclase de servicio, el *nombre-superclase-servicio* debe especificarse utilizando la cláusula UNDER.

UNDER *nombre-superclase-servicio*

La cláusula sólo se utiliza para modificar una subclase de servicio. El *nombre-superclase-servicio* identifica la superclase de servicio de la subclase de servicio que exista en la base de datos (SQLSTATE 42704).

SOFT CPU SHARES *constante-entera* **o** **HARD CPU SHARES** *constante-entera*

Especifica el número de cuotas de recursos de CPU que el asignador de WLM asigna a esta clase de servicio cuando el trabajo se ejecuta dentro de esta clase de servicio. Los valores válidos de la *constante-entera* son enteros entre 1 y 65535. Calificar **CPU SHARES** con la palabra clave **HARD** o especificar **CPU SHARES** sin calificarlo con la palabra clave **HARD** o **SOFT** indica que se asignarán cuotas de CPU fijas a esta clase de servicio. Especificar la palabra clave **SOFT** indica que se asignarán cuotas de CPU flexibles a esta clase de servicio. Para utilizar cuotas de CPU fijas y flexibles con el asignador del gestor de carga de trabajo de DB2, debe habilitar el parámetro de configuración del gestor de bases de datos **wlm_disp_cpu_shares**.

CPU LIMIT *constante-entero* **o** **CPU LIMIT NONE**

Especifica el porcentaje máximo de los recursos de CPU que el asignador de WLM puede asignar a esta clase de servicio. Los valores válidos de la *constante-entera* son enteros entre 1 y 100. Puede especificar también **CPU LIMIT NONE** para indicar que no hay ningún límite de CPU.

AGENT PRIORITY DEFAULT **o** **AGENT PRIORITY** *constante-entero*

Especifica la prioridad del sistema operativo (delta) relativa de los agentes que se ejecutan en la clase de servicio o la prioridad normal de las hebras que se ejecutan en DB2. El valor por omisión es **DEFAULT**.

Importante: El parámetro de configuración del gestor de bases de datos **agentpri** está en desuso desde la Versión 9.5. Todavía se puede utilizar en los clientes y servidores de datos anteriores a la Versión 9.5. Además, la prioridad de agente para la clase de servicio de WLM ha quedado en desuso en la Versión 10.1 y podría eliminarse en un release futuro. Comience a utilizar la función de asignador de WLM en lugar de la prioridad de agente. Para obtener más información, consulte "Prioridad de agente de clases de servicio ha quedado en desuso" en *Novedades en DB2 Versión 10.1*.

Cuando se establece en **DEFAULT**, no se adopta ninguna acción especial y los agentes de la clase de servicio se planifican con arreglo a la prioridad normal con la que el sistema operativo planifica todas las hebras de DB2. Cuando se establece este parámetro en un valor que no sea **DEFAULT**, los agentes se establecen en una prioridad que sea igual a la prioridad normal más **AGENT PRIORITY** cuando comienza la siguiente actividad. Por ejemplo, si la prioridad normal es 20 y **AGENT PRIORITY** se establece en -10, la prioridad de los agentes de la clase de servicio se establece en $20 - 10 = 10$.

Nota: No se pueden utilizar conjuntamente comparticiones de prioridad de agente y asignador de WLM. Cuando el asignador se habilita estableciendo el valor del parámetro de configuración de gestor de bases de datos **wlm_dispatcher** en ON, la prioridad de agente especificada se ignora y la prioridad de agente se establece en el valor por omisión hasta que el asignador está inhabilitado.

El gestor de carga de trabajo de DB2 (WLM) no asigna prioridades de agente de clase de servicio a tareas que se realicen dentro de un proceso en protegido (FMP). Los procedimientos delimitados no ejecutan la lógica dentro de una clase de servicio. Los procedimientos delimitados se ejecutan dentro de DB2 FMP y los agentes de DB2 no realizan estas tareas. A modo de recordatorio, DB2 WLM controla los agentes de DB2.

En sistemas operativos UNIX y Linux, los valores válidos son DEFAULT y de -20 a 20 (SQLSTATE 42615). Los valores negativos denotan una prioridad relativa más alta. Los valores positivos denotan una prioridad relativa más baja.

En sistemas operativos Windows, los valores válidos son DEFAULT y de -6 a 6 (SQLSTATE 42615). Los valores negativos denotan una prioridad relativa más baja. Los valores positivos denotan una prioridad relativa más alta.

Si AGENT PRIORITY es DEFAULT para una subclase de servicio, hereda el valor de AGENT PRIORITY de su superclase padre. AGENT PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032). AGENT PRIORITY debe establecerse en DEFAULT si se establece OUTBOUND CORRELATOR (SQLSTATE 42613).

Nota: En AIX, el propietario de instancia debe tener posibilidades CAP_NUMA_ATTACH y CAP_PROPAGATE para establecer una prioridad relativa más alta para los agentes en una clase de servicio utilizando AGENT PRIORITY. Para otorgar estas posibilidades, inicie la sesión como root y ejecute el mandato siguiente:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

En Solaris 10 o posterior, el propietario de instancia debe tener el privilegio proc_priocntl para establecer una prioridad relativa más alta para los agentes en una clase de servicio con AGENT PRIORITY. Para otorgar este privilegio, inicie la sesión como root y ejecute el siguiente mandato:

```
usermod -K defaultpriv=basic,proc_priocntl db2user
```

En este ejemplo, se añade proc_priocntl al conjunto de privilegios por omisión del usuario db2user.

Asimismo, cuando DB2 se está ejecutando en una zona de Solaris no global, el privilegio proc_priocntl debe añadirse al conjunto de privilegios de límite de la zona. Para otorgar este privilegio a la zona, inicie la sesión como root y ejecute el siguiente mandato:

```
global# zonecfg -z db2zone
zonecfg:db2zone> set limitpriv="default,proc_priocntl"
```

En este ejemplo, proc_priocntl se añade al conjunto de privilegios de límite de la zona db2zone.

En Solaris 9, no existe ningún recurso para DB2 para incrementar la prioridad relativa de los agentes. Actualice a Solaris 10 o posterior para utilizar la prioridad de agente de la clase de servicio.

ALTER SERVICE CLASS

PREFETCH PRIORITY DEFAULT | HIGH | MEDIUM | LOW

Este parámetro controla la prioridad con la que los agentes de la clase de servicio pueden someter sus peticiones de captación previa. Los valores válidos son HIGH, MEDIUM, LOW o DEFAULT (SQLSTATE 42615). HIGH, MEDIUM y LOW significan que las peticiones de captación previa se someterán a las colas de prioridad alta, media y baja, respectivamente. Los captadores previos vacían la cola de prioridad de la alta a la baja. Los agentes de la clase de servicio envían sus peticiones de captación previa al nivel de PREFETCH PRIORITY cuando comienza la siguiente actividad. Si se altera PREFETCH PRIORITY después de someter una petición de captación previa, no se cambiará la prioridad de petición. El valor por omisión es DEFAULT, que se correlaciona internamente con MEDIUM para la superclases de servicio. Si se especifica DEFAULT para una subclase de servicio, hereda el valor de PREFETCH PRIORITY o de su superclase padre.

PREFETCH PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032).

OUTBOUND CORRELATOR NONE o OUTBOUND CORRELATOR *constante-serie*

Especifica si asociar o no hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa.

Si se establece OUTBOUND CORRELATOR en *constante-serie* para la superclase de servicio y se establece OUTBOUND CORRELATOR NONE para una subclase de servicio, la subclase de servicio hereda el OUTBOUND CORRELATOR de su padre. OUTBOUND CORRELATOR debe establecerse en NONE si AGENT PRIORITY no se establece en DEFAULT (SQLSTATE 42613).

OUTBOUND CORRELATOR NONE

Para una superclase de servicio, especifica que no hay ninguna asociación de clase de servicio de gestor de carga de trabajo externa con esta clase de servicio y, para una subclase de servicio, especifica que la asociación de clase de servicio de gestor de carga de trabajo externa es la misma que la de su padre.

OUTBOUND CORRELATOR *constante-serie*

Especifica la *constante-serie* que ha de utilizarse como correlacionador para asociar hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa. El gestor de carga de trabajo externa debe estar activa (SQLSTATE 5U030). El gestor de carga de trabajo externa debería configurarse para reconocer el valor de *constante-serie*.

BUFFERPOOL PRIORITY DEFAULT | HIGH | MEDIUM | LOW

Este parámetro controla la prioridad de agrupación de almacenamientos intermedios de páginas captadas por actividades de esta clase de servicio. Los valores válidos son HIGH, MEDIUM, LOW o DEFAULT (SQLSTATE 42615). La probabilidad de intercambio de las páginas captadas por actividades de una clase de servicio con una prioridad de agrupación de almacenamientos intermedios más alta es menor que la captura de páginas mediante actividades en una clase de servicio con una prioridad de agrupación de almacenamientos intermedios más baja. Si se especifica DEFAULT para una subclase de servicio, hereda el valor de BUFFERPOOL PRIORITY de su superclase padre.

BUFFERPOOL PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032).

COLLECT ACTIVITY DATA

Especifica que la información relacionada con cada actividad que se ejecuta en esta clase de servicio ha de enviarse a cualquier supervisor de sucesos de

actividades activo cuando se haya completado la actividad. La cláusula COLLECT ACTIVITY DATA sólo es válida para una subclase de servicio.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad que se ejecuta en la clase de servicio deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la ejecución completa la actividad. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. Los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actuals**.

La configuración efectiva para la recopilación de datos reales de sección es la combinación de la cláusula INCLUDE ACTUALS (especificada en WORK ACTION, SERVICE CLASS o WORKLOAD), el parámetro de configuración de base de datos **section_actuals** y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del parámetro de configuración de la base de datos **section_actuals** es NONE y <collectsectionactuals> se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

ALTER SERVICE CLASS

BASE especifica que han de habilitarse y recopilarse los datos reales siguientes durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

NONE

Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que se ejecutan en esta clase de servicio.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividades agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos `wlm_collect_int`. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. La cláusula COLLECT AGGREGATE ACTIVITY DATA sólo es válida para una subclase de servicio.

BASE

Especifica que los datos de actividades agregados básicos deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal

Nota: Sólo las actividades a las que se ha aplicado un umbral SQLTEMPSPACE participan en esta marca de límite superior.

- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

NONE

Especifica que no debe capturarse ningún dato de actividad agregados para esta clase de servicio.

COLLECT AGGREGATE REQUEST DATA

Especifica que los datos de peticiones agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo especificado por medio del parámetro de configuración de base de datos

wlm_collect_int. El valor por omisión es COLLECT AGGREGATE REQUEST DATA NONE. La cláusula COLLECT AGGREGATE REQUEST DATA sólo es válida para una subclase de servicio.

BASE

Especifica que los datos de peticiones agregados básicos deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo.

NONE

Especifica que no debe capturarse ningún dato de petición agregado para esta clase de servicio.

COLLECT AGGREGATE UNIT OF WORK DATA

Especifica que deben capturarse los datos agregados de unidad de trabajo para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión cuando se especifica COLLECT AGGREGATE UNIT OF WORK DATA es COLLECT AGGREGATE UNIT OF WORK DATA BASE.

BASE

Especifica que deben capturarse los datos agregados básicos de unidad de trabajo para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos agregados básicos de unidad de trabajo incluyen:

- Unidad de histograma de ciclo de vida de trabajo

NONE

Especifica que no debe recopilarse ningún dato agregado de unidad de trabajo para esta clase de servicio.

COLLECT REQUEST METRICS

Especifica que la métrica del supervisor debe recopilarse para cualquier petición enviada por una conexión asociada con la superclase de servicio especificada y enviarse a los supervisores de sucesos estadísticos y de unidad de trabajo, si están activos. El valor por omisión es COLLECT REQUEST METRICS NONE. La cláusula COLLECT REQUEST METRICS sólo es válida para una superclase de servicio (SQLSTATE 50U44).

Nota: La configuración de la recopilación de métricas de petición efectiva es la combinación del atributo que se especifica mediante la cláusula COLLECT REQUEST METRICS en la superclase de servicio asociada a la conexión que envía la petición y del parámetro de configuración de base de datos **mon_req_metrics**. Si el atributo de superclase de servicio o bien el parámetro de configuración tiene un valor distinto de NONE, se recopilarán métricas para la petición.

BASE

Especifica que la métrica básica se recopilará para cualquier petición enviada por una conexión asociada con la superclase de servicio.

EXTENDED

Especifica que la métrica básica se recopilará para cualquier petición enviada por una conexión asociada con la superclase de servicio.

Asimismo, especifica que los valores de los siguientes elementos de supervisor deben determinarse con granularidad adicional:

- **total_section_time**

ALTER SERVICE CLASS

- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

NONE

Especifica que no se recopilará ninguna métrica para las peticiones enviadas por una conexión asociada con la superclase de servicio.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos de actividades de DB2 en ejecución en la clase de servicio durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la clase de servicio están en cola durante un intervalo específico. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en el miembro coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las peticiones de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la petición se recopila en este histograma en cada miembro donde se ejecuta la petición. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE REQUEST DATA con la opción BASE. Esta cláusula sólo es válida para una subclase de servicio.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML en ejecución en la clase de servicio. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

UOW LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma que se utiliza para recopilar datos estadísticos sobre la duración, en milisegundos, de unidades de trabajo que se ejecutan en la clase de servicio durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE UNIT OF WORK DATA con la opción BASE.

ENABLE o DISABLE

Especifica si pueden correlacionarse o no conexiones y actividades con la clase de servicio.

ENABLE

Las conexiones y actividades pueden correlacionarse con la clase de servicio.

DISABLE

Las conexiones y actividades no pueden correlacionarse con la clase de servicio. Las conexiones o actividades correlacionadas a una clase de servicio inhabilitada se rechazarán (SQLSTATE 5U028). Cuando se inhabilita una superclase de servicio, también se inhabilitarán sus subclases de servicio. Cuando vuelve a habilitarse la superclase de servicio, sus subclases de servicio vuelven a los estados definidos en el catálogo del sistema. Una clase de servicio por omisión no puede inhabilitarse (SQLSTATE 5U032).

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (plantilla de histograma)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (clase de servicio)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (umbral)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (conjunto de acciones de trabajo)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (conjunto de clases de trabajo)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (carga de trabajo)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

ALTER SERVICE CLASS

Notas

- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todos los miembros. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.
- Después de confirmar una sentencia ALTER SERVICE CLASS, los cambios en AGENT PRIORITY, PREFETCH PRIORITY, OUTBOUND CORRELATOR y COLLECT surtirán efecto para la siguiente actividad nueva de la clase de servicio. Las actividades de la clase de servicio siguen completando su trabajo utilizando los valores antiguos.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - o Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - o Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* Modifique la prioridad de agente de los agentes de la superclase PETSALLES para aumentarla del valor DEFAULT al valor más alto posible (se muestra para los sistemas operativos UNIX y Linux; en sistemas operativos Windows, sustituya 6).

```
ALTER SERVICE CLASS PETSALLES AGENT PRIORITY -20
```

- *Ejemplo 2:* Modifique superclase de servicio BARNSALLES y añada un correlacionador de salida 'osLowPriority'. Las hebras que se ejecutan en la superclase de servicio y en sus subclases de servicio tendrán asociado a las mismas el correlacionador de salida 'osLowPriority'.

```
ALTER SERVICE CLASS BARNSALLES OUTBOUND CORRELATOR 'osLowPriority'
```

ALTER STOGROUP

La sentencia ALTER STOGROUP se utiliza para alterar la definición de un grupo de almacenamiento.

Invocación

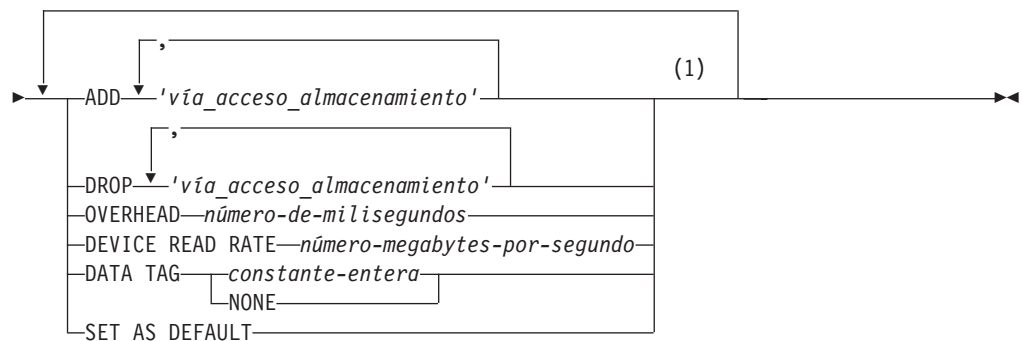
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis

► ALTER STOGROUP *nombre-grupo-almacenamiento* ►



Notas:

- 1 Cada cláusula sólo puede especificarse una vez.

Descripción

nombre-grupo-almacenamiento

Identifica el grupo de almacenamiento que se debe modificar; *nombre-grupo-almacenamiento* debe identificar un grupo de almacenamiento que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte.

ADD

Especifica que se deben añadir una o más vías de acceso de almacenamiento nuevas al grupo de almacenamiento especificado.

vía-acceso-almacenamiento

Constante de tipo serie que especifica una vía de acceso absoluta o la letra de una unidad (sólo en los sistemas operativos Windows) donde se crearán los contenedores para los espacios de tablas de almacenamiento automático. La serie puede incluir expresiones de partición de base de datos para especificar la información del número de partición de base de

ALTER STOGROUP

datos en la vía de acceso de almacenamiento. Para el rendimiento previsible, las vías de acceso de almacenamiento añadidas a un grupo de almacenamiento han de tener características de soportes similares.

La longitud máxima de una vía de acceso de almacenamiento es de 175 caracteres (SQLSTATE 54036). La vía de acceso de almacenamiento que se está añadiendo debe ser válida según las normas de denominación y debe ser accesible (SQLSTATE 57019). De igual modo, en un entorno de base de datos particionada, la vía de acceso de almacenamiento debe existir y ser accesible en cada partición de base de datos (SQLSTATE 57019).

DROP

Especifica que una o más vías de almacenamiento deben eliminarse del grupo de almacenamiento en cuestión. Si los espacios de tablas están utilizando de forma activa una vía de acceso de almacenamiento que se va a descartar, el estado de la vía de acceso de almacenamiento cambia de En uso a Descarte pendiente y se impedirá que se utilice en el futuro la vía de acceso de almacenamiento.

La cláusula *DROP vía-acceso-almacenamiento* no se soporta en un entorno DB2 pureScale (SQLSTATE 56038).

vía-acceso-almacenamiento

Constante de serie que especifica una vía de acceso absoluta o la letra de una unidad (sólo en los sistemas operativos Windows). La serie puede incluir expresiones de partición de base de datos para especificar la información del número de partición de base de datos en la vía de acceso de almacenamiento.

La vía de acceso de almacenamiento que se está descartando debe existir actualmente en el grupo de almacenamiento (SQLSTATE 57019) y no puede tener ya el estado Descarte pendiente (SQLSTATE 55073).

OVERHEAD *número-de-milisegundos*

Especifica el tiempo de latencia, búsqueda de disco y uso de controlador de E/S. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. El valor *número-de-milisegundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todas las vías de acceso de almacenamiento, establezca el valor en un literal numérico que represente el promedio de todas las vías de acceso de almacenamiento que pertenecen al grupo de almacenamiento.

DEVICE READ RATE *número-megabytes-por-segundo*

Representa la especificación de dispositivo para la velocidad de transferencia de lectura en megabytes por segundo. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. El valor de *número-megabytes-por-segundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todas las vías de acceso de almacenamiento, establezca el valor en un literal numérico que represente el promedio de todas las vías de acceso de almacenamiento que pertenecen al grupo de almacenamiento.

DATA TAG *constante-entero* o DATA TAG NONE

Especifica un código para los datos en un grupo de almacenamiento determinado. Este valor se puede utilizar como parte de una configuración de WLM en una definición de clase de trabajo o se puede referenciar en una definición de umbral. Para obtener más información, consulte las sentencias CREATE WORK CLASS SET, ALTER WORK CLASS SET, CREATE THRESHOLD y ALTER THRESHOLD.

constante-entera

Los valores de *constante-entera* son los enteros del 1 al 9.

NONE

Si se especifica NONE, no hay ningún código de datos.

SET AS DEFAULT

Especifica que el grupo de almacenamiento que se modifica se designa como grupo de almacenamiento por omisión. Sólo puede haber un grupo de almacenamiento designado como grupo de almacenamiento por omisión. No se produce ningún efecto en los espacios de tablas existentes que utilizan ese grupo de almacenamiento. Los espacios de tablas de almacenamiento automático utilizan el grupo de almacenamiento por omisión designado cuando no se especifica ningún grupo de almacenamiento durante la creación de espacios de tablas y un espacio de tablas gestionado por base de datos se convierte en almacenamiento automático gestionado durante la restauración redirigida.

Normas

- Un grupo de almacenamiento debe tener, como mínimo, una vía de acceso de almacenamiento. No se pueden descartar del grupo de almacenamiento todas las vías de acceso de almacenamiento (SQLSTATE 428HH).
- No se puede ejecutar la sentencia ALTER STOGROUP mientras se está añadiendo un servidor de particiones de base de datos (SQLSTATE 55071).
- Un grupo de almacenamiento puede tener un máximo de 128 vías de acceso de almacenamiento definidas (SQLSTATE 5U009).
- Una transacción puede tener, como máximo, una sentencia ALTER STOGROUP por cada grupo de almacenamiento. En el caso del grupo de almacenamiento por omisión, puede haber, como máximo, una sentencia ALTER DATABASE o una sentencia ALTER STOGROUP en el grupo de almacenamiento por omisión (SQLSTATE 25502).

Notas

- *Adición de vías de acceso de almacenamiento nuevas:* al añadir vías de acceso de almacenamiento nuevas:
 - Los espacios de tablas REGULAR (normales) y LARGE (grandes) existentes que utilicen este grupo de almacenamiento no utilizarán estas nuevas vías de acceso inicialmente. El gestor de bases de datos puede elegir crear nuevos contenedores de espacio de tablas en estas vías de acceso únicamente si se produce una condición de falta de espacio. Puede emitir sentencias ALTER TABLESPACE REBALANCE para que los espacios de tablas existentes los dividan en bandas por las vías de acceso de almacenamiento recién añadidas.
 - Los espacios de tablas temporales existentes gestionados por almacenamiento automático no utilizan automáticamente las vías de acceso de almacenamiento nuevas. Se debe detener con normalidad la base de datos y reiniciarla para que los contenedores de estos espacios de tablas utilicen las nuevas vías de acceso de almacenamiento. Como alternativa, los espacios de tablas temporales pueden descartarse y volver a crearse. Al crearlos, estos espacios de tablas utilizan automáticamente todas las vías de acceso de almacenamiento que disponen de espacio libre suficiente.
- *Cálculo del espacio libre:* cuando se calcula el espacio libre para una vía de acceso de almacenamiento en una partición de base de datos, el gestor de bases de datos comprueba la existencia de los siguientes directorios o puntos de montaje en la vía de acceso de almacenamiento, y utiliza la primera que se encuentre.

ALTER STOGROUP

```
<vía acceso almacenamiento>/<nombre instancia>/NODE####/<nombre base datos>  
<vía acceso almacenamiento>/<nombre instancia>/NODE####  
<vía acceso almacenamiento>/<nombre instancia>  
<vía acceso almacenamiento>
```

Donde:

- <vía de acceso almacenamiento> es una vía de acceso de almacenamiento asociada con la base de datos.
 - <nombre instancia> es la instancia en la que reside la base de datos.
 - NODE#### corresponde al número de partición de base de datos (por ejemplo, NODE0000 o NODE0001).
 - <nombre base de datos> es el nombre de la base de datos.
- **Aislamiento de varias particiones de base de datos en una vía de acceso de almacenamiento:** los sistemas de archivos se pueden montar en un punto bajo la vía de acceso de almacenamiento, y el gestor de bases de datos reconocerá que la cantidad real de espacio libre disponible para los contenedores del espacio de tablas puede no ser la misma que la asociada con el directorio de la vía de acceso de almacenamiento.

Considere un ejemplo donde existan dos particiones de base de datos lógicas en un sistema físico, y haya una única vía de acceso de almacenamiento (/db2data). Cada partición de base de datos utilizará esta vía de acceso de almacenamiento, pero es posible que se desee aislar los datos de cada partición en su propio sistema de archivos. En este caso, se puede crear un sistema de archivos diferente para cada partición y se puede montar en /db2data/<instancia>/NODE####. Al crear contenedores en la vía de acceso de almacenamiento y determinar el espacio libre, el gestor de bases de datos no recuperará la información sobre el espacio libre para /db2data, sino que la recuperará para el directorio /db2data/<instancia>/NODE#### correspondiente.

- **Descarte de una vía de acceso de almacenamiento que utilizan uno o más espacios de tablas:** al descartar una vía de acceso de almacenamiento que están utilizando uno o más espacios de tablas, el estado de la vía de acceso cambia de En uso a Descarte pendiente. La vía de acceso no crecerá más. Antes de que se pueda eliminar completamente la vía de acceso del grupo de almacenamiento, es necesario reequilibrar todos los espacios de tablas afectados (con la cláusula REBALANCE o la sentencia ALTER TABLESPACE), para que los datos de sus contenedores se retiren de la vía de acceso. El reequilibrado sólo recibe soporte para los espacios de tablas REGULAR y LARGE. Descarte y vuelva a crear los espacios de tablas temporales para que sus contenedores se eliminen de la vía de acceso descartada. Cuando ningún espacio de tablas esté utilizando ya la vía de acceso, se eliminará físicamente de la base de datos.

Para un entorno de base de datos particionada, la vía de acceso se mantiene de forma independiente en cada partición. Cuando ya no se esté utilizando una vía de acceso en una partición de base de datos dada, se eliminará físicamente de dicha partición. Puede que otras particiones todavía muestren la vía de acceso con el estado "Descarte pendiente". Se puede obtener la lista de los espacios de tablas de almacenamiento automático que utilizan vías de acceso de almacenamiento con estado de descarte pendiente emitiendo la sentencia de SQL siguiente:

```
SELECT DISTINCT TBSP_NAME, TBSP_ID, TBSP_CONTENT_TYPE  
FROM TABLE(MON_GET_TABLESPACE(NULL,-2)) AS T  
WHERE TBSP_PATHS_DROPPED = 1
```

- **Descarte de una vía de acceso de almacenamiento que se haya añadido a un grupo de almacenamiento varias veces:** se puede añadir varias veces una vía de acceso de almacenamiento dada a una base de datos. Al utilizar la cláusula

DROP, si se especifica esa vía de acceso concreta una vez se descartarán del grupo de almacenamiento todas las instancias de la vía de acceso.

Ejemplos

- *Ejemplo 1:* añadir las unidades D y E al grupo de almacenamiento denominado COMPLIANCE.

```
ALTER STOGROUP COMPLIANCE ADD 'D:\', 'E:\'
```

- *Ejemplo 2:* cambiar el código de datos para el grupo de almacenamiento OPERATIONAL y designarlo como grupo de almacenamiento por omisión.

```
ALTER STOGROUP OPERATIONAL DATA TAG 3 SET AS DEFAULT
```

- *Ejemplo 3:* añadir una vía de acceso de almacenamiento que utilice una expresión de partición de base de datos para diferenciar las vías de acceso de almacenamiento en cada una de las particiones de base de datos.

```
ALTER STOGROUP TESTDATA ADD '/dataForPartition $N'
```

- *Ejemplo 4:* eliminar las vías de acceso /db2/filesystem1 y /db2/filesystem2 del grupo de almacenamiento TESTDATA.

```
ALTER STOGROUP TESTDATA DROP '/db2/filesystem1', '/db2/filesystem2'
```

ALTER TABLE

La sentencia ALTER TABLE modifica la definición de una tabla.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTER para la tabla que se debe modificar
- Privilegio CONTROL sobre la tabla que se debe modificar
- Privilegio ALTERIN para el esquema de la tabla
- Autorización DBADM

Para crear o descartar una clave foránea, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes para la tabla padre:

- Privilegio REFERENCES para la tabla
- Privilegio REFERENCES para todas las columnas de la clave padre especificada
- Privilegio CONTROL sobre la tabla
- Autorización DBADM

Para descartar la clave primaria o una restricción de unicidad en la tabla T, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla que sea dependiente de esta clave padre de T:

- Privilegio ALTER para la tabla
- Privilegio CONTROL sobre la tabla
- Privilegio ALTERIN para el esquema de la tabla
- Autorización DBADM

Para modificar una tabla con el fin de convertirla en una tabla de consulta materializada (utilizando una selección completa), los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla
- Autorización DBADM

y, como mínimo, una de las siguientes autorizaciones para cada tabla o vista identificada en la selección completa (excluyendo los privilegios de grupo):

- Privilegio SELECT y privilegio ALTER (incluyendo los privilegios de grupo) para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Privilegio SELECT para la tabla o vista y privilegio ALTERIN (incluyendo los privilegios de grupo) para el esquema de la tabla o vista
- Autorización DATAACCESS

Para modificar una tabla con el fin de que ya no sea una tabla de consulta materializada, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla o vista identificada en la selección completa que se utiliza para definir la tabla de consulta materializada:

- Privilegio ALTER para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Privilegio ALTERIN para el esquema de la tabla o vista
- Autorización DBADM

Para añadir una columna de tipo DB2SECURITYLABEL a una tabla, los privilegios que tiene el ID de autorización de la sentencia deben incluir al menos una etiqueta de seguridad de la política de seguridad asociada a la tabla.

Para eliminar la política de seguridad de una tabla, los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Para modificar una tabla a fin de enlazar una partición de datos, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para la tabla fuente:

- Privilegio SELECT para la tabla y privilegio DROPIN para el esquema de la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

y, como mínimo, una de las autorizaciones siguientes para la tabla de destino:

- Privilegios ALTER e INSERT para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

Para modificar una tabla a fin de desenlazar una partición de datos, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para la tabla de destino de la partición desenlazada:

- Autorización CREATETAB para la base de datos y privilegio USE para los espacios de tablas que utiliza la tabla, así como uno de los elementos siguientes:
 - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla nueva no existe
 - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla nueva hace referencia a un esquema existente
- Autorización DBADM

y, como mínimo, una de las autorizaciones siguientes para la tabla fuente:

- Privilegios SELECT, ALTER y DELETE para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

Para modificar una tabla para la activación no registrada inicialmente con tabla vacía, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegios ALTER y DELETE para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DBADM

ALTER TABLE

Para modificar una tabla protegida mediante una política de seguridad para la activación no registrada inicialmente con tabla vacía, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla
- Autorización DBADM

Para modificar una tabla al control de acceso de columnas y filas ACTIVATE y DEACTIVATE, los privilegios del ID de autorización de la sentencia deben incluir la autorización SECADM.

Para modificar una tabla con ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE, si esa tabla tiene el control de acceso a filas activado, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

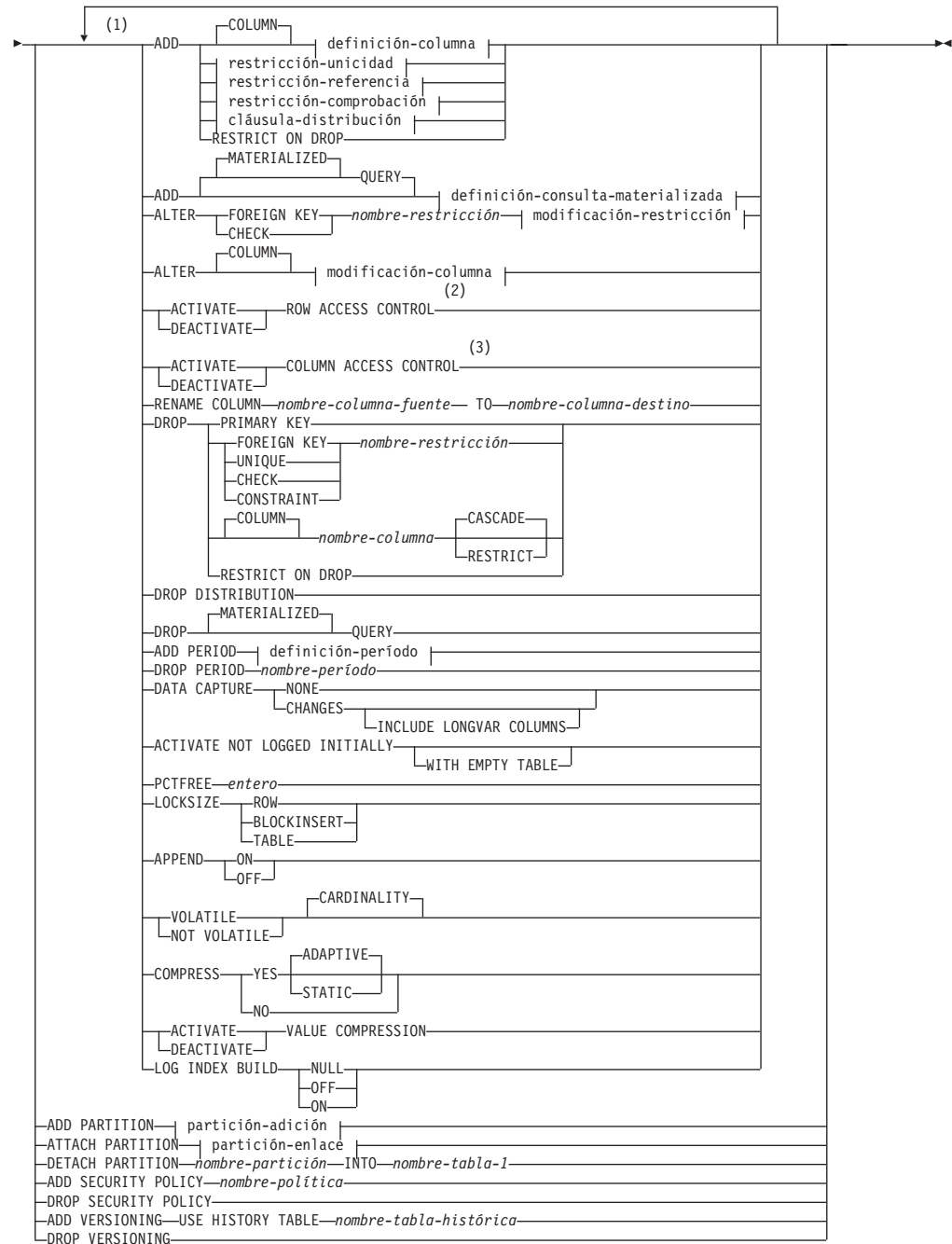
- Privilegio CONTROL sobre la tabla
- Autorización DBADM

Para modificar una tabla y convertirla en una tabla temporal de período de sistema (con la cláusula ADD VERSIONING) o modificar una tabla temporal de período de sistema cuando uno o varios de los cambios también provocan cambios en la tabla histórica asociada, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

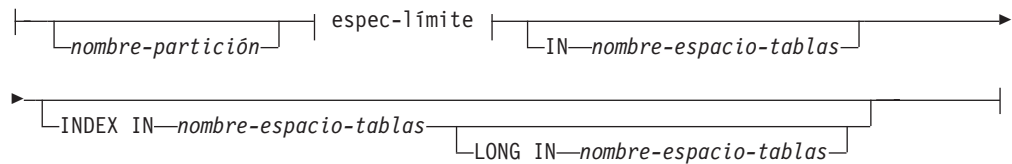
- Privilegio ALTER sobre la tabla histórica
- Privilegio CONTROL sobre la tabla histórica
- Privilegio ALTERIN sobre el esquema de la tabla histórica
- Autorización DBADM

Sintaxis

►►—ALTER TABLE—*nombre-tabla*—————→

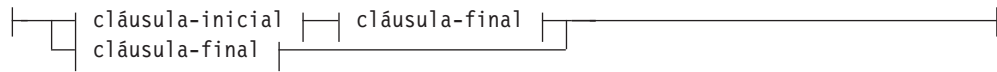


partición-adición:

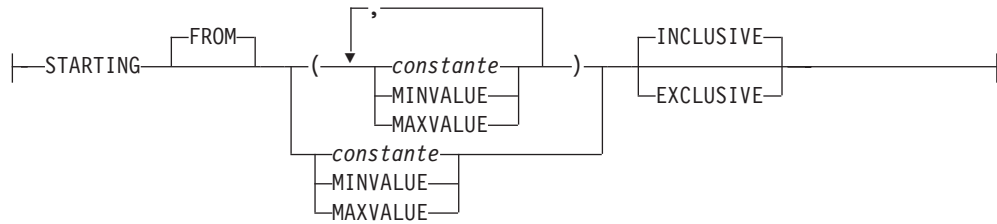


espec-límite:

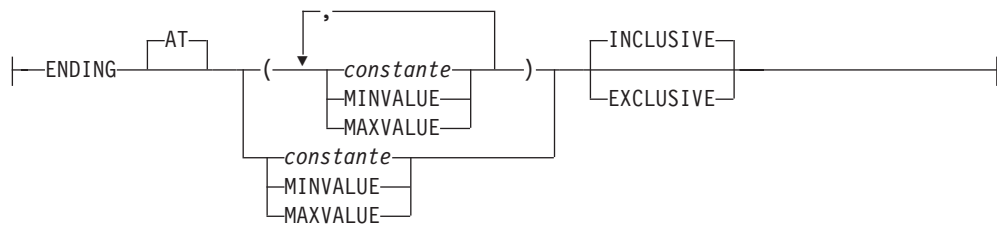
ALTER TABLE



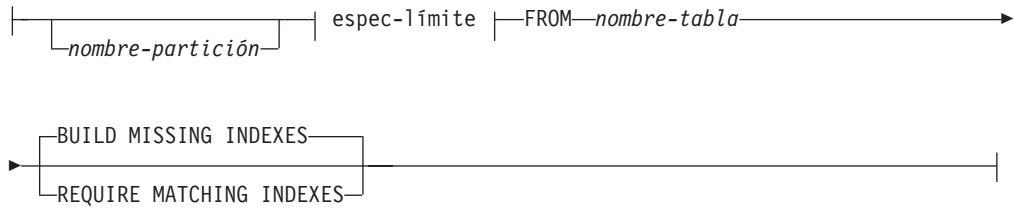
cláusula-inicial:



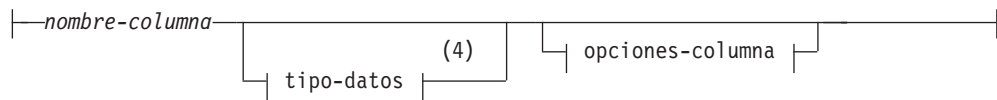
cláusula-final:



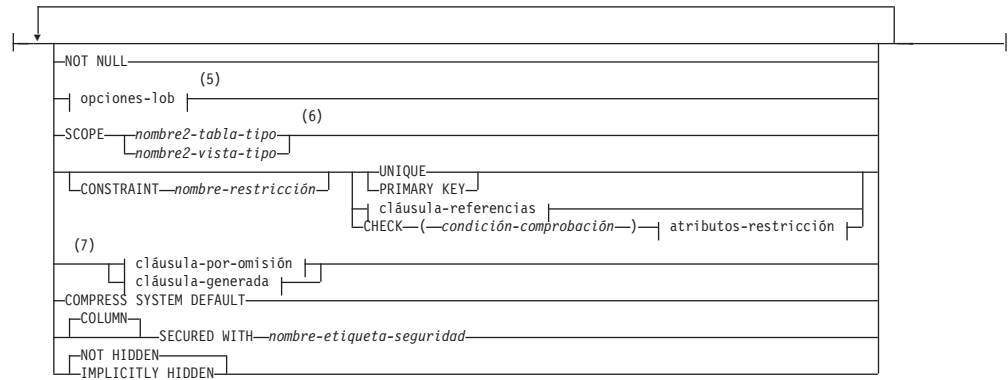
partición-enlace:



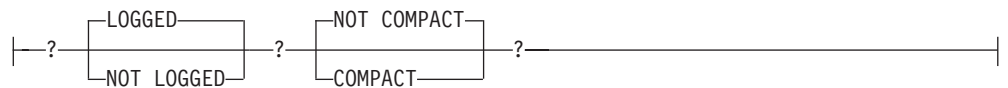
definición-columna:



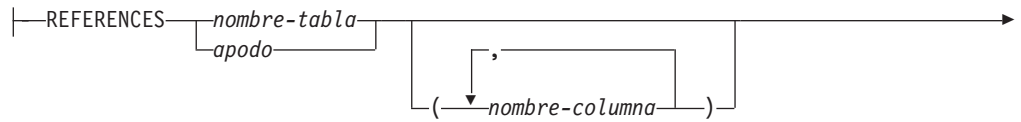
opciones-columna:



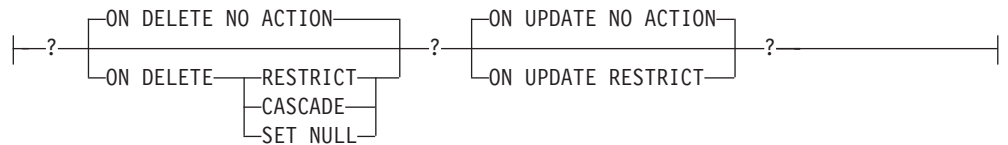
opciones-lob:



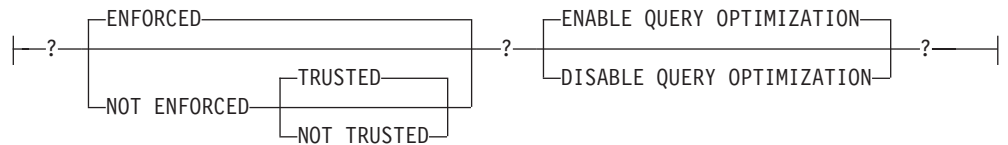
cláusula-referencias:



cláusula-norma:

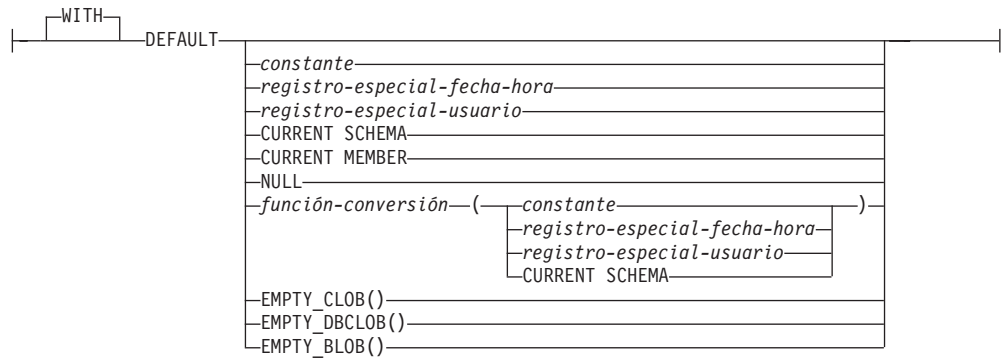


atributos-restricción:

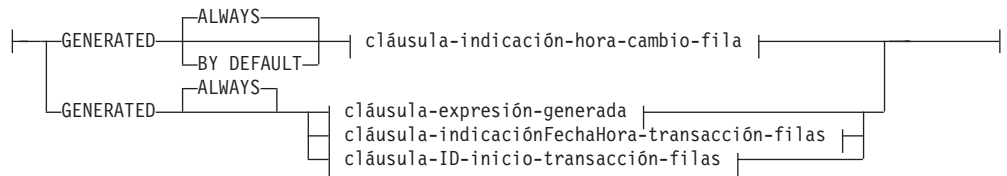


cláusula-por-omisión:

ALTER TABLE



cláusula-generada:



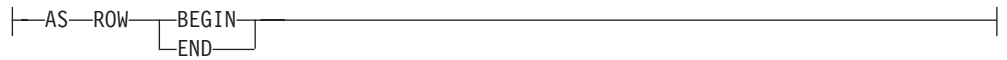
cláusula-indicación-hora-cambio-fila:



cláusula-expresión-generada:



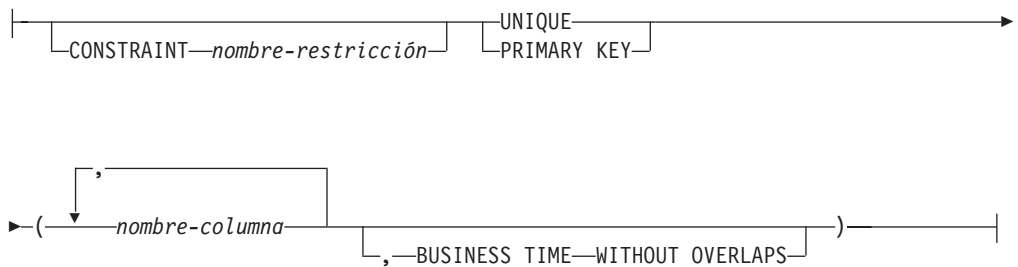
cláusula-indicaciónFechaHora-transacción-filas:

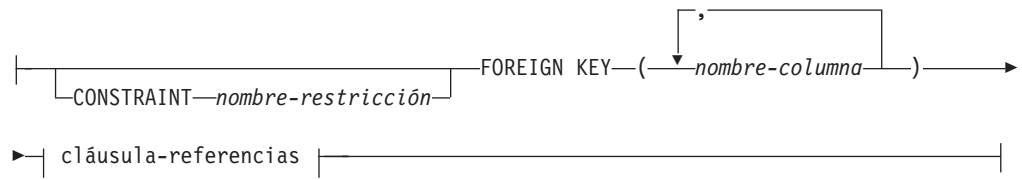
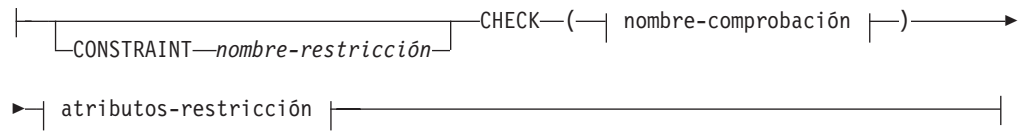
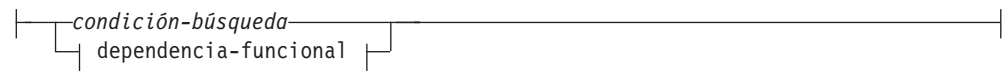
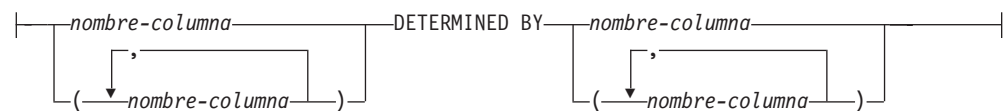
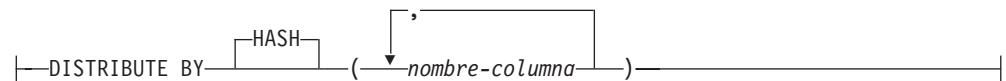
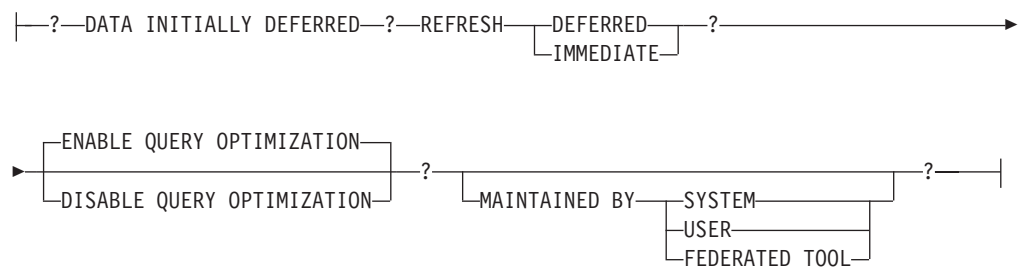


cláusula-ID-inicio-transacción-filas:



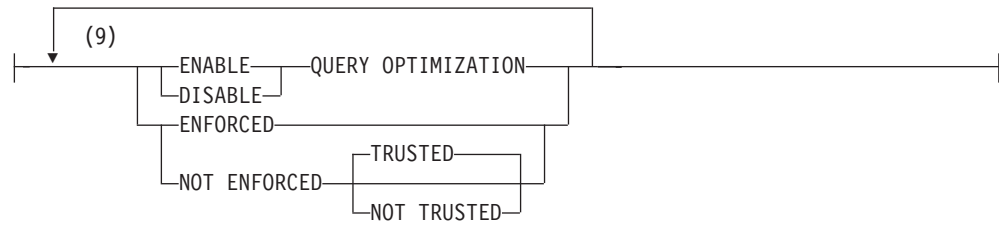
restricción-unicidad:



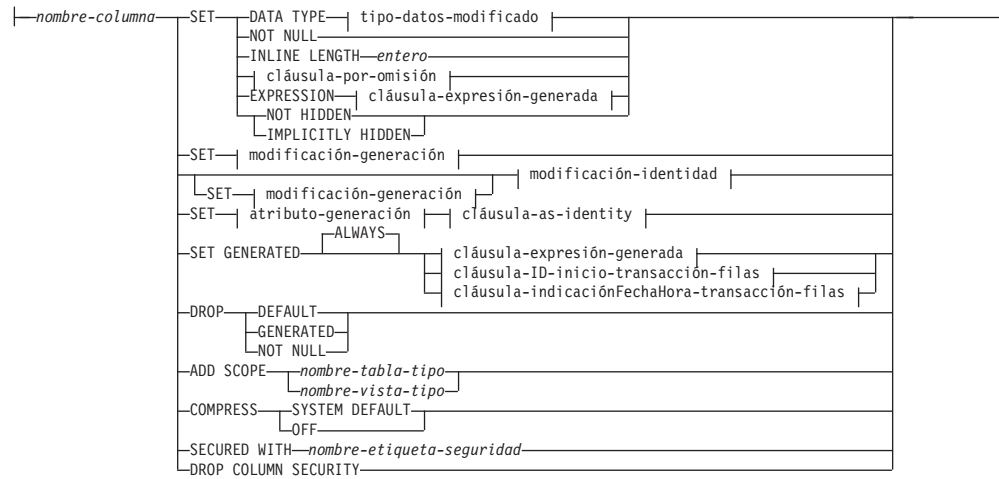
restricción-referencia:**restricción-comprobación:****condición-error:****dependencia-funcional:****cláusula-distribución:****definición-consulta-materializada:****opciones-tabla-renovable:**

ALTER TABLE

modificación-restricción:



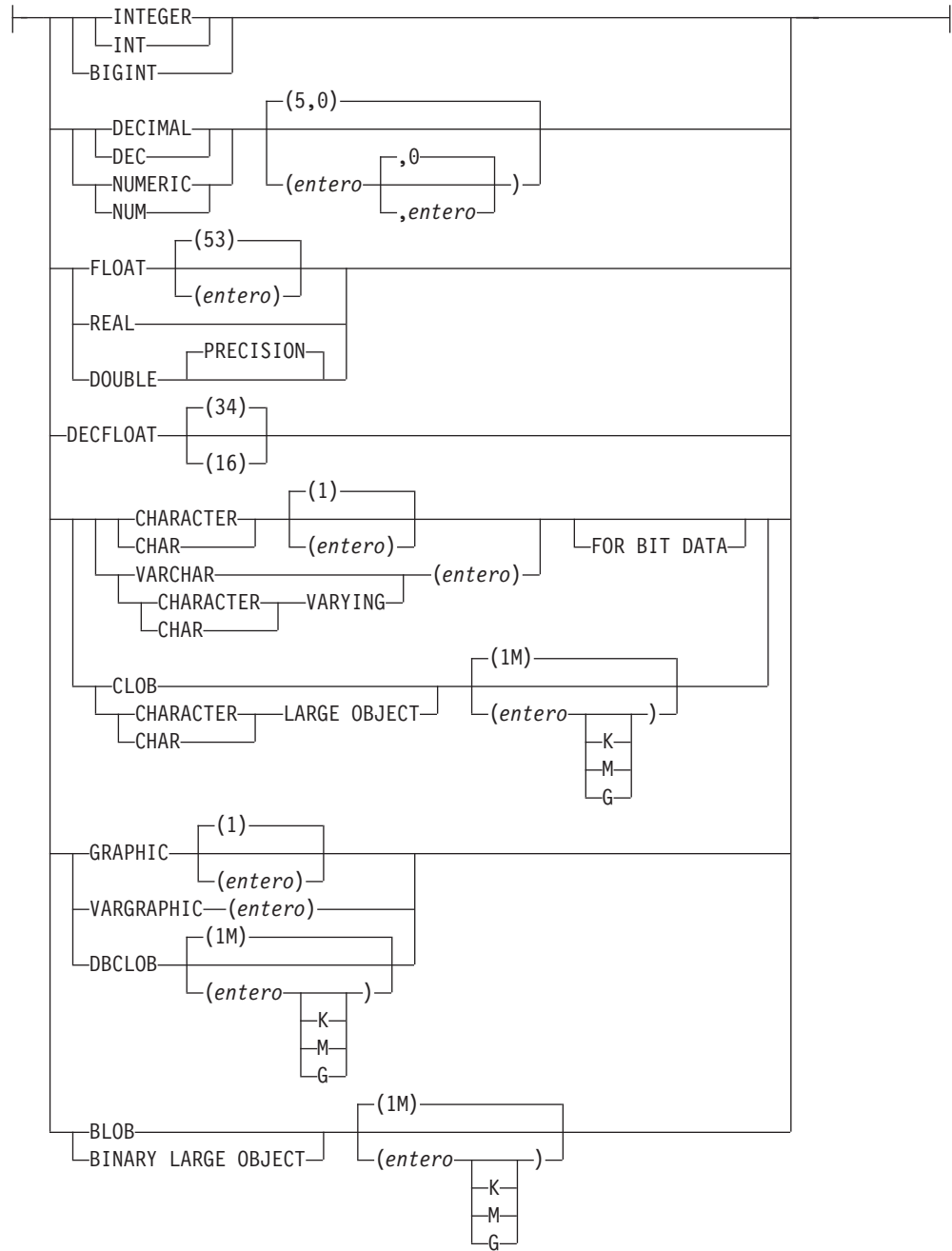
modificación-columna:



tipo-datos-modificado:

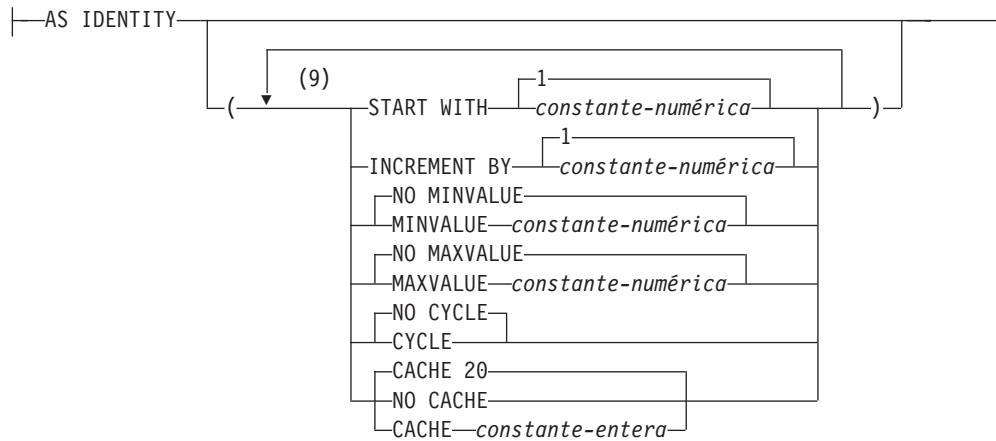


tipo-incorporado:

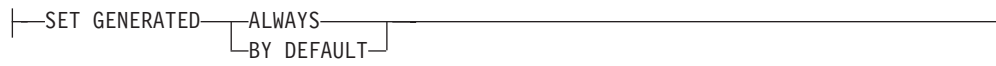


cláusula-as-identity:

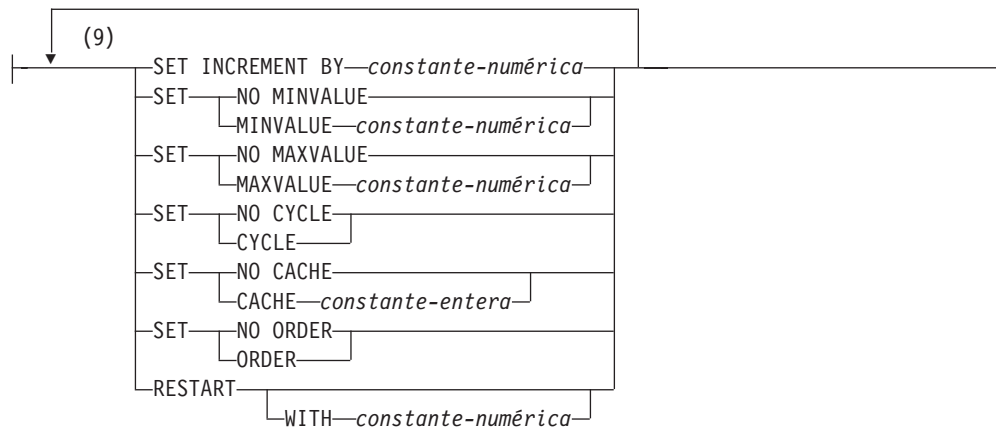
ALTER TABLE



modificación-generación:



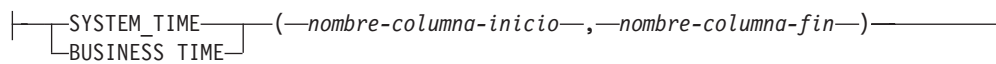
modificación-identidad:



atributo-generación:



definición-período:



Notas:

- 1 No debe especificarse la misma cláusula más de una vez (SQLSTATE 42614).
- 2 Si se especifica una cláusula `ACTIVATE` o `DEACTIVATE` para el control de

- acceso a filas, no se puede especificar ningún otro control de acceso a columnas, salvo ACTIVATE o DEACTIVATE, en la misma sentencia ALTER TABLE (SQLSTATE 42613).
- 3 Si se especifica una cláusula ACTIVATE o DEACTIVATE para el control de acceso a columnas, no se puede especificar ningún otro control de acceso a filas, salvo ACTIVATE o DEACTIVATE, en la misma sentencia ALTER TABLE (SQLSTATE 42613).
 - 4 Si la primera opción de columna seleccionada es *cláusula-generada*, se puede omitir *tipo-datos*; la expresión de generación calculará este valor.
 - 5 La cláusula *opciones-lob* sólo se aplica a los tipos de objeto grande (CLOB, DBCLOB y BLOB) y a los tipos diferenciados que se basan en tipos de objeto grande.
 - 6 La cláusula SCOPE sólo se aplica al tipo REF.
 - 7 La cláusula por omisión y la cláusula generada no se pueden especificar para la misma definición de columna (SQLSTATE 42614).
 - 8 El tipo de datos es opcional para una columna de indicación de fecha y hora de cambio de fila si la primera opción de columna especificada es una cláusula generada; el tipo de datos por omisión es TIMESTAMP(6). El tipo de datos es opcional para las columnas de inicio de fila, de fin de fila y de ID de inicio de transacción, si la primera opción de columna es una cláusula generada; el tipo de datos por omisión es TIMESTAMP(12)
 - 9 Una misma cláusula no se debe especificar más de una vez.
 - 10 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2).

Descripción

nombre-tabla

El *nombre-tabla* debe identificar una tabla que exista en el servidor actual. No puede ser un apodo (SQLSTATE 42809) y no debe ser una vista, una tabla de catálogo, una tabla temporal creada ni una tabla temporal declarada (SQLSTATE 42995).

Si *nombre-tabla* identifica una tabla de consulta materializada, las modificaciones se limitan a la adición o el descarte de la consulta materializada, la invocación de la cláusula ACTIVATING NOT LOGGED INITIALLY, la adición o el descarte de RESTRICT ON DROP, la modificación de DATA CAPTURE, PCTFREE, LOCKSIZE, APPEND, VOLATILE, la compresión de filas de datos, la compresión de valores y la activación o desactivación del control de acceso a filas y columnas.

Si el *nombre-tabla* identifica una tabla agrupada por clústeres de rangos, las modificaciones se limitan a añadir, cambiar o descartar restricciones, a activar NOT LOGGED INITIALLY, a añadir o descartar RESTRICT ON DROP, a cambiar el tamaño de bloqueo, a capturar datos o volátil y a establecer los valores por omisión de columna.

ADD *definición-columna*

Añade una columna a la tabla. La tabla no debe ser una tabla histórica para una tabla temporal de período de sistema (SQLSTATE 428HZ) o una tabla con tipo (SQLSTATE 428DH). Para todas las filas existentes en la tabla, el valor de

ALTER TABLE

la nueva columna se establece en su valor por omisión. La nueva columna es la última columna de la tabla; es decir, si al principio hay n columnas, la columna añadida es la columna $n+1$.

La adición de la nueva columna no debe dar lugar a que el número total de bytes de todas las columnas exceda el tamaño de registro máximo.

Si la tabla es una tabla temporal de período de sistema, la columna también se añade a la tabla histórica asociada.

Si la columna añadida es una columna generada que se basa en una expresión, la expresión no debe hacer referencia a una columna para la que se define una máscara de columna (SQLSTATE 42621).

Si se añade una columna a una tabla en la que se define una máscara o permiso, o a una tabla a la que se hace referencia en la definición de una máscara o permiso, dicha máscara o permiso queda invalidado. El acceso a una tabla que tenga activado el control de acceso a columnas y tenga definida una máscara no válida queda bloqueado hasta que la máscara no válida se inhabilite, descarte o vuelva a crear (SQLSTATE 560D0). El acceso a una tabla que tenga activado el control de acceso a filas y tenga definido un permiso de fila no válido queda bloqueado hasta que el permiso no válido se inhabilite, descarte o vuelva a crear (SQLSTATE 560D0).

nombre-columna

Es el nombre de la columna que se va a añadir a la tabla. El nombre no puede estar calificado. No pueden utilizarse nombres de columna o nombres de período existentes en la tabla (SQLSTATE 42711).

tipo-datos

Es uno de los tipos de datos que se listan en "CREATE TABLE".

NOT NULL

Evita que la columna contenga valores nulos. También debe especificarse la *cláusula-por-omisión* (SQLSTATE 42601).

opciones-lob

Especifica opciones para los tipos de datos LOB. Consulte *opciones-lob* en "CREATE TABLE".

SCOPE

Especifique un ámbito para una columna de tipo de referencia.

nombre2-tabla-con-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-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*.

nombre2-vista-con-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 *nombre2-vista-con-tipo* (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 ha especificado el nombre de restricción, el sistema genera un identificador exclusivo de 18 bytes dentro de los identificadores de las restricciones existentes definidas en la tabla. (El identificador se compone de la palabra "SQL" seguida de una secuencia de 15 caracteres numéricos que se generan mediante una función basada en la indicación de fecha y hora).

Si se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* se puede utilizar como nombre de un índice creado para dar soporte a la restricción. Consulte el apartado Notas 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 PRIMARY KEY en la descripción de *restricción-unicidad*.

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 UNIQUE en la descripción de *restricción-unicidad*.

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 *cláusula-referencias* en "CREATE TABLE".

CHECK (*condición-comprobación*)

Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Consulte *condición-error* en "CREATE TABLE".

cláusula-por-omisión

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 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 13 en la página 136. Si una columna se define como un tipo XML o estructurado, no se puede especificar una cláusula DEFAULT.

ALTER TABLE

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 13. 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 fecha y 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, indicación de fecha y 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 normas 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
- ser una constante numérica o un valor especial de coma flotante si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y después se convierten a coma flotante decimal. Para las columnas DECFLOAT(16), las constantes decimales deben tener una precisión igual o inferior a 16.
- no tener dígitos que no sean cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal (por ejemplo, 1.234 no puede ser el valor por omisión de una columna DECIMAL(5,2)).
- estar expresada con un máximo de 254 bytes, 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-fecha-hora

Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de 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 actual, la hora actual o la indicación de fecha y hora actual al procesarse la sentencia ALTER TABLE.

registro-especial-usuario

Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER. Para las filas existentes, el valor es CURRENT USER, SESSION_USER o SYSTEM_USER de la sentencia ALTER TABLE.

CURRENT SCHEMA

Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud superior o igual al atributo de longitud del registro especial CURRENT SCHEMA. Para las filas existentes, el valor del registro especial de CURRENT SCHEMA al procesarse la sentencia ALTER TABLE.

CURRENT MEMBER

Especifica el valor que tenía el registro especial CURRENT MEMBER en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT MEMBER, el tipo de datos de la columna debe permitir la asignación desde un entero. Para las filas existentes, el valor del registro especial CURRENT MEMBER al procesarse 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.

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 esta debe coincidir con el nombre del tipo de 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 indicación 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 normas de una constante para el tipo de 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 serie de caracteres.

registro-especial-fecha-hora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario

Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de 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.

CURRENT SCHEMA

Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()

Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

cláusula-generada

Especifica un valor generado para la columna. Esta cláusula no debe especificarse con *cláusula-por-omisión* en una definición de columna (SQLSTATE 42623). Una columna generada no se puede añadir a una tabla temporal de período de sistema (SQLSTATE 428HZ). Para ver detalles sobre la generación de columnas, consulte "CREATE TABLE".

GENERATED

Especifica que el gestor de bases de datos genere valores para la columna. Debe especificarse GENERATED si la columna se tiene que considerar como columna de identidad, columna de indicación de

fecha y hora de cambio de fila, columna de inicio de fila, columna de fin de fila, columna de ID de inicio de transacción o columna de expresión generada.

Si la columna puede contener nulos, el valor nulo se asigna como valor para la columna en las filas existentes. De lo contrario, el valor de la columna en las filas existentes depende de la definición de la columna:

- ROW CHANGE TIMESTAMP utiliza un valor que se corresponde con la indicación de fecha y hora de la sentencia ALTER TABLE
- ROW BEGIN utiliza una fecha que corresponde al 1 de enero, 0001 y una hora que corresponde a las 0 horas, 0 minutos, 0 segundos y 0 fracciones de segundo
- ROW END utiliza una fecha que corresponde al 30 de diciembre, 9999 y una hora que corresponde a las 0 horas, 0 minutos, 0 segundos y 0 fracciones de segundo
- TRANSACTION START ID utiliza una fecha que corresponde al 1 de enero, 0001 y una hora que corresponde a las 0 horas, 0 minutos, 0 segundos y 0 fracciones de segundo
- Las expresiones utilizan el valor que deriva de la expresión

ALWAYS

Especifica que el gestor de bases de datos genere siempre un valor para la columna cuando se inserta o se actualiza una fila y se debe generar un valor. El resultado de la expresión se almacena en la tabla. GENERATED ALWAYS es la opción recomendada, a menos que estén realizándose operaciones de propagación o de descarga y de recarga. GENERATED ALWAYS es el valor por omisión de las columnas generadas.

BY DEFAULT

Especifica que el gestor de bases de datos genere un valor para la columna cuando se inserte una fila en la tabla, o cuando se actualice especificando DEFAULT para la columna, a menos que se especifique un valor explícito. BY DEFAULT sólo se puede especificar con *cláusula-indicación-hora-cambio-fila*. BY DEFAULT es la opción recomendada cuando se utiliza la propagación de datos o cuando se realizan operaciones de descarga y recarga.

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

Especifica que la columna es una columna de indicación de fecha y hora con valores generados por el gestor de bases de datos. Se genera un valor para la columna en cada fila que se inserta y para cualquier fila de cualquier columna que se actualiza. El valor que se genera para una columna ROW CHANGE TIMESTAMP es una indicación de fecha y hora que corresponde a la hora de inserción o actualización de la fila. Si se insertan o actualizan varias filas con una sentencia única, el valor de la columna ROW CHANGE TIMESTAMP puede ser diferente para cada fila.

Una tabla sólo puede tener una columna ROW CHANGE TIMESTAMP (SQLSTATE 428C1). Si se especifica *tipo-datos*, debe ser TIMESTAMP o TIMESTAMP(6) (SQLSTATE 42842). Una columna ROW CHANGE TIMESTAMP no puede tener una cláusula DEFAULT (SQLSTATE 42623). Debe especificarse NOT NULL para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42831).

ALTER TABLE

AS (*expresión-generación*)

Especifica que la definición de la columna se basa en una expresión. Requiere que la tabla se coloque en estado sin acceso pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF NO ACCESS. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY con las opciones IMMEDIATE CHECKED y FORCE GENERATED para actualizar y comprobar todos los valores de esa columna con la nueva expresión. Para ver los detalles acerca de la especificación de una columna con una *expresión-generación*, consulte "CREATE TABLE".

AS ROW BEGIN

Especifica que el gestor de bases de datos asigna el valor siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El valor se genera mediante una lectura de la hora del reloj durante la ejecución del primer suceso de los sucesos de la transacción siguientes:

- Una sentencia de cambio de datos que exija que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla
- La eliminación de una fila en una tabla temporal de período de sistema

En el caso de una tabla temporal de período de sistema, el gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de inicio de fila en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas insertadas en una tabla histórica asociada tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio (SQLSTATE 01695). Ello puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema. El parámetro de configuración de base de datos **system_time_period_adj** debe establecerse en Yes para que tenga lugar este ajuste del valor de indicación de fecha y hora; de lo contrario, se devuelve un error (SQLSTATE 57062). Si se insertan o actualizan varias filas en una sola transacción de SQL y no es necesario un ajuste, los valores de la columna de inicio de fila son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción. Se necesita una columna de inicio de fila como columna de inicio de un período SYSTEM_TIME, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de inicio de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12) (SQLSTATE 42842). La columna debe definirse como NOT NULL (SQLSTATE 42831). Las columnas de inicio de fila no se pueden actualizar.

AS ROW END

Especifica que el gestor de bases de datos asigna el valor máximo del tipo de datos de la columna siempre que se inserta una fila o se actualiza una columna de la fila.

Se necesita una columna de fin de fila como segunda columna de un período `SYSTEM_TIME`, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de fin de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como `TIMESTAMP(12)`. Si se especifica *tipo-datos*, debe ser `TIMESTAMP(12)` (SQLSTATE 42842). La columna debe definirse como `NOT NULL` (SQLSTATE 42831). Las columnas de fin de fila no se pueden actualizar.

AS TRANSACTION START ID

Especifica que el gestor de bases de datos asigna el valor siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El gestor de bases de datos asigna un valor de indicación de fecha y hora exclusivo por transacción o el valor nulo. El valor nulo se asigna a la columna de ID de inicio de transacción si la columna puede contener nulos y si existe una columna de inicio de fila en la tabla en la que el valor no tiene que ajustarse. En caso contrario, el valor se genera mediante una lectura de la hora del reloj durante la ejecución del primer suceso de los sucesos de la transacción siguientes:

- Una sentencia de cambio de datos que exija que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla
- La eliminación de una fila en una tabla temporal de período de sistema

Si se insertan o actualizan varias filas en una sola transacción de SQL, los valores de la columna de ID de inicio de transacción son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.

Se necesita una columna de ID de inicio de transacción para una tabla temporal de período de sistema, lo cual es el uso esperado para este tipo de columna generada.

Una tabla sólo puede tener una columna de ID de inicio de transacción (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como `TIMESTAMP(12)`. Si se especifica *tipo-datos*, debe ser `TIMESTAMP(12)`. Las columnas de ID de inicio de transacción no se pueden actualizar.

COMPRESS SYSTEM DEFAULT

Especifica que los valores por omisión del sistema (es decir, los valores por omisión que se utilizan para los tipos de datos cuando no se ha especificado ningún valor específico) van a almacenarse utilizando el espacio mínimo. Si no se especifica la cláusula `VALUE COMPRESSION`, se devuelve un mensaje de aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el espacio mínimo.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

El tipo de datos base no puede ser `DATE`, `TIME`, `TIMESTAMP`, `XML` ni un tipo de datos estructurado (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto.

ALTER TABLE

Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

COLUMN SECURED WITH *nombre-etiqueta-seguridad*

Identifica una etiqueta de seguridad existente para la política de seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064). La tabla no debe ser una tabla temporal de período de sistema.

NOT HIDDEN o IMPLICITLY HIDDEN

Especifica si la columna ha de definirse como oculta. El atributo oculto determina si la columna se incluye como referencia implícita a la tabla o si se le puede hacer referencia de manera explícita en sentencias de SQL. El valor por omisión es NOT HIDDEN.

NOT HIDDEN

Especifica que la columna se incluye como referencia implícita a la tabla y que se puede hacer referencia a la columna de manera explícita.

IMPLICITLY HIDDEN

Especifica que la columna no es visible en sentencias de SQL a menos que se haga referencia a la columna de manera explícita por el nombre. Por ejemplo, suponiendo que una tabla incluye una columna definida con la cláusula IMPLICITLY HIDDEN, el resultado de SELECT * no incluye la columna oculta implícitamente. Sin embargo, el resultado de SELECT que haga referencia de manera explícita al nombre de una columna implícitamente oculta incluirá dicha columna en la tala de resultados.

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, consulte *nombre-restricción* en "CREATE TABLE".

UNIQUE (*nombre-columna*, ... **BUSINESS_TIME WITHOUT OVERLAPS**)

Define una clave exclusiva compuesta por las columnas y los períodos identificados. 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 más dos veces el número de períodos identificados no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XML". No se puede utilizar como parte de una clave exclusiva ningún LOB, tipo diferenciado basado en ninguno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008). El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). Si LANGLEVEL es SQL92E o MIA, se

devuelve un error, SQLSTATE 42891. 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 exclusiva (omitiendo 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). No obstante, para las tablas particionadas, los índices particionados no únicos cuyas columnas no son un superconjunto de columnas clave de particionamiento de tabla no se consideran índices coincidentes.

Cuando una partición está enlazada a una tabla temporal de período de aplicación particionada por rangos que tiene un índice BUSINESS_TIME WITHOUT OVERLAPS particionado, la tabla fuente debe tener un índice que coincida con el índice BUSINESS_TIME WITHOUT OVERLAPS particionado. Además, los atributos PERIODNAME y PERIODPOLICY en los índices también deben coincidir.

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. Si hay varios índices únicos, la selección es arbitraria con una excepción:

- Para las tablas particionadas, los índices particionados únicos coincidentes tienen prioridad respecto a los índices no particionados únicos coincidentes o índices no únicos coincidentes (particionados o no particionados).

Si no se encuentra ningún índice coincidente, se creará automáticamente un índice bidireccional exclusivo para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado Notas para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

BUSINESS_TIME WITHOUT OVERLAPS

En el caso de una restricción, BUSINESS_TIME indica el nombre del período en esta tabla. El período debe existir en la tabla (SQLSTATE 42727).

BUSINESS_TIME WITHOUT OVERLAPS especifica que no se permiten los períodos que se solapan para BUSINESS_TIME y que los valores para el resto de las claves deben ser exclusivos con respecto a cualquier período de BUSINESS_TIME. Cuando se especifica BUSINESS_TIME WITHOUT OVERLAPS, la columna final y la columna inicial del período BUSINESS_TIME (en este orden de las columnas) se añadirán automáticamente a la clave de índice en orden ascendente, y se impone que no haya solapamientos en el tiempo. Las columnas que se utilizan para definir BUSINESS_TIME no deben especificarse como parte de la restricción (SQLSTATE 428HW).

PRIMARY KEY (nombre-columna, ... BUSINESS_TIME WITHOUT OVERLAPS)

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 ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte “Número total de bytes” en

ALTER TABLE

“CREATE TABLE”. Para los límites de longitud de clave, consulte “Límites de SQL”. La tabla no debe tener una clave primaria y las columnas identificadas deben definirse como NOT NULL. No se puede utilizar como parte de una clave primaria ningún LOB, tipo diferenciado basado en ninguno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008). El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.) Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515). *nombre-columna* no debe ser el nombre de una columna de indicación de fecha y hora de cambio de fila o una columna de inicio o fin del período (SQLSTATE 428HW).

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). No obstante, para las tablas particionadas, los índices particionados no únicos cuyas columnas no son un superconjunto de columnas clave de particionamiento de tabla no se consideran índices coincidentes.

Cuando una partición está enlazada a una tabla temporal de período de aplicación particionada por rangos que tiene un índice BUSINESS_TIME WITHOUT OVERLAPS particionado, la tabla fuente debe tener un índice que coincida con el índice BUSINESS_TIME WITHOUT OVERLAPS particionado. Además, los atributos PERIODNAME y PERIODPOLICY en los índices también deben coincidir.

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. Si hay varios índices únicos, la selección es arbitraria con una excepción:

- Para las tablas particionadas, los índices particionados únicos coincidentes tienen prioridad respecto a los índices no particionados únicos coincidentes o índices no únicos coincidentes (particionados o no particionados).

Si no se encuentra ningún índice coincidente, se creará automáticamente un índice bidireccional exclusivo para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado Notas para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

En una tabla sólo se puede definir una clave primaria.

BUSINESS_TIME WITHOUT OVERLAPS

En el caso de una restricción, BUSINESS_TIME indica el nombre del período en esta tabla. El período debe existir en la tabla (SQLSTATE 42727).

BUSINESS_TIME WITHOUT OVERLAPS especifica que no se permiten los períodos que se solapan para BUSINESS_TIME y que los valores para el resto de las claves deben ser exclusivos con respecto a cualquier período de BUSINESS_TIME. Cuando se especifica BUSINESS_TIME WITHOUT OVERLAPS, la columna final y la

columna inicial del período BUSINESS_TIME (en este orden de las columnas) se añadirán automáticamente a la clave de índice en orden ascendente, y se impone que no haya solapamientos en el tiempo. Las columnas que se utilizan para definir BUSINESS_TIME no deben especificarse como parte de la restricción (SQLSTATE 428HW).

ADD restricción-referencial

Define una restricción de referencia. Consulte *restricción-referencia* en “CREATE TABLE”.

ADD restricción-comprobación

Define una restricción de comprobación o una dependencia funcional. Consulte *restricción-comprobación* en “CREATE TABLE”.

ADD cláusula-distribución

Define una clave de distribución. La tabla debe definirse en un espacio de tablas de un grupo de particiones de base de datos de partición única (SQLSTATE 55037) y no puede tener ya una clave de distribución (SQLSTATE 42889). Si ya existe una clave de distribución para la tabla, la clave existente debe descartarse antes de añadir la clave de distribución nueva. No puede añadirse una clave de distribución a una tabla que sea una subtabla (SQLSTATE 428DH).

DISTRIBUTE BY HASH (nombre-columna...)

Define una clave de distribución 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. No se puede utilizar como parte de una clave de distribución ninguna columna si el tipo de datos de la columna es BLOB, CLOB, DBCLOB, XML, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado.

ADD RESTRICT ON DROP

Especifica que la tabla no puede descartarse y que el espacio de tablas que contiene la tabla no puede descartarse.

ADD MATERIALIZED QUERY

definición-consulta-materializada

Cambia una tabla normal por una tabla de consulta materializada para utilizarla durante la optimización de la consulta. La tabla especificada por *nombre-tabla*:

- No debe haberse definido anteriormente como una tabla de consulta materializada.
- No debe ser una tabla con tipo.
- No debe tener ninguna restricción, índice exclusivo ni activador definido.
- No debe hacer referencia a un apodo marcado con el almacenamiento en antememoria inhabilitado.
- No debe existir una referencia a esta en la definición de otra tabla de consulta materializada.
- No debe existir una referencia a esta en la definición de una vista habilitada para la optimización de consulta.

Si *nombre-tabla* no cumple estos criterios, se devolverá un error (SQLSTATE 428EW).

ALTER TABLE

Si el control de acceso de nivel de fila o de nivel de columna se activa para alguna tabla a la que se haga referencia directa o indirectamente en la *selecciónCompleta* de *definición-consulta-materializada* y el control de acceso de nivel de fila no está activado para la tabla que se va a modificar, el control de acceso de nivel de fila se activa de forma explícita para la tabla modificada. Esto limita el acceso directo al contenido de la tabla de consulta materializada. Una consulta que hace referencia explícita a la tabla antes de que se defina ese permiso de fila devuelve un aviso de que no existen datos en la tabla (SQLSTATE 02000). Para proporcionar acceso a la tabla de consulta materializada, se puede crear un permiso de fila adecuado o se puede emitir un mandato **ALTER TABLE DEACTIVATE ROW ACCESS CONTROL** en la tabla de consulta materializada para eliminar la protección de nivel de fila si ello es adecuado.

Si la tabla de consulta materializada hace referencia a cualquier tabla con el control de acceso de nivel de fila o columna activado, las funciones a las que se hace referencia en la *selección-completa* de *definición-consulta-materializada* deben definirse con el atributo SECURED (SQLSTATE 428EC).

Si la tabla que se está modificando para convertirla en tabla de consulta materializada tiene permisos (excluido el permiso por omisión generado por el sistema) o máscaras definidas, se produce un error de ALTER (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 información detallada acerca de la especificación de la *selección completa* para una tabla de consulta materializada, consulte "CREATE TABLE". Una restricción adicional es que *nombre-tabla* no puede estar referenciada, directa o indirectamente, en la selección completa.

opciones-tabla-renovable

Especifica las opciones que pueden renovarse para modificar una tabla de consulta materializada.

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 en el momento en que se procesa la sentencia REFRESH TABLE. Las tablas de consulta materializada que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

IMMEDIATE

Los cambios que se han realizado en las tablas subyacentes como parte de una sentencia DELETE, INSERT o UPDATE se aplican en cascada a la tabla de consulta materializada. En este caso, el contenido de la tabla, en cualquier momento, es igual como si se procesara la subselección especificada. Las tablas de consulta materializada que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

Las tablas de consulta materializada pueden utilizarse para la optimización de la consulta.

DISABLE QUERY OPTIMIZATION

La tabla de consulta materializada no se utilizará para la optimización de la consulta. La tabla todavía puede consultarse directamente.

MAINTAINED BY

Especifica quién mantiene los datos de la tabla de consulta materializada: el sistema, el usuario o una herramienta de duplicación.

SYSTEM

Especifica que el sistema mantiene los datos de la tabla de consulta materializada.

USER

Especifica que el usuario mantiene los datos de la tabla de consulta materializada. El usuario puede realizar operaciones de actualización, supresión o inserción en las tablas de consulta materializada mantenidas por el usuario. La sentencia REFRESH TABLE, que se utiliza para las tablas de consulta materializada mantenidas por el sistema, no puede invocarse para las tablas de consulta materializada mantenidas por el usuario. Sólo una tabla de consulta materializada REFRESH DEFERRED puede definirse como MAINTAINED BY USER.

FEDERATED_TOOL

Especifica que la herramienta de duplicación mantiene los datos de la tabla de consulta materializada. La sentencia REFRESH TABLE, que se utiliza para las tablas de consulta materializada mantenidas por el sistema, no se puede invocar para las tablas de consulta materializada mantenidas por herramientas federadas. Sólo se puede definir una única tabla de consulta materializada REFRESH DEFERRED como MAINTAINED BY FEDERATED_TOOL.

ALTER FOREIGN KEY *nombre-restricción*

Altera los atributos de restricción de la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia existente (SQLSTATE 42704).

ALTER CHECK *nombre-restricción*

Altera los atributos de restricción de la restricción de comprobación o dependencia funcional *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente o una dependencia funcional (SQLSTATE 42704).

ALTER TABLE

modificación-restricción

Opciones para cambiar los atributos que se asocian a las restricciones de referencia o de comprobación.

ENABLE QUERY OPTIMIZATION o DISABLE QUERY OPTIMIZATION

Especifica si se puede utilizar la restricción o la dependencia funcional para la optimización de la consulta bajo circunstancias adecuadas.

ENABLE QUERY OPTIMIZATION

La restricción se supone que es verdadera y se puede utilizar para la optimización de consulta.

DISABLE QUERY OPTIMIZATION

No se puede utilizar la restricción para la optimización de consulta.

ENFORCED o NOT ENFORCED

Especifica si el gestor de bases de datos obliga a aplicar la restricción durante operaciones normales como, por ejemplo, insertar, actualizar o suprimir.

ENFORCED

Cambie la restricción por ENFORCED. No se puede especificar ENFORCED para una dependencia funcional (SQLSTATE 42621).

NOT ENFORCED

Cambie la restricción por NOT ENFORCED.

TRUSTED

Se puede confiar en los datos para que se ajusten a la restricción. TRUSTED sólo debe utilizarse si se reconoce de forma independiente que los datos de la tabla se ajustan a la restricción. Los resultados de la consulta pueden ser imprevisibles si los datos no se ajustan realmente a la restricción. Es la opción por omisión.

Las restricciones informativas no deben incumplirse en ningún momento. Las restricciones informativas se utilizan en la optimización de consultas y en el proceso incremental de las MQT REFRESH IMMEDIATE y las tablas de etapas. Estos procesos podrían generar resultados imprevisibles o contenido de tablas de etapas o MQT incorrecto si se incumplen las restricciones. Por ejemplo, el orden en el que se mantienen las tablas padre-hijo es importante. Si se quieren añadir filas a una tabla padre-hijo, primero debe insertar las filas en la tabla padre. Para eliminar filas de una tabla padre-hijo, primero debe suprimir las filas de la tabla hijo. De esta forma se garantiza que no queden filas huérfanas en la tabla hijo en ningún momento. Si se incumplen las restricciones informativas, el mantenimiento incremental de los datos de tablas de etapas y de los datos de MQT dependientes podría optimizarse en función de las restricciones informativas incumplidas, lo que produciría datos incorrectos.

NOT TRUSTED

No se puede confiar en los datos para que se ajusten a la restricción. NOT TRUSTED está pensado para casos en los que los datos se ajustan a la restricción para la mayor parte de las filas, pero no se reconoce de forma independiente que todas las filas o las futuras adiciones se ajustarán a la restricción. Si una restricción es NOT TRUSTED y se ha habilitado para la optimización de consultas, no se utilizará para realizar optimizaciones que dependan de los datos que se ajusten por completo a la restricción.

Sólo se puede especificar NOT TRUSTED para las restricciones de integridad referenciales (SQLSTATE 42613).

ALTER *modificación-columna*

Modifica la definición de una columna. Sólo se modificarán los atributos especificados; los demás permanecerán sin modificar. No se pueden alterar las columnas de una tabla con tipo (SQLSTATE 428DH). La tabla no debe definirse como tabla histórica (SQLSTATE 428FR).

nombre-columna

Especifica el nombre de la columna que se debe modificar. El *nombre-columna* debe identificar una columna existente de la tabla (SQLSTATE 42703). El nombre no puede estar calificado. El nombre no puede identificar una columna que de algún otro modo se añade, descarta o modifica en la misma sentencia ALTER TABLE (SQLSTATE 42711).

SET DATA TYPE *tipo-datos-modificado*

Especifica el nuevo tipo de datos de la columna. El nuevo tipo de datos debe poder convertirse a partir del tipo de datos existente de la columna (SQLSTATE 42837), excepto cuando uno de los tipos de datos es un tipo diferenciado, en cuyo caso el tipo de datos fuente del tipo diferenciado se utiliza al determinar si los tipos de datos admiten conversión.

No está permitida la alteración de un tipo de datos de serie que da como resultado el truncamiento de los caracteres no blancos de los datos existentes (SQLSTATE 42837).

Las alteraciones del tipo de datos requieren una reorganización de tabla para que pueda accederse completamente a la tabla (SQLSTATE 57016), excepto en las situaciones siguientes:

- Incremento de la longitud de una columna VARCHAR o VARGRAPHIC
- Reducción de la longitud de una columna VARCHAR o VARGRAPHIC sin truncar los blancos de cola de los datos existentes

Se puede invocar la rutina administrativa SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS para efectuar la reorganización de la tabla según sea necesario. Una alteración de tipo de datos que requiere una reorganización de tabla no puede especificarse si la tabla se encuentra en estado SET INTEGRITY PENDING (SQLSTATE 57007).

Un tipo de datos de serie no puede modificarse si la columna es una columna de clave de particionamiento de tabla.

Si se trata de una columna de una clave de distribución, el nuevo tipo de datos debe cumplir con los requisitos siguientes (SQLSTATE 42997):

- Debe ser del mismo tipo de datos que el tipo de columna actual.
- Debe tener la misma longitud que el tipo de columna actual, excepto en el caso de aumento de longitud de las columnas de los tipos de datos VARCHAR y VARGRAPHIC
- No se pueden modificar a FOR BIT DATA ni viceversa en los casos de tipos de datos CHAR y VARCHAR

Si el tipo de datos es LOB, la longitud especificada no puede ser inferior a la longitud existente (SQLSTATE 42837).

El tipo de datos de una columna de identidad no se puede modificar (SQLSTATE 42997).

ALTER TABLE

El tipo de datos de una columna definida como ROW BEGIN, ROW END o TRANSACTION START ID no se puede alterar (SQLSTATE 428FR).

El tipo de datos y la capacidad de nulos de las columnas de período BUSINESS_TIME no se pueden alterar (SQLSTATE 428FR).

La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997).

El tipo de datos de una columna no se puede modificar si se aplica alguna de las condiciones siguientes (SQLSTATE 42893):

- La columna es una columna de expresión generada y los datos de la columna de expresión generada cambiarán si se altera la columna.
- Se hace referencia a la columna en una expresión de una columna de expresión generada y los datos de la columna de expresión generada cambiarán si se altera la columna.
- Se hace referencia a la columna en una restricción de comprobación y la restricción de comprobación no se cumplirá si se altera la columna.
- La columna se utiliza en una restricción de integridad referencial y la restricción de integridad referencial no se cumplirá si se altera la columna.

Al modificar una columna, el número total de bytes de todas las columnas no puede superar el tamaño de registro máximo (SQLSTATE 54010). Si se utiliza la columna en una restricción de unicidad o en un índice, la longitud nueva no puede hacer que la suma de las longitudes almacenadas para la restricción de unicidad sean superiores al límite de la longitud de la clave de índice para el tamaño de página (SQLSTATE 54008). Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XML".

Si **auto_reval** se establece en DISABLED, los efectos en cascada de la modificación de una columna se muestran en la Tabla 14.

Si un permiso de fila o una máscara de columna depende de la columna que se va a alterar (tal como se registra en la vista de catálogo SYSCAT.CONTROLDEP), se devuelve un error (SQLSTATE 42917).

Tabla 14. Efectos en cascada de alterar una columna

Operación	Efecto
Modificación de una columna a la que hace referencia una vista o restricción de comprobación	El objeto se vuelve a generar durante el proceso de modificación. En el caso de una vista, la resolución del método o la función para el objeto puede ser diferente tras la operación de modificación y cambiar la semántica del objeto. En el caso de una restricción de comprobación, si cambia la semántica del objeto como consecuencia de la operación de modificación, la operación falla.
Modificación de una columna de una tabla que tiene un paquete, activador o rutina de SQL dependiente	El objeto se marca como no válido y se vuelve a validar la próxima vez que se utiliza.

Tabla 14. Efectos en cascada de alterar una columna (continuación)

Operación	Efecto
Modificación del tipo de una columna de una tabla a la que hace referencia un XSROBJECT habilitado para la descomposición	El XSROBJECT se marca como inoperativo para la descomposición. Para volver a habilitar el XSROBJECT, puede que sea necesario volver a ajustar sus correlaciones; a continuación, emita una sentencia ALTER XSROBJECT ENABLE DECOMPOSITION para el XSROBJECT.
Modificación de una columna a la que se hace referencia en la expresión por omisión de una variable global	La expresión por omisión de la variable global se valida durante el proceso de modificación. Si no se puede resolver una función definida por el usuario en la expresión por omisión, la operación falla.

Si la tabla es una tabla temporal de período de sistema, la columna también cambia en cualquier tabla histórica asociada. Si la tabla es una tabla temporal de período del sistema, las columnas de tipo de datos de serie no se pueden alterar para que tengan una longitud que exija que los datos se trunquen y el tipo de datos numérico no se puede alterar por tipos de datos de menos precisión (SQLSTATE 42837).

SET NOT NULL

Especifica que la columna no puede contener valores nulos. Ningún valor de esta columna en las filas existentes de la tabla puede ser nulo (SQLSTATE 23502). Esta cláusula no está permitida si la columna se especifica en la clave foránea de una restricción de referencia con la norma DELETE establecida en SET NULL y no existe ninguna otra columna con posibilidad de nulos en la clave foránea (SQLSTATE 42831). Para modificar este atributo para una columna, es necesaria la reorganización de la tabla para poder acceder posteriormente a la tabla (SQLSTATE 57016). Tenga en cuenta que, dado que esta operación requiere la validación de los datos de la tabla, no se puede realizar si la tabla se encuentra en estado pendiente de reorganización (SQLSTATE 57016). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997).

Si existe un permiso de fila o una máscara de columna, que depende de la columna que se debe alterar, se emitirá un error (SQLSTATE 42917).

Si la tabla es una tabla temporal de período de sistema, la columna también cambia en cualquier tabla histórica asociada.

SET INLINE LENGTH *entero*

Cambia la longitud en línea de un tipo estructurado existente, XML o una columna de tipo de datos LOB. La longitud en línea indica el tamaño máximo en bytes de una instancia de un tipo estructurado, XML o tipo de datos LOB que se debe almacenar en la fila de la tabla base. Las instancias de un tipo estructurado o tipo de datos XML que no se pueden almacenar en línea en la fila de tabla base se almacenan por separado, de forma parecida al modo como se almacenan los valores de LOB.

El tipo de datos *nombre-columna* debe ser un tipo estructurado, XML o un tipo de datos LOB (SQLSTATE 42842).

La longitud en línea por omisión de una columna de tipo estructurado es la longitud en línea de su tipo de datos (especificada explícitamente o por

ALTER TABLE

omisión en la sentencia CREATE TYPE). Si la longitud en línea de un tipo estructurado es menor que 292, el valor 292 se utiliza para la longitud en línea de la columna.

El valor explícito de la longitud en línea sólo puede incrementarse (SQLSTATE 429B2); no puede exceder 32673 (SQLSTATE 54010). En el caso de una columna de tipo estructurado o de tipo de datos XML, debe ser, como mínimo, 292. En el caso de una columna de tipo de datos LOB, el valor de INLINE LENGTH no debe ser menor al tamaño máximo del descriptor de LOB.

La modificación de la columna no debe dar lugar a que el número total de bytes de todas las columnas supere el límite de tamaño de fila (SQLSTATE 54010).

Esta sentencia no establecerá en línea en la fila de tabla base los datos que ya se habían almacenado por separado del resto de la fila. Para utilizar la longitud en línea modificada de una columna de tipo estructurado, invoque el mandato REORG para la tabla especificada tras haber modificado la longitud en línea de su columna. Para utilizar la longitud en línea modificada de una columna de tipo de datos XML en una tabla existente, actualice todas las filas con una sentencia UPDATE. El mandato REORG no tiene efecto alguno en el almacenamiento de filas de los documentos XML. Para utilizar la longitud en línea modificada de una columna de tipo de datos LOB, utilice el mandato REORG con la opción LONGLOBDATA o ejecute UPDATE en la columna LOB correspondiente. Por ejemplo:

```
UPDATE nombre-tabla SET columna-lob = columna-lob
WHERE LENGTH(columna-lob) <= longitud-en-línea-seleccionada - 4
```

donde *nombre-tabla* es la tabla que tenía la longitud en línea de la columna de tipo de datos LOB modificada, *columna-lob* es la columna de tipo de datos LOB que se ha modificado y *longitud-en-línea-seleccionada* es el nuevo valor que se ha seleccionado para INLINE LENGTH.

Si existe un permiso de fila o una máscara de columna, que depende de la columna que se debe alterar, se devolverá un error (SQLSTATE 42917).

Si la tabla es una tabla temporal de período de sistema, los cambios de longitud en línea se propagan a la tabla histórica.

SET cláusula-por-omisión

Especifica un nuevo valor por omisión para la columna que se va a modificar. La columna no debe estar definida todavía como una columna generada (SQLSTATE 42623). El valor por omisión especificado debe representar un valor que pueda asignarse a la columna según las normas de asignación descritas en "Asignaciones y comparaciones". Al modificar el valor por omisión, no se cambia el valor asociado a esta columna para las filas existentes.

SET EXPRESSION AS (*expresión-generación*)

Cambia la expresión de la columna por la *expresión-generación* especificada. SET EXPRESSION requiere que la tabla se coloque en estado pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY con las opciones IMMEDIATE CHECKED y FORCE GENERATED para actualizar y comprobar todos los valores de esa columna con la nueva expresión. La columna ya debe estar definida como una columna generada basada en una expresión (SQLSTATE 42837) y no

puede haber aparecido en las cláusulas PARTITIONING KEY, DIMENSIONS o KEY SEQUENCE de la tabla (SQLSTATE 42997). La expresión de generación debe seguir las mismas normas 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).

expresión-generación no debe hacer referencia a una columna para la que se define una máscara de columna (SQLSTATE 42621).

SET NOT HIDDEN o SET IMPLICITLY HIDDEN

Especifica el atributo oculto para la columna.

Si la tabla es una tabla temporal de período de sistema, la columna también cambia en cualquier tabla histórica asociada.

NOT HIDDEN

Especifica que la columna se incluye como referencia implícita a la tabla y que se puede hacer referencia a la columna de manera explícita.

IMPLICITLY HIDDEN

Especifica que la columna no es visible en sentencias de SQL a menos que se haga referencia a la columna de manera explícita por el nombre. Por ejemplo, suponiendo que una tabla incluye una columna definida con la cláusula IMPLICITLY HIDDEN, el resultado de SELECT * no incluye la columna oculta implícitamente. Sin embargo, el resultado de SELECT que haga referencia de manera explícita al nombre de una columna implícitamente oculta incluirá dicha columna en la tala de resultados.

No debe especificarse IMPLICITLY HIDDEN para la última columna de la tabla que no está oculta (SQLSTATE 428GU).

SET *modificación-generación*

Especifica que se debe cambiar el atributo de generación para la columna. Se puede especificar GENERATED si la columna es una columna de identidad o una columna de indicación de fecha y hora de cambio de fila (SQLSTATE 42837). Si la tabla es una tabla temporal de período de sistema, la columna de la tabla histórica asociada no resulta afectada por el cambio. Si existe un valor por omisión para la columna, se debe descartar el valor por omisión; esto se puede hacer en la misma *modificación-columna* utilizando una de las cláusulas DROP DEFAULT. SET GENERATED no debe especificarse para una columna de una tabla histórica temporal (SQLSTATE 428FR).

GENERATED ALWAYS

Especifica que el gestor de bases de datos genere siempre un valor para la columna cuando se inserta o se actualiza una fila y se debe generar un valor. GENERATED ALWAYS es la opción recomendada, a menos que estén realizándose operaciones de propagación o de descarga y de recarga. ALWAYS es el valor por omisión de las columnas generadas.

GENERATED BY DEFAULT

Especifica que el gestor de bases de datos genere un valor para la columna cuando se inserte una fila en la tabla, o cuando se actualice especificando DEFAULT para la columna, a menos que se especifique un valor explícito. GENERATED BY DEFAULT sólo se puede especificar con *cláusula-indicación-hora-cambio-fila*. GENERATED BY

ALTER TABLE

DEFAULT es la opción recomendada cuando se utiliza la propagación de datos o cuando se realizan operaciones de descarga y recarga.

modificación-identidad

Modifica los atributos de identidad de la columna. Debe ser una columna de identidad.

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 puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente tras la sentencia ALTER. Si este valor es 0 o positivo, es una secuencia ascendente tras la sentencia ALTER.

SET NO MINVALUE o MINVALUE *constante-numérica*

Especifica el valor mínimo en el que una columna de identidad 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).

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.

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 esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor o igual al valor máximo (SQLSTATE 42815).

SET NO MAXVALUE o MAXVALUE *constante-numérica*

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).

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.

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 42815), sin dígitos distintos de cero a la

derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser superior o igual al valor mínimo (SQLSTATE 42815).

SET NO CYCLE o CYCLE

Especifica si esta columna de identidad debe o no seguir generando valores tras la generación de su valor máximo o de su valor mínimo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

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.

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, pueden generarse 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 exclusivo en una columna de identidad de este tipo y se genera un valor que no es exclusivo, se produce un error (SQLSTATE 23505).

SET NO CACHE o CACHE *constante-entera*

Especifica si deben mantenerse en la memoria algunos valores preasignados, para conseguir 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).

NO CACHE

Especifica que no se deben preasignar los valores de la columna de identidad. En un entorno DB2 pureScale, si los valores de identidad *deben* generarse en orden de petición, debe utilizarse la opción NO CACHE.

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.

CACHE *constante-entera*

Especifica cuántos valores de la secuencia de identidad deben asignarse previamente y conservarse en la memoria. Cuando se generan valores para la columna de identidad, la asignación previa y el almacenamiento de los valores en la antememoria reducen la E/S síncrona en el archivo de anotaciones cronológicas.

Si se necesita un nuevo valor para la columna de identidad y no existen valores no utilizados disponibles en la antememoria, la asignación del valor implica tener que esperar a que se produzca la

E/S en el archivo de anotaciones cronológicas. 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.

En caso de que se produzca una desactivación de la base de datos, realizada con normalidad o como consecuencia de una anomalía en el sistema, todos los valores de secuencia que se han colocado en la antememoria y que no se han utilizado en las sentencias confirmadas se pierden (es decir, nunca se utilizarán). El valor especificado para la 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 42815).

En un entorno DB2 pureScale, si se especifican tanto CACHE como ORDER, la especificación de ORDER altera temporalmente la especificación de CACHE y, en su lugar, NO CACHE entrará en vigor.

SET NO ORDER o 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).

NO ORDER

Especifica que los valores de la columna de identidad no necesitan generarse según el orden de petición.

ORDER

Especifica que los valores de la columna de identidad deben generarse en el orden en que se solicitan.

RESTART o RESTART WITH *constante-numérica*

Restablece el estado de la secuencia asociada con la columna de identidad. Si no se ha especificado WITH *constante-numérica*, la secuencia de la columna de identidad se reiniciará en el valor que se especificó, de forma implícita o explícita, como valor inicial para la columna de identidad durante su creación original.

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 START WITH original.

La *constante-numérica* es una constante numérica exacta que puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). La *constante-numérica* se utilizará como el siguiente valor para la columna.

SET *atributo-generación cláusula-as-identity*

Cambia la columna a una columna de identidad. Esta modificación de columna no se debe especificar si la columna tiene un valor por omisión o ya es una columna generada (SQLSTATE 42837). Si la tabla es una tabla temporal de período de sistema, la columna de la tabla histórica asociada no resulta afectada por el cambio.

GENERATED ALWAYS

Especifica que el gestor de bases de datos genere siempre un valor para la columna cuando se inserta o se actualiza una fila y se debe generar un valor. ALWAYS es el valor por omisión de las columnas generadas.

GENERATED BY DEFAULT

Especifica que el gestor de bases de datos genere un valor para la columna cuando se inserta o se actualiza una fila y se debe generar un valor por omisión, salvo que se especifique un valor explícito.

cláusula-as-identity

Especifica que se trata de la columna de identidad para la tabla. Una tabla sólo puede contener una única columna de identidad (SQLSTATE 428C1). La columna se debe especificar como sin posibilidad de nulos (SQLSTATE 42997) y el tipo de datos asociado a la columna debe ser numérico exacto con una escala de cero (SQLSTATE 42815). El tipo de datos numérico exacto puede ser: SMALLINT, INTEGER, BIGINT, DECIMAL o NUMERIC con una escala de cero, o un tipo diferenciado basado en uno de estos tipos. Para ver detalles sobre las opciones de identidad, consulte "CREATE TABLE".

SET GENERATED ALWAYS

Cambia la columna a una columna de expresión generada, una columna de inicio de fila, una columna de fin de fila o una columna de ID de inicio de transacción. GENERATED ALWAYS especifica que el gestor de bases de datos genere siempre un valor para la columna cuando se inserta o se actualiza una fila y se debe generar un valor.

AS (expresión-generación)

Especifica que la definición de la columna se basa en una expresión. La columna no debe estar definida ya con una expresión de generación, no puede ser la columna de identidad ni tampoco contener un valor por omisión explícito (SQLSTATE 42837). La *expresión-generación* debe cumplir las mismas normas que se aplican al definir una columna generada. Debe ser posible asignar el tipo de datos resultante de la *expresión-generación* al tipo de datos de la columna (SQLSTATE 42821). No se puede hacer referencia a la columna en la columna de clave de distribución ni en la cláusula ORGANIZE BY (SQLSTATE 42997).

expresión-generación no debe hacer referencia a una columna para la que se define una máscara de columna (SQLSTATE 42621).

AS ROW BEGIN

Especifica que el gestor de bases de datos asigna el valor siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El valor se genera mediante una lectura de la hora del reloj durante la ejecución del primer suceso de los sucesos de la transacción siguientes:

- Una sentencia de cambio de datos que exija que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla
- La eliminación de una fila en una tabla temporal de período de sistema

En el caso de una tabla temporal de período de sistema, el gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de inicio de fila en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas

ALTER TABLE

insertadas en una tabla histórica asociada tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio (SQLSTATE 01695). Ello puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema. El parámetro de configuración de base de datos **system_time_period_adj** debe establecerse en Yes para que tenga lugar este ajuste del valor de indicación de fecha y hora; de lo contrario, se devuelve un error (SQLSTATE 57062). Si se insertan o actualizan varias filas en una sola transacción de SQL y no es necesario un ajuste, los valores de la columna de inicio de fila son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción. Se necesita una columna de inicio de fila como columna de inicio de un período SYSTEM_TIME, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de inicio de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12) (SQLSTATE 42842). La columna debe definirse como NOT NULL (SQLSTATE 42831). Las columnas de inicio de fila no se pueden actualizar.

AS ROW END

Especifica que el gestor de bases de datos asigna el valor máximo del tipo de datos de la columna siempre que se inserta una fila o se actualiza una columna de la fila.

Se necesita una columna de fin de fila como segunda columna de un período SYSTEM_TIME, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de fin de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12) (SQLSTATE 42842). La columna debe definirse como NOT NULL (SQLSTATE 42831). Las columnas de fin de fila no se pueden actualizar.

AS TRANSACTION START ID

Especifica que el gestor de bases de datos asigna el valor siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El gestor de bases de datos asigna un valor de indicación de fecha y hora exclusivo por transacción o el valor nulo. El valor nulo se asigna a la columna de ID de inicio de transacción si la columna puede contener nulos y si existe una columna de inicio de fila en la tabla en la que el valor no tiene que ajustarse. En caso contrario, el valor se genera mediante una lectura de la hora del reloj durante la ejecución del primer suceso de los sucesos de la transacción siguientes:

- Una sentencia de cambio de datos que exija que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla
- La eliminación de una fila en una tabla temporal de período de sistema

Si se insertan o actualizan varias filas en una sola transacción de SQL, los valores de la columna de ID de inicio de transacción son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.

Se necesita una columna de ID de inicio de transacción para una tabla temporal de período de sistema, lo cual es el uso esperado para este tipo de columna generada.

Una tabla sólo puede tener una columna de ID de inicio de transacción (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como `TIMESTAMP(12)`. Si se especifica *tipo-datos*, debe ser `TIMESTAMP(12)`. Las columnas de ID de inicio de transacción no se pueden actualizar.

DROP DEFAULT

Descarta el valor por omisión actual para la columna. La columna especificada debe tener un valor por omisión (SQLSTATE 42837). Esta acción se propaga a la tabla histórica para una tabla temporal de período de sistema.

DROP GENERATED

Descarta los atributos generados de la columna. La columna debe definirse como una columna generada (SQLSTATE 42837). La columna no debe definirse como una columna de inicio de fila, de fin de fila ni de ID de inicio de transacción en una tabla temporal de período de sistema (SQLSTATE 428FR).

DROP NOT NULL

Descarta el atributo NOT NULL de la columna, lo que permite a la columna tener un valor nulo. Esta cláusula no se permite si la columna se especifica en la clave primaria, en una restricción exclusiva de la tabla (SQLSTATE 42831), una columna de inicio de fila o una columna de fin de fila (SQLSTATE 42837). Para modificar este atributo para una columna, es necesaria la reorganización de la tabla para poder acceder posteriormente a la tabla (SQLSTATE 57016). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997). DROP NOT NULL está bloqueado para columnas que pertenecen al período `BUSINESS_TIME` (SQLSTATE 428FR).

Si la tabla es una tabla temporal de período de sistema, el atributo NOT NULL también se descarta de la columna correspondiente en cualquier tabla histórica asociada.

Si existe un permiso de fila o una máscara de columna, que depende de la columna que se debe alterar, se emitirá un error (SQLSTATE 42917).

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 que se modifica es una tabla con tipo, la columna no debe ser herencia de una supertabla (SQLSTATE 428DJ).

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 ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en el *nombre-tabla-tipo*.

nombre-vista-tipo

El nombre de una vista con tipo. El tipo de datos de *nombre-columna*

ALTER TABLE

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*.

COMPRESS

Especifica si los valores por omisión de esta columna van a almacenarse o no de forma más eficaz.

SYSTEM DEFAULT

Especifica que los valores por omisión del sistema (es decir, los valores por omisión que se utilizan para los tipos de datos cuando no se ha especificado ningún valor específico) van a almacenarse utilizando el espacio mínimo. Si la tabla todavía no se ha establecido con el atributo VALUE COMPRESSION activado, se devuelve un mensaje de aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el espacio mínimo.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

Los datos que existen en la columna no cambian. Considere la reorganización de la tabla fuera de línea para que los datos existentes puedan beneficiarse del almacenamiento de los valores por omisión del sistema en el que se utiliza el espacio mínimo.

OFF

Especifica que los valores por omisión del sistema van a almacenarse en la columna como valores normales. Los datos que existen en la columna no cambian. Se recomienda la reorganización fuera de línea para cambiar los datos existentes.

El tipo de datos base no debe ser DATE, TIME ni TIMESTAMP (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto. Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

Si la tabla que se modifica es una tabla con tipo, la columna no debe ser herencia de una supertabla (SQLSTATE 428DJ).

SECURED WITH *nombre-etiqueta-seguridad*

Identifica una etiqueta de seguridad existente para la política de seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064). La tabla no debe ser una tabla temporal de período de sistema.

DROP COLUMN SECURITY

Modifica una columna para establecerla como no protegida.

ACTIVATE ROW ACCESS CONTROL

Activa el control de acceso de nivel de fila en la tabla. La tabla no debe ser una tabla con tipo, una tabla de catálogo (SQLSTATE 55019), una tabla temporal creada, una tabla temporal declarada (SQLSTATE 42995), un apodo (SQLSTATE 42809) o una vista (SQLSTATE 42809).

Se crea un permiso de fila por omisión de forma implícita que no permite el acceso a ninguna fila de la tabla, a menos que lo permita de forma explícita un permiso de fila creado por un usuario con autorización SECADM.

Cuando se hace referencia a la tabla en una sentencia de manipulación de datos, la base de datos DB2 aplica de forma implícita todos los permisos de fila que se han creado para la tabla, incluido el permiso de fila por omisión, con el fin de controlar el conjunto de filas de la tabla a las que se puede acceder.

Si existe un activador para la tabla, el activador debe definirse con el atributo SECURED (SQLSTATE 55019).

No debe hacerse referencia a la tabla en la definición de una vista si existe un activador INSTEAD OF que se ha definido con el atributo NOT SECURED para la vista (SQLSTATE 55019).

Si una tabla de consulta materializada hace referencia a la tabla, las funciones a las que se hace referencia en la *selección-completa de definición-consulta-materializada* deben definirse con el atributo SECURED (SQLSTATE 55019).

Si una tabla de consulta materializada (o una tabla de etapas) que depende de la tabla (directa o indirectamente a través de una vista) para la que se va a activar el control de acceso de nivel de fila y esa tabla de consulta materializada (o tabla de etapas) aún no tienen el control de acceso de nivel de fila activado, el control de acceso de nivel de fila se activa de forma implícita para la tabla de consulta materializada (o la tabla de etapas). Esto limita el acceso directo al contenido de la tabla de consulta materializada (o la tabla de etapas). Una consulta que hace referencia explícita a la tabla antes de que se defina ese permiso de fila devolverá un aviso de que no existen datos en la tabla (SQLSTATE 02000). Para proporcionar acceso a la tabla de consulta materializada (o tabla de etapas), se puede crear un permiso de fila adecuado o se puede emitir una sentencia ALTER TABLE DEACTIVATE ROW ACCESS CONTROL en la tabla de consulta materializada (o tabla de etapas) para eliminar la protección de nivel de fila si ello es adecuado.

ACTIVATE ROW ACCESS CONTROL se ignora si el control de acceso a filas ya se ha definido como activado para la tabla.

Si la tabla es una tabla temporal de período de sistema, el gestor de bases de datos activa automáticamente el control de acceso a filas en la tabla histórica y crea un permiso de fila por omisión para la tabla histórica.

ACTIVATE COLUMN ACCESS CONTROL

Activa el control de acceso de nivel de columna en la tabla. La tabla no debe ser una tabla con tipo, una tabla de catálogo (SQLSTATE 55019), una tabla temporal creada, una tabla temporal declarada (SQLSTATE 42995), un apodo (SQLSTATE 42809) o una vista (SQLSTATE 42809).

El acceso a la tabla no se restringe, pero cuando se hace referencia a la tabla en una sentencia de manipulación de datos, el gestor de bases de datos aplica de forma implícita todas las máscaras de columna habilitadas que se hayan creado para la tabla con el fin de enmascarar los valores devueltos para las columnas referenciadas en la tabla de resultados final de las consultas.

Si existe un activador para la tabla, el activador debe definirse con el atributo SECURED (SQLSTATE 55019).

Si una tabla de consulta materializada hace referencia a la tabla, las funciones a las que se hace referencia en la *selección-completa de definición-consulta-materializada* deben definirse con el atributo SECURED (SQLSTATE 55019).

ALTER TABLE

No debe hacerse referencia a la tabla en la definición de una vista si existe un activador **INSTEAD OF** que se ha definido con el atributo **NOT SECURED** para la vista (SQLSTATE 55019). Si una tabla de consulta materializada que depende de la tabla (directa o indirectamente a través de una vista) para la que se va a activar el control de acceso de nivel de columna y esa tabla de consulta materializada aún no tienen el control de acceso de nivel de fila activado, el control de acceso de nivel de fila se activa de forma implícita para la tabla de consulta materializada. Esto limita el acceso directo al contenido de la tabla de consulta materializada. Una consulta que hace referencia explícita a la tabla antes de que se defina ese permiso de fila devuelve un aviso de que no existen datos en la tabla (SQLSTATE 02000). Para proporcionar acceso a la tabla de consulta materializada, se puede crear un permiso de fila adecuado o se puede emitir una sentencia **ALTER TABLE DEACTIVATE ROW ACCESS CONTROL** en la tabla de consulta materializada para eliminar la protección de nivel de fila si ello es adecuado.

ACTIVATE COLUMN ACCESS CONTROL se ignora si el control de acceso a columnas ya se ha definido como activado para la tabla.

Si la tabla es una tabla temporal de período de sistema, el gestor de bases de datos activa automáticamente el control de acceso a filas en la tabla histórica y crea un permiso de fila por omisión para la tabla histórica.

DEACTIVATE ROW ACCESS CONTROL

Desactiva el control de acceso de nivel de fila en la tabla. Cuando se hace referencia a la tabla en una sentencia de manipulación de datos, el gestor de bases de datos no aplica los permisos de fila habilitados existentes definidos en la tabla con el fin de controlar el conjunto de filas en la tabla a las que se puede acceder.

DEACTIVATE ROW ACCESS CONTROL se ignora si el control de acceso a filas no se activa para la tabla.

DEACTIVATE COLUMN ACCESS CONTROL

Desactiva el control de acceso de nivel de columna en la tabla. Cuando se hace referencia a la tabla en una sentencia de manipulación de datos, el gestor de bases de datos no aplica las máscaras de columna habilitadas definidas en la tabla con el fin de controlar los valores devueltos para las columnas referencias en la tabla de resultados final de las consultas.

DEACTIVATE COLUMN ACCESS CONTROL se ignora si el control de acceso a columnas no se activa para la tabla.

RENAME COLUMN *nombre-columna-fuente* TO *nombre-columna-destino*

Cambia el nombre de la columna que se especifica en *nombre-columna-fuente* por el nombre que se especifica en *nombre-columna-destino*. Si el parámetro de configuración de base de datos **auto_reval** se ha establecido en **DISABLED**, la opción **RENAME COLUMN** de la sentencia **ALTER TABLE** se comporta como si estuviera bajo el control de la semántica **REVALIDATION IMMEDIATE**.

La tabla no debe definirse como tabla histórica (SQLSTATE 42986). Si la tabla es una tabla temporal de período de sistema, la columna también se renombra en cualquier tabla histórica asociada.

RENAME COLUMN no debe renombrar una columna a la que se hace referencia en la definición de un permiso de fila o máscara de columna. Además, no debe renombrar una columna para la que hay definida una máscara de columna (SQLSTATE 42917). Si modifica el nombre de una columna que pertenece a una tabla en la que se define una máscara o permiso, o a una tabla a la que se hace

referencia en la definición de una máscara o permiso, dicha máscara o permiso queda invalidado. El acceso a una tabla que tenga activado el control de acceso a columnas y tenga definida una máscara no válida queda bloqueado hasta que la máscara no válida se inhabilite, descarte o vuelva a crear (SQLSTATE 560D0). El acceso a una tabla que tenga activado el control de acceso a filas y tenga definido un permiso de fila no válido queda bloqueado hasta que el permiso no válido se inhabilite, descarte o vuelva a crear (SQLSTATE 560D0).

nombre-columna-fuente

Especifica el nombre de la columna a la que se va a cambiar el nombre. El *nombre-columna-fuente* debe identificar una columna existente de la tabla (SQLSTATE 42703). El nombre no puede estar calificado. El nombre no puede identificar una columna que de algún otro modo se añade, descarta o modifica en la misma sentencia ALTER TABLE (SQLSTATE 42711).

nombre-columna-destino

El nuevo nombre de la columna. El nombre no puede estar calificado. No deben utilizarse nombres de columna o nombres de período existentes en la tabla (SQLSTATE 42711).

DROP PRIMARY KEY

Descarta la definición de la clave primaria y todas las restricciones de referencia dependientes de esta clave primaria. La tabla ya debe tener una clave primaria (SQLSTATE 42888).

DROP FOREIGN KEY *nombre-restricción*

Descarta la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción referencial, consulte la sección Notas.

DROP UNIQUE *nombre-restricción*

Descarta 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 de unicidad existente (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción de unicidad, consulte el apartado Notas.

DROP CHECK *nombre-restricción*

Descarta la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente definida en la tabla (SQLSTATE 42704).

DROP CONSTRAINT *nombre-restricción*

Descarta la restricción *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación, restricción de referencia, clave primaria o restricción de unicidad existente definida en la tabla (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción, consulte la sección Notas.

DROP COLUMN

Descarta la columna identificada de la tabla. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997). Si se descarta una columna, la tabla debe reorganizarse antes de realizar una operación de actualización, inserción o supresión o una exploración del índice en la tabla (SQLSTATE 57016). Sólo se puede descartar una columna XML si el resto de columnas XML de la tabla se descartan a la vez.

ALTER TABLE

DROP COLUMN no debe descartar una columna a la que se hace referencia en la definición de un permiso de fila o máscara de columna (SQLSTATE 42917). Sin embargo, se puede descartar una columna para la que se defina una máscara de columna. Cuando la columna se descarta, las máscaras de columna definidas en esa columna también se descartan.

nombre-columna

Identifica la columna que se descartará. El nombre de columna no puede estar calificado. El nombre debe identificar una columna de la tabla especificada (SQLSTATE 42703). El nombre no puede identificar la única columna de la tabla (SQLSTATE 42814) o una columna referenciada en la definición de un período (SQLSTATE 42817). El nombre no debe identificar la última columna de la tabla que no está oculta (SQLSTATE 428GU). El nombre no debe identificar una columna en una tabla que se defina como tabla temporal de período de sistema o tabla histórica (SQLSTATE 428FR). El nombre no debe identificar una columna que forme parte de la clave de distribución, la clave de particionamiento de la tabla o las dimensiones de organización (SQLSTATE 42997).

CASCADE

Especifica las siguientes acciones, en función del objeto:

- Las vistas que dependen de la columna que se desea descartar se marcan como no operativas.
- Los índices, activadores, funciones de SQL, restricciones o variables globales que dependen de la columna que se desea descartar también se descartan.
- Los XSROBJECT habilitados para descomposición que dependen de la tabla que contiene la columna se establecen como no operativos para la descomposición.

Un activador depende de la columna si se le hace referencia en la lista de columnas UPDATE OF o en cualquier ubicación de la acción activada. Un XSROBJECT con la descomposición habilitada depende de una tabla si contiene una correlación de un atributo o elemento XML con la tabla. Si una función SQL o variable global depende de otro objeto de base de datos, tal vez no sea posible descartar la función o variable global mediante la opción CASCADE. CASCADE es el valor por omisión.

RESTRICT

Especifica que no se puede descartar la columna si hay vistas, índices, activadores, restricciones o variables globales dependientes de la columna o si hay un XSROBJECT con la descomposición habilitada dependiente de la tabla que contiene la columna (SQLSTATE 42893). Un activador depende de la columna si se le hace referencia en la lista de columnas UPDATE OF o en cualquier ubicación de la acción activada. Un XSROBJECT con la descomposición habilitada depende de una tabla si contiene una correlación de un atributo o elemento XML con la tabla. El primer objeto dependiente que se detecta se identifica en las anotaciones cronológicas de administración.

Tabla 15. Efectos en cascada de descartar una columna

Operación	Efecto de RESTRICT	Efecto de CASCADE
Descartar una columna a la que hace referencia una vista o un activador	No se puede descartar la columna.	El objeto y todos los objetos dependientes de ese objeto se descartan.

Tabla 15. Efectos en cascada de descartar una columna (continuación)

Operación	Efecto de RESTRICT	Efecto de CASCADE
Descartar una columna a la que se hace referencia en la clave de un índice	Si todas las columnas a las que se hace referencia en el índice se descartan en la misma sentencia ALTER TABLE, se puede descartar el índice. De lo contrario, no se puede descartar la columna.	Se descarta el índice.
Descartar una columna a la que se hace referencia en una restricción de unicidad	Si todas las columnas a las que se hace referencia en la restricción de unicidad se descartan en la misma sentencia ALTER TABLE, y no hace referencia a la restricción de unicidad una restricción de referencia, se descartan las columnas y la restricción. (El índice empleado para satisfacer la restricción también se descarta.) De lo contrario, no se puede descartar la columna.	La restricción de unicidad y las restricciones de referencia que hacen referencia a esa restricción de unicidad se descartan. (Los índices empleados por esas restricciones también se descartan.)
Descartar una columna a la que se hace referencia en una restricción de referencia	Si todas las columnas a las que se hace referencia en la restricción de referencia se descartan en la misma sentencia ALTER TABLE, se descartan las columnas y la restricción. De lo contrario, no se puede descartar la columna.	Se descarta la restricción de referencia.
Descartar una columna a la que hace referencia una columna generada por el sistema que no se descarta	No se puede descartar la columna.	No se puede descartar la columna.
Descartar una columna a la que se hace referencia en una restricción de comprobación	No se puede descartar la columna.	Se descarta la restricción de comprobación.
Descartar una columna a la que se hace referencia en un XSROBJECT con la descomposición habilitada	No se puede descartar la columna.	El XSROBJECT se marca como inoperativo para la descomposición. Para volver a habilitar el XSROBJECT, puede que sea necesario volver a ajustar sus correlaciones; a continuación, emita una sentencia ALTER XSROBJECT ENABLE DECOMPOSITION para el XSROBJECT.

ALTER TABLE

Tabla 15. Efectos en cascada de descartar una columna (continuación)

Operación	Efecto de RESTRICT	Efecto de CASCADE
Descartar una columna a la que se hace referencia en la expresión por omisión de una variable global	No se puede descartar la columna.	Se descarta la variable global, a menos que no esté permitido descartar la variable global porque hay otros objetos, que no permiten la cascada, que dependen de la variable global.

DROP RESTRICT ON DROP

Elimina la restricción, si existe, sobre la operación de descartar la tabla y el espacio de tablas que contiene la tabla.

DROP DISTRIBUTION

Descarta la definición de distribución de la tabla. La tabla debe tener una definición de distribución (SQLSTATE 428FT). El espacio de tablas de la tabla debe estar definido en un grupo de particiones de base de datos de partición única.

DROP MATERIALIZED QUERY

Cambia una tabla de consulta materializada de modo que ya no se considera una tabla de consulta materializada. La tabla que *nombre-tabla* especifica debe definirse como una tabla de consulta materializada que no está duplicada (SQLSTATE 428EW). La definición de las columnas de *nombre-tabla* no se cambia, pero ya no se puede emplear la tabla para la optimización de consulta, y ya no se puede utilizar la sentencia REFRESH TABLE.

Si el control de acceso de nivel de fila o el control de acceso de nivel de columna está en vigor para la tabla, este control permanece después de que la tabla deje de ser una tabla de consulta materializada.

ADD PERIOD definición-período

Añade una definición de período a la tabla.

SYSTEM_TIME (*nombre-columna-inicio*, *nombre-columna-fin*)

Define un período de sistema con el nombre SYSTEM_TIME. No debe haber una columna en la tabla con el nombre SYSTEM_TIME (SQLSTATE 42711). Una tabla sólo puede tener un período SYSTEM_TIME (SQLSTATE 42711). *nombre-columna-inicio* debe definirse como ROW BEGIN y *nombre-columna-fin* debe definirse como ROW END (SQLSTATE 428HN).

BUSINESS_TIME (*nombre-columna-inicio*, *nombre-columna-fin*)

Define un período de aplicación con el nombre BUSINESS_TIME. No debe haber una columna en la tabla con el nombre BUSINESS_TIME (SQLSTATE 42711). Una tabla sólo puede tener un período BUSINESS_TIME (SQLSTATE 42711). *nombre-columna-inicio* y *nombre-columna-fin* deben definirse ambos como DATE o TIMESTAMP(p) donde p es de 0 a 12 (SQLSTATE 42842) y las columnas deben definirse como NOT NULL (SQLSTATE 42831). *nombre-columna-inicio* y *nombre-columna-fin* no deben identificar una columna que se defina con una cláusula GENERATED (SQLSTATE 428HZ). Las columnas de período de tiempo empresarial no pueden añadirse a una tabla que se encuentra en estado pendiente de establecimiento de integridad.

Una restricción de comprobación implícita se genera para garantizar que el valor de *nombre-columna-fin* sea superior que el valor de

nombre-columna-inicio. El nombre de la limitación de comprobación creada de forma implícita es `DB2_GENERATED_CHECK_CONSTRAINT_FOR_BUSINESS_TIME` y no debe ser el nombre de una limitación de comprobación existente (SQLSTATE 42710).

DROP PERIOD *nombre-período*

Descarta el período identificado de la tabla. El nombre no debe identificar un período que ya se haya añadido o alterado en esta sentencia `ALTER TABLE` (SQLSTATE 42711). Las restricciones de comprobación generadas de forma implícita para el período (creadas cuando se ha definido el período) y los índices que hacen referencia al período también se descartan.

nombre-período

Identifica el período. Los nombres de período válido son `BUSINESS_TIME` o `SYSTEM_TIME`. El período debe existir en la tabla (SQLSTATE 4274M).

Cuando se descarta un período `BUSINESS_TIME`, se invalidan todos los paquetes con el tipo de dependencia de tabla temporal de período de aplicación de esa tabla. Otros objetos dependientes como vistas y activadores que registran una dependencia en la tabla también se marcan como no válidos.

El período `SYSTEM_TIME` no se puede descartar si la tabla es una tabla temporal de período de sistema (SQLSTATE 428HZ).

DATA CAPTURE

Indica si debe grabarse información adicional para la duplicación de datos en las anotaciones cronológicas.

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 cronológicas se registrará 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 el nombre de esquema (implícito o explícito) de la tabla tiene más de 18 bytes, esta opción no recibe soporte (SQLSTATE 42997).

INCLUDE LONGVAR COLUMNS

Permite que los programas de utilidad de duplicación de datos capten los cambios realizados en las columnas `LONG VARCHAR` o `LONG VARGRAPHIC`. La cláusula se puede especificar para las tablas que no tengan columnas `LONG VARCHAR` ni `LONG VARGRAPHIC` puesto que es posible modificar la tabla para que incluya estas columnas.

ACTIVATE NOT LOGGED INITIALLY

Activa el atributo `NOT LOGGED INITIALLY` de la tabla para esta unidad de trabajo actual.

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

ALTER TABLE

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 la ejecución de esta sentencia y la finalización de 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).

Si la tabla es una tabla temporal de período de sistema o una tabla histórica, no se da soporte a esta opción.

Para obtener más información sobre el atributo NOT LOGGED INITIALLY, consulte la descripción de este atributo en "CREATE TABLE".

Nota: Si se produce actividad que no es de anotaciones cronológicas para una tabla que tiene activado el atributo NOT LOGGED INITIALLY y si una sentencia no se ejecuta satisfactoriamente (dando lugar a una retroacción) o si se ejecuta ROLLBACK TO SAVEPOINT, se retrotraerá la unidad de trabajo completa (SQL1476N). Además, la tabla para la que se había activado el atributo NOT LOGGED INITIALLY se marcará como no accesible tras producirse la retroacción y sólo podrá descartarse. 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 activador DELETE definido en la tabla afectada. Los datos de índice también se suprimirá para todos los índices que existen en la tabla.

No puede vaciarse una tabla particionada con particiones de datos enlazadas o particiones desenlazadas lógicamente (SQLSTATE 42928).

PCTFREE *entero*

Especifica el porcentaje de cada página que se debe dejar como espacio libre durante una operación de carga o de reorganización de tabla. La primera fila de cada página se añade sin restricciones. Cuando se añaden filas adicionales a una página, al menos *entero* por ciento de la página se deja como espacio libre. El valor PCTFREE sólo se tiene en cuenta en los programas de utilidad de carga y de reorganización de tablas. El valor de *entero* entra en un rango que va de 0 a 99. El valor -1 de PCTFREE en el catálogo del sistema (SYSCAT.TABLES) se interpreta como el valor por omisión. El valor de

PCTFREE por omisión para una página de tabla es 0. 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. El uso de esta opción en la definición de la tabla no impedirá que se produzca el escalamiento 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. Este es el tamaño de bloqueo por omisión cuando se crea una tabla.

BLOCKINSERT

Indica el uso de bloqueos de bloques en las operaciones de inserción. Esto significa que se adquiere el bloqueo exclusivo adecuado en el bloque antes de la inserción y el bloqueo de fila no se realiza en la fila insertada. Esta opción resulta de utilidad al insertar distintas transacciones en diferentes celdas de la tabla. Las transacciones que efectúan inserciones en las mismas celdas pueden seguir haciéndolo de manera simultánea, pero realizarán las inserciones en distintos bloques, y esto puede afectar al tamaño de la celda si se necesitan más bloques. Esta opción solo es válida para las tablas MDC (SQLSTATE 42613).

TABLE

Indica la utilización de bloqueos de tabla. Esto significa que se adquiere el bloqueo de compartimiento o exclusivo adecuado en la tabla y no se utilizan bloqueos de intento (excepto el valor de ningún intento). En el caso de las tablas particionadas, esta técnica de bloqueo se aplica tanto al bloqueo de tabla como a los bloqueos de partición de datos para las particiones de datos a las que se accede. El uso de este valor puede mejorar el rendimiento de las consultas limitando el número de bloqueos que es necesario adquirir. Sin embargo, también se reduce la simultaneidad, dado que se retienen todos los bloqueos en toda la tabla.

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. Este 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 CARDINALITY o NOT VOLATILE CARDINALITY

Indica al optimizador si la cardinalidad de la tabla *nombre-tabla* puede variar significativamente en tiempo de ejecución o no. La volatilidad se aplica al número de filas de la tabla, no a la propia tabla. **CARDINALITY** es una palabra clave opcional. El valor por omisión es **NOT VOLATILE**.

ALTER TABLE

VOLATILE

Especifica que la cardinalidad de la tabla *nombre-tabla* puede variar significativamente en tiempo de ejecución, de vacía a grande. Para acceder a la tabla, el optimizador utilizará una exploración de índice (en vez de una exploración de tabla, sin tener en cuenta las estadísticas) si el índice es sólo índice (todas las columnas de referencia se encuentran en el índice) o si puede aplicar un predicado en la exploración de índice. No se utilizará el método de acceso de captación previa de lista para acceder a la tabla. 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

Especifica que la cardinalidad de *nombre-tabla* no es volátil. Los planes de acceso a esta tabla seguirán basándose en las estadísticas existentes y en el nivel de optimización actual.

COMPRESS

Especifica si se aplica la compresión de datos a las filas de la tabla.

YES

Especifica que la compresión de filas y de XML están habilitadas. Las operaciones de inserción y actualización de la tabla estarán sujetas a compresión. La compresión de índice estará habilitada para los índices nuevos, a menos que se deshabilite explícitamente en la sentencia CREATE INDEX. Los índices existentes pueden comprimirse utilizando la sentencia ALTER INDEX.

Después de que se haya alterado una tabla para habilitar la compresión de filas, todas las filas de la tabla se pueden comprimir de inmediato realizando una de las acciones siguientes:

- Mandato REORG
- Traslado de tablas en línea
- Carga y recarga de datos

ADAPTIVE

Habilita la compresión adaptativa para la tabla. Las filas de datos están sujetas a la compresión con diccionarios de compresión tanto de nivel de tabla como de nivel de página. Los documentos XML del objeto de almacenamiento XML están sujetos a la compresión con un diccionario de compresión XML de nivel de tabla. Los diccionarios de compresión de nivel de página se crean automáticamente a medida que se insertan o actualizan las filas. Los diccionarios de compresión de nivel de tabla se crean tanto para datos XML como para filas automáticamente cuando se añade una cantidad de datos suficiente, a menos que ya existan.

STATIC

Habilita la compresión de filas clásica para la tabla. Las filas de datos están sujetas a compresión con un diccionario de compresión de nivel de tabla y los documentos XML del objeto de almacenamiento XML están sujetos a compresión utilizando un diccionario de compresión XML de nivel de tabla. Si no existen diccionarios de compresión de nivel de tabla para datos XML o filas, se crearán automáticamente cuando se añada una cantidad de datos suficiente.

Si no se especifica ninguna de las dos opciones anteriores con la cláusula COMPRESS YES, se utiliza ADAPTIVE de forma implícita.

NO Especifica que la compresión de filas de datos y de XML están

inhabilitadas. Los documentos XML y de filas de datos insertados y actualizados de la tabla ya no estarán sujetos a compresión. Los documentos XMLS y de filas que tengan el formato comprimido permanecerán en formato comprimido hasta que se conviertan al formato no comprimido cuando se actualicen. Al efectuarse una reorganización fuera de línea de la tabla, se descomprimen todas las filas que continúen comprimidas. Si existen diccionarios de compresión de nivel de tabla o de nivel de página, se descartan durante el truncamiento o la reorganización de tablas (por ejemplo, una operación LOAD REPLACE). La compresión de índice se inhabilita para los índices nuevos creados en esa tabla a menos que se habilite explícitamente en la sentencia CREATE INDEX. La compresión de índices para índices existentes puede inhabilitarse de forma explícita utilizando la sentencia ALTER INDEX.

VALUE COMPRESSION

Determina el formato de fila que debe utilizarse. Cada tipo de datos tiene un número total de bytes distinto según el formato de fila que se utilice. Si desea más información, consulte “Número total de bytes” en “CREATE TABLE”. Una operación de actualización causa que una fila existente pase a tener el nuevo formato de fila. Para mejorar el rendimiento de las operaciones de actualización en las filas existentes, se recomienda la reorganización de tablas fuera de línea. Esta acción también puede dar como resultado que la tabla ocupe menos espacio. Si el tamaño de la fila, calculado utilizando la columna correspondiente de la tabla llamada “Número total de bytes de las columnas por tipo de datos” (consulte “CREATE TABLE”), ya no cabe dentro del límite del tamaño de fila, indicado en la tabla llamada “Límites para el número de columnas y el tamaño de fila de cada tamaño de página del espacio de tablas”, se devolverá un error (SQLSTATE 54010). 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).

ACTIVATE

El valor NULL se almacena utilizando tres bytes. Es el mismo espacio o menor que cuando VALUE COMPRESSION no se ha activado para las columnas de todos los tipos de datos, con la excepción de CHAR(1). El hecho de definir o no una columna como con posibilidad de nulos no afecta al cálculo del tamaño de fila. Los valores de datos de longitud cero para las columnas cuyo tipo de datos es VARCHAR, VARGRAPHIC, CLOB, DBCLOB o BLOB deben almacenarse utilizando sólo dos bytes, que es un valor inferior al almacenamiento que se necesita cuando VALUE COMPRESSION no se ha activado. Cuando una columna se define con la opción COMPRESS SYSTEM DEFAULT, se permite también que el valor por omisión del sistema perteneciente a la columna se almacene utilizando tres bytes de almacenamiento total. El formato de fila utilizado para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos al actualizar al valor NULL, a un valor de longitud cero o al valor por omisión del sistema o bien al actualizar desde estos valores.

DEACTIVATE

El valor null se almacena con espacio establecido aparte para posibles actualizaciones futuras. Este espacio no se establece aparte para las columnas de longitud variable. Además, no da soporte al almacenamiento eficaz de los valores por omisión del sistema correspondientes a una columna. Si la columna ya existe y tiene el atributo COMPRESS SYSTEM DEFAULT, se devolverá un mensaje de aviso (SQLSTATE 01648).

ALTER TABLE

LOG INDEX BUILD

Especifica el nivel de anotación cronológica que se debe realizar durante las operaciones de crear, volver a crear o reorganizar el índice de esta tabla.

NULL

Especifica que se utilizará el valor del parámetro de configuración de base de datos **logindexbuild** para determinar si se debe realizar una anotación cronológica completa de las operaciones de creación de índice o no. Es el valor por omisión cuando se crea la tabla.

OFF

Especifica que se realizará una anotación cronológica mínima de cualquier operación de creación de índice de esta tabla. Este valor altera temporalmente el valor del parámetro de configuración de base de datos **logindexbuild**.

ON

Especifica que se realizará una anotación cronológica completa de las operaciones de creación de índice de esta tabla. Este valor altera temporalmente el valor del parámetro de configuración de base de datos **logindexbuild**.

ADD PARTITION *partición-adición*

Añade una o varias particiones de datos a una tabla particionada. Si la tabla especificada no es una tabla particionada, se devuelve un error (SQLSTATE 428FT). El número de particiones de datos no puede ser superior a 32 767.

nombre-partición

Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

espec-límite

Especifica el rango de valores de la partición de datos nueva. Este rango no puede solapar el de una partición de datos existente (SQLSTATE 56016). Para ver una descripción de la cláusula-inicial y la cláusula-final, consulte "CREATE TABLE".

Si se omite la cláusula-inicial, la partición de datos nueva se supone que se encuentra al final de la tabla. Si se omite la cláusula-final, la partición de datos nueva se supone que se encuentra al principio de la tabla.

IN *nombre-espacio-tablas*

Especifica el espacio de tablas donde debe almacenarse la partición de datos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas de la tabla particionada (SQLSTATE 42838). Puede ser un espacio de tablas que ya esté en uso para otra partición de datos de la misma tabla o un espacio de tablas que esta tabla no utilice actualmente, pero debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE (SQLSTATE 42727). Si no se especifica esta cláusula, se utiliza el espacio de tablas de la primera partición de datos visible o adjunta de la tabla.

INDEX IN *nombre-espacio-tablas*

Especifica el espacio de tablas donde se almacenan los índices particionados de la partición de datos. Si no se especifica la cláusula

INDEX IN, los índices particionados de la partición de datos se almacenan en el mismo espacio de tablas que la partición de datos.

El espacio de tablas que utiliza la nueva partición de índice, tanto si es el valor por omisión o el que especifica la cláusula INDEX IN, debe coincidir con el tipo (SMS o DMS), el tamaño de página y el tamaño de extensión de los espacios de tablas que utilizan otras particiones de índice (SQLSTATE 42838).

LONG IN *nombre-espacio-tablas*

Especifica el espacio de tablas donde debe almacenarse la partición de datos que contiene datos de columna largos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas y particiones de datos de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE. El tamaño de página y el tamaño de extensión del espacio de tablas especificado puede ser distinto al tamaño de página y al tamaño de extensión de las otras particiones de datos de la tabla particionada.

Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

ATTACH PARTITION *partición-enlace*

Enlaza otra tabla como una partición de datos nueva. El objeto de datos de la tabla que se enlaza se convierte en una partición nueva de la tabla a la que se enlaza. No existe ningún movimiento de datos. La tabla se coloca en estado pendiente de establecer integridad y la comprobación de la integridad de referencia se difiere hasta la ejecución de una sentencia SET INTEGRITY. La operación ALTER TABLE ATTACH no permite el uso de la cláusula IN o LONG. La colocación de LOB para la partición de datos en cuestión se determina al crear la tabla fuente. Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

Si la tabla que se va a enlazar tiene el control de acceso de nivel de fila o el control de acceso de nivel de columna activado, la tabla que se vaya a enlazar debe tener los mismos controles activados. Los permisos de fila o las máscaras de columna no se transfieren de forma automática de la tabla que se va a enlazar a la tabla de destino. Las máscaras de columna y los permisos de fila no tienen que ser exactamente iguales en las dos tablas, aunque sería lo idóneo desde el punto de vista de la seguridad. No obstante, si la tabla que se va a conectar tiene el control de acceso de nivel de fila activado, la tabla con la que se debe conectar también debe tener el control de acceso de nivel de fila activado (SQLSTATE 428GE). De forma similar, si la tabla que se va a conectar tiene el control de acceso de nivel de columna activado y, como mínimo, un objeto de máscara de columna habilitado, la tabla con la que se debe conectar también debe tener el control de acceso de nivel de columna activado y un objeto de máscara de columna habilitado para las columnas correspondientes (SQLSTATE 428GE).

nombre-partición

Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

ALTER TABLE

espec-límite

Especifica el rango de valores de la partición de datos nueva. Este rango no puede solapar el de una partición de datos existente (SQLSTATE 56016). Para ver una descripción de la cláusula-inicial y la cláusula-final, consulte "CREATE TABLE".

Si se omite la cláusula-inicial, la partición de datos nueva se supone que se encuentra al final de la tabla. Si se omite la cláusula-final, la partición de datos nueva se supone que se encuentra al principio de la tabla.

FROM *nombre1-tabla*

Especifica la tabla que se utilizará como fuente de datos de la partición nueva. La definición de tabla de *nombre-tabla1* no puede tener varias particiones de datos y debe coincidir con la tabla modificada en los aspectos siguientes (SQLSTATE 428GE):

- El número de columnas debe ser el mismo.
- Los tipos de datos de las columnas de la misma posición ordinal en la tabla deben ser los mismos.
- La característica de capacidad de nulos de las columnas de la misma posición ordinal en la tabla debe ser la misma.
- Si la tabla de destino tiene una columna de indicación de fecha y hora de cambio de fila, la columna correspondiente de la tabla de origen debe ser una columna de indicación de fecha y hora de cambio de fila.
- Si los datos también se distribuyen, deben distribuirse en el mismo grupo de particiones de base de datos mediante el mismo método de distribución.
- Si los datos de cualquiera de las tablas están organizados, la organización debe coincidir.
- Para tipos de datos estructurados, XML o LOB, el valor de `INLINE LENGTH` debe ser igual.
- Si la tabla de destino tiene definido un periodo `BUSINESS_TIME`, la tabla fuente debe tener definido un periodo `BUSINESS_TIME` en las columnas correspondientes.

Una vez enlazados correctamente los datos de *nombre-tabla1*, se realiza una operación equivalente a `DROP TABLE nombre-tabla1` para eliminar esta tabla, que ya no tiene datos, de la base de datos.

BUILD MISSING INDEXES

Especifica que si la tabla fuente no tiene índices que corresponden a los índices particionados en la tabla de destino, una operación `SET INTEGRITY` crea índices particionados en la nueva partición de datos para que se correspondan con los índices particionados de las particiones de datos existentes. Los índices de la tabla fuente que no coinciden con los índices particionados de la tabla de destino se descartan durante el proceso de enlace.

REQUIRE MATCHING INDEXES

Especifica que la tabla fuente debe tener índices que coincidan con los índices particionados de la tabla de destino; de lo contrario, se devuelve un error (SQLSTATE 428GE) y se graba información en la anotación cronológica de administración acerca de los índices que no coinciden.

Si la cláusula `REQUIRE MATCHING INDEXES` no se especifica y los índices de la tabla fuente no coinciden con todos los índices particionados en la tabla de destino, se genera el comportamiento siguiente:

1. En el caso de los índices de la tabla de destino que no tengan una coincidencia en la tabla fuente y sean índices exclusivos o índices XML que se definen con REJECT INVALID VALUES, la operación ATTACH (de enlace) falla (SQLSTATE 428GE).
2. En el caso del resto de índices de la tabla de destino que no tengan una coincidencia en la tabla fuente, el objeto de índice de la tabla fuente se marca como no válido durante la operación de enlace (ATTACH). Si la tabla fuente no tiene índices, se crea un objeto de índice vacío y se marca como no válido. La operación ATTACH será satisfactoria, pero el objeto de índice en la partición de datos nuevos se marca como no válido. Normalmente, SET INTEGRITY es la operación siguiente que se debe ejecutar en la partición de datos. SET INTEGRITY forzará una reconstrucción, si es necesario, del objeto de índice en las particiones de datos que se han enlazado recientemente. La reconstrucción de índice puede aumentar el tiempo obligatorio para poner los datos nuevos en línea.
3. En las anotaciones cronológicas de administración se graba información acerca de los índices que no coinciden.

DETACH PARTITION *nombre-partición INTO nombre1-tabla*

Desenlaza la partición de datos *nombre-partición* de la tabla alterada y utiliza la partición de datos para crear una nueva tabla denominada *nombre-tabla1*. La partición de datos se desenlaza de la tabla alterada y se utiliza para crear la nueva tabla sin ningún movimiento de datos. La partición de datos especificada no puede ser la última partición restante de la tabla modificada (SQLSTATE 428G2). La tabla que se está modificando para desconectar una partición no debe ser una tabla temporal de período de sistema (SQLSTATE 428HZ).

Cuando se desenlaza una tabla para la que se ha definido el control de acceso de nivel de fila o un control de acceso de nivel de columna, la tabla nueva que se crea para los datos desenlazados tendrá automáticamente el control de acceso de nivel de fila (aunque no el control de acceso de nivel de columna) activado para proteger los datos desenlazados. El acceso directo a esta tabla nueva no devolverá ninguna fila hasta que se definan los permisos de fila adecuados para la tabla o se desactive el control de acceso de nivel de fila para la tabla.

ADD SECURITY POLICY *nombre-política*

Añade una política de seguridad a la tabla. La política de seguridad debe existir en el servidor actual (SQLSTATE 42704). La tabla no puede tener ya una política de seguridad (SQLSTATE 55065) y no debe ser una tabla con tipo (SQLSTATE 428DH), una tabla de consulta materializada (MQT) o una tabla de etapas (SQLSTATE 428FG).

DROP SECURITY POLICY

Elimina la política de seguridad y toda la protección LBAC de la tabla. La tabla especificada por *nombre-tabla* debe estar protegida mediante una política de seguridad (SQLSTATE 428GT). Si la tabla tiene una columna con el tipo de datos DB2SECURITYLABEL, el tipo de datos se cambia a VARCHAR (128) FOR BIT DATA. Si la tabla tiene una o varias columnas protegidas, esas columnas pasarán a estar desprotegidas.

ADD VERSIONING USE HISTORY TABLE *nombre-tabla-histórica*

Especifica que la tabla es una tabla temporal de período de sistema. La tabla no debe estar ya definida como tabla temporal de período de sistema o tabla histórica (SQLSTATE 428HM). La tabla debe tener definidos un período

ALTER TABLE

SYSTEM_TIME y una columna de ID de inicio de transacción (SQLSTATE 428HM). La tabla no debe ser una tabla de consulta materializada (SQLSTATE 428HM).

El gestor de bases de datos retiene las versiones históricas de las filas de la tabla. El gestor de bases de datos registra información adicional que indica cuando se ha insertado una fila en la tabla y cuando se ha actualizado o suprimido. Cuando una fila de una tabla temporal de período de sistema se actualiza, se mantiene una versión anterior de la fila. Cuando se suprime una tabla temporal de período de sistema, la versión anterior de la fila se inserta como registro histórico. Se utiliza un historial asociado para almacenar filas históricas de la tabla.

Las referencias a la tabla pueden incluir una condición de búsqueda de período de tiempo para indicar las versiones del sistema de los datos que se deben devolver. *nombre-tabla-histórica* identifica una tabla histórica donde se conservan las filas históricas de la tabla temporal de período de sistema. *nombre-tabla-histórica* debe identificar una tabla que exista en el servidor actual (SQLSTATE 42704) y no sea una tabla de catálogo (SQLSTATE 42832), una tabla temporal de período de sistema existente, una tabla histórica existente, una tabla temporal global declarada, una tabla temporal global creada, una tabla de consulta materializada o una vista (SQLSTATE 428HX).

La tabla histórica identificada no debe contener una columna de identidad, una columna de indicación de fecha y hora de cambio de fila, una columna de inicio de fila, una columna de fin de fila, una columna de ID de inicio de transacción o una columna de expresión generada, ni incluir un período (SQLSTATE 428HX).

La tabla temporal de período de sistema y la tabla histórica identificada deben tener el mismo número y el mismo orden de columnas (SQLSTATE 428HX). Los siguientes atributos para las columnas correspondientes de las dos tablas deben ser los mismos (SQLSTATE 428HX):

- Nombre de columna
- Tipo de datos de columna
- Longitud de columna (incluidas las longitudes de LOB en línea), precisión y escala
- Atributo FOR BIT de columna para columnas de serie de caracteres
- Atributo nulo de columna
- Atributo oculto de columna

Si el control de acceso a filas o el control de acceso a columnas está activado para la tabla temporal de período de sistema y el control de acceso a filas no está activado en la tabla histórica, el gestor de bases de datos activa automáticamente el control de acceso a filas en la tabla histórica y crea un permiso de fila por omisión para la tabla histórica.

DROP VERSIONING

Especifica que la tabla ya no es una tabla temporal de período de sistema. La tabla debe ser una tabla temporal de período de sistema (SQLSTATE 428HZ). Los datos históricos ya no se registran ni se mantienen para la tabla. La definición de las columnas y los datos de la tabla no cambian, pero la tabla ya no se trata como tabla temporal de período de sistema. El período SYSTEM_TIME se retiene. Las consultas posteriores que hacen referencia a la tabla no deben especificar una especificación de período SYSTEM_TIME para

la tabla. La relación entre la tabla temporal de período de sistema y la tabla histórica asociada se elimina. La tabla histórica no se descarta y el contenido de la tabla histórica no se ve afectado.

Cuando se altera una tabla con DROP VERSIONING, se invalidan todos los paquetes con el tipo de dependencia de tabla temporal de período de sistema de esa tabla. Otros objetos dependientes como vistas y activadores que registran una dependencia en la tabla también se marcan como no válidos.

Normas

- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de distribución, si existe (SQLSTATE 42997).
- Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).
- Únicamente se puede hacer referencia a una columna en una cláusula ADD, ALTER o DROP COLUMN de una sola sentencia ALTER TABLE (SQLSTATE 42711).
- No se puede modificar una longitud de columna, un tipo de datos un atributo oculto, ni se puede descartar la columna, si la tabla tiene alguna tabla de consulta materializada dependiente de la tabla (SQLSTATE 42997).
- 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).
- No se puede modificar una longitud de columna si la tabla tiene alguna vista habilitada para la optimización de consulta dependiente de la tabla (SQLSTATE 42997).
- La tabla debe colocarse en estado pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF (SQLSTATE 55019) antes de:
 - Añadir una columna con una expresión de generación
 - Modificar la expresión generada de una columna
 - Cambiar una columna para que contenga una expresión generada
- No se puede modificar una columna existente para establecerla como de tipo DB2SECURITYLABEL (SQLSTATE 42837).
- La definición de una columna de tipo DB2SECURITYLABEL falla si la tabla no tiene asociada una política de seguridad (SQLSTATE 55064).
- No se puede modificar ni descartar una columna de tipo DB2SECURITYLABEL (SQLSTATE 42817).
- Una operación ALTER TABLE para marcar una tabla como protegida falla si hay una MQT dependiente de esa tabla (SQLSTATE 55067).
- El enlace de una partición a una tabla particionada protegida falla si la tabla fuente y la tabla de destino no están protegidas con la misma política de seguridad, no tienen la misma columna de etiqueta de seguridad de fila y no tienen el mismo conjunto de columnas protegidas (SQLSTATE 428GE).
- Si se hace referencia a una columna generada en una clave de particionamiento de tabla, no puede modificarse la expresión de columna generada (SQLSTATE 42837).
- La *cláusula-isolation* no puede especificarse en la selección completa (*fullselect*) de la *definición-consulta-materializada* (SQLSTATE 42601).
- La adición o el enlace de una partición de datos a una tabla particionada no se ejecutarán correctamente y se generará el error SQL0612N tras desenlazar el

mismo nombre de partición si la limpieza de índices asíncrona no ha completado la supresión de entradas de índice para la partición (SQLSTATE 42711).

Notas

- Se ha producido una *operación de REORG recomendada* cuando los cambios obtenidos de una sentencia ALTER TABLE afectan al formato de fila de los datos. En este caso, la mayoría de las operaciones posteriores de la tabla quedan restringidas hasta que se ejecuta correctamente una operación de reorganización de la tabla. Pueden ejecutarse hasta tres sentencias ALTER TABLE de este tipo para una tabla antes de que deba realizarse la reorganización (SQLSTATE 57016). Pueden realizarse varias modificaciones que constituirían una operación de REORG recomendada en una única sentencia ALTER TABLE (una por columna); esto se considera una sola operación de REORG recomendada. Por ejemplo, descartar dos columnas en una única sentencia ALTER TABLE no se considera dos operaciones de REORG recomendada. Sin embargo, si descartara dos columnas en dos sentencias ALTER TABLE independientes, se consideraría que son dos sentencias que contienen operaciones de REORG recomendada.
- Las operaciones de tabla siguientes están permitidas tras realizarse correctamente una operación de REORG recomendada:
 - ALTER TABLE (sin necesidad de validación de los datos de fila) Sin embargo, las operaciones siguientes no están permitidas (SQLSTATE 57007):
 - ADD CHECK CONSTRAINT
 - ADD REFERENTIAL CONSTRAINT
 - ADD UNIQUE CONSTRAINT
 - ALTER COLUMN SET NOT NULL
 - DROP TABLE
 - RENAME TABLE
 - REORG TABLE
 - TRUNCATE TABLE
 - Acceso de exploración de tabla de datos de tabla
- Al modificar una tabla para convertirla en una tabla de consulta materializada, la tabla se coloca en estado pendiente de establecer integridad. Si la tabla se define como REFRESH IMMEDIATE, la tabla se debe sacar del estado pendiente de establecer integridad para poder invocar los mandatos INSERT, DELETE o UPDATE en la tabla a la que hace referencia la selección completa. La tabla se puede sacar del estado pendiente de establecer integridad 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 pendiente de establecer integridad.
- La modificación de una tabla con el fin de convertirla en una tabla de consulta materializada REFRESH IMMEDIATE dará lugar a que se invalide cualquier paquete sujeto a la utilización de INSERT, DELETE o UPDATE en la tabla a la que hace referencia la selección completa.
- La modificación de una tabla con el fin de convertir una tabla de consulta materializada en una tabla normal dará lugar a que se invalide cualquier paquete que dependa de la tabla.
- La modificación de una tabla de consulta materializada MAINTAINED BY FEDERATED_TOOL para que sea una tabla normal no efectuará ningún cambio en la configuración de suscripción de la herramienta de duplicación. Puesto que

un cambio posterior en una tabla de consulta materializada MAINTAINED BY SYSTEM hará que falle la herramienta de duplicación, deberá cambiar el valor de suscripción al cambiar una tabla de consulta materializada MAINTAINED BY FEDERATED_TOOL.

- Si una tabla de consulta materializada diferida se ha asociado a una tabla de etapas, la tabla de etapas se descartará si la tabla de consulta materializada se modifica con el fin de convertirla en una tabla normal.
- Las cláusulas ADD se procesan antes que todas las demás cláusulas. Las demás cláusulas se procesan en el orden en que se especifican.
- Las columnas añadidas mediante una operación de modificación de tabla no se añadirán automáticamente a ninguna vista existente de la tabla.
- La adición o el enlace de una partición de datos a una tabla particionada, o la eliminación del enlace de una partición de datos con una tabla particionada, da lugar a la invalidación de los paquetes que dependen de esa tabla.
- En DB2 Versión 9.7 Fixpack 1 y releases posteriores, tras haber desenlazado una partición de datos de una tabla particionada de datos, el valor STATUS de la partición desenlazada en el catálogo SYSCAT.DATAPARTITIONS puede ser 'L' cuando la partición se ha desenlazado lógicamente y la operación de desenlace no se ha completado. Si el valor STATUS de la partición desenlazada es 'L', en la tabla de origen no podrán realizarse las operaciones que se indican a continuación (SQLSTATE 55057):
 - Adición de una restricción de clave exclusiva o primaria que intenta crear un índice no particionado.
 - Adición, descarte o cambio de nombre de una columna.
 - Activación de la compresión de valor u operación comprimir
 - Desactivación de la compresión de valor u operación comprimir
- Para eliminar el particionamiento de una tabla, es necesario eliminar la tabla y volver a crearla.
- Para eliminar la organización de una tabla, es necesario eliminar la tabla y volver a crearla.
- 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 si 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 fecha y hora.
- Cuando se crea un índice no particionado en una tabla particionada con particiones de datos enlazadas, el índice no incluirá los datos de las particiones de datos enlazadas. Utilice la sentencia SET INTEGRITY para mantener todos los índices de todas las particiones de datos enlazadas.
- Al crear un índice particionado cuando existen particiones de datos enlazadas (el valor STATUS es 'A' en SYSCAT.DATAPARTITIONS), también se creará una partición de índice para cada partición de datos enlazada. Si el índice particionado se desea crear como índice exclusivo, o si es un índice XML que se desea crear con REJECT INVALID VALUES, la creación de índices podría no ejecutarse correctamente si la partición de datos enlazada contiene violaciones (duplicados para un índice exclusivo o valores no válidos para el índice XML).
- Si una tabla tiene un índice no particionado, no puede acceder a una nueva partición de datos de esa tabla dentro de la misma transacción, como la

ALTER TABLE

operación para añadir o enlazar que ha creado la partición, si la transacción no tiene la tabla bloqueada en modalidad exclusiva (SQLSTATE 57007).

- 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 activadores 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 activadores ejecutados por el paquete en nombre de una de sus sentencias.
- En un sistema federado, se puede modificar una tabla base remota que se ha creado utilizando un DDL transparente. Sin embargo, el DDL transparente impone algunos límites en las modificaciones que pueden efectuarse:
 - Una tabla base remota sólo se puede modificar añadiendo nuevas columnas o especificando una clave primaria.
 - Entre las clases específicas que admite el DDL transparente se encuentran las siguientes:
 - ADD COLUMN *definición-columna*
 - NOT NULL y PRIMARY KEY en la cláusula *opciones-columna*
 - ADD *restricción-unicidad* (solo PRIMARY KEY)
 - No se puede especificar un comentario en una columna existente de una tabla base remota.
 - No se puede modificar ni descartar una clave primaria existente en una tabla base remota.
 - La modificación de una tabla base remota invalida cualquier paquete que dependa del apodo asociado a esa tabla base remota.
 - La fuente de datos remota debe dar soporte a los cambios que se piden con la sentencia ALTER TABLE. Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.
 - El intento de modificar una tabla base remota que no se ha creado utilizando un DDL transparente devuelve un error.
- Los cambios (implícitos o explícitos) en la clave primaria, las claves exclusivas o las claves foráneas pueden tener los efectos siguientes en los paquetes, los índices y las demás claves foráneas.
 - Si se añade una clave primaria o una clave exclusiva:
 - Ello no afecta a los paquetes, a las claves foráneas ni a las claves exclusivas existentes. (Si la clave primaria o exclusiva utiliza un índice exclusivo existente que se ha creado en una versión anterior y que no se ha convertido para dar soporte a la unicidad diferida, el índice se convierte y se invalidan los paquetes sujetos a la utilización de UPDATE en la tabla asociada.)
 - Si se descarta una clave primaria o una clave exclusiva:
 - El índice se descarta 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 descartan 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, descarta o altera una clave primaria de NOT ENFORCED a ENFORCED (o de ENFORCED a NOT ENFORCED):
 - Se invalidan todos los paquetes que están 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.
 - Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.
- Si se modifica una clave foránea o una dependencia funcional ENABLE QUERY OPTIMIZATION por DISABLE QUERY OPTIMIZATION:
 - Se invalidan todos los paquetes que dependen de la restricción para la realización de la optimización.
- 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 que no está en estado pendiente de establecer integridad, o la modificación de la restricción de comprobación o de referencia existente de NOT ENFORCED a ENFORCED en una tabla existente que no está en estado pendiente de establecer integridad hará que las filas existentes de la tabla se evalúen inmediatamente para la restricción. Si la verificación falla, se devuelve un error (SQLSTATE 23512). Si una tabla está en estado pendiente de establecer integridad, la adición de una restricción de comprobación o de referencia o la modificación de una restricción de NOT ENFORCED a ENFORCED no dará lugar a la aplicación inmediata de la restricción. Emita la sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED para iniciar la aplicación de la restricción.
- La adición, modificación o descarte de una restricción de comprobación dará lugar a la invalidación de todos los paquetes sujetos a inserción en la tabla de objetos, sujetos a actualización en como mínimo una de las columnas implicadas en la restricción o sujetos a selección que utilicen la restricción para mejorar el rendimiento.
- La adición de una clave de distribución invalida todos los paquetes sujetos a actualización en, como mínimo, una de las columnas de la clave de distribución.
- Una clave de distribución que esté definida por omisión como la primera columna de la clave primaria no se ve afectada por el descarte de la clave primaria y la adición de una clave primaria diferente.
- Al descartar una columna o cambiar su tipo de datos, se elimina toda la información de RUNSTATS de la tabla modificada. Se deberá ejecutar

ALTER TABLE

RUNSTATS en la tabla una vez que se pueda acceder a ella nuevamente. El perfil estadístico de la tabla se conserva si la tabla no contiene una columna descartada explícitamente.

- Al modificar una columna (para cambiar su longitud, tipo de datos, capacidad de nulos o atributo oculto) o descartar una columna, se invalidan todos los paquetes que hacen referencia a su tabla (directa o indirectamente a través de una restricción referencial o un activador).
- Al modificar una columna (para cambiar su longitud, tipo de datos, capacidad de nulos o atributo oculto), se regeneran las vistas (excepto las vistas con tipo) que dependen de su tabla. Si se produce un problema al volver a generar una vista de este tipo, se devuelve un error (SQLSTATE 56098). Las vistas con tipo dependientes de la tabla se marcan como no operativas.
- Al modificar una columna (para cambiar su longitud, tipo de datos o atributo oculto), se marcan todos los activadores dependientes y funciones SQL como no válidos; estos se vuelven a compilar implícitamente la próxima vez que se utilizan. Si se produce un problema al volver a generar un objeto de este tipo, se devuelve un error (SQLSTATE 56098).
- Al modificar una columna (para cambiar su longitud, tipo de datos o atributo de capacidad de nulos), se pueden producir errores (SQLSTATE 54010) al procesar un activador o una función SQL si hay una sentencia en la que interviene el activador o la función SQL preparada o vinculada. Puede producirse este problema si el tamaño de fila basado en la suma de las longitudes de las variables de transición y las columnas de tabla de transición es excesivo. Si se descarta este activador o esta función SQL, un posterior intento de volver a crear este objeto devolverá un error (SQLSTATE 54040).
- A partir de DB2 Versión 9.7 Fixpack 1, un supervisor de sucesos de actividades de WLM creado en una versión anterior debe descartarse y crearse de nuevo para añadir nuevas columnas de tabla incorporadas por este fixpack y cualquiera de los fixpacks o releases posteriores.
- La modificación de una columna de tipo XML o estructurado con el fin de incrementar la longitud en línea invalidará todos los paquetes que hacen referencia a la tabla, ya sea directa o indirectamente por medio de un activador o de una restricción de referencia.
- La modificación de una columna de tipo XML o estructurado con el fin de incrementar la longitud en línea volverá a generar las vistas que dependen de la tabla.
- Se puede crear un diccionario de compresión para el objeto de almacenamiento XML de una tabla sólo si las columnas XML se añaden a la tabla en DB2 Versión 9.7 o una versión posterior o si la tabla se migra mediante un movimiento de tabla en línea.
- Cambiar LOCKSIZE para una tabla dará como resultado una invalidación de todos los paquetes que dependan de la tabla alterada.
- Al cambiar VOLATILE o NOT VOLATILE CARDINALITY se obtendrá como resultado una invalidación de todos los paquetes que tienen una dependencia en la tabla modificada.
- **Duplicación:** tenga precaución cuando aumente la longitud o cambie el tipo de datos de una columna. La tabla de datos de cambio asociada a una tabla de aplicación podría estar en el límite de tamaño de fila de DB2 o próximo a él. La tabla de datos de cambio se debe modificar antes de modificar la tabla de aplicación, o se deben modificar las dos tablas dentro de la misma unidad de trabajo, para asegurarse de que la modificación puede completarse para ambas tablas. Debe prestarse atención a las copias, que también pueden estar en el

límite de tamaño de fila, o cerca de él, o residir en plataformas que no permiten aumentar la longitud de una columna existente.

Si la tabla de datos de cambio no se modifica antes de que el programa Capture procese los registros de anotación cronológica con los atributos modificados, es probable que el programa Capture falle. Si una copia que contiene la columna modificada no se modifica antes de que se ejecute la suscripción que mantiene la copia, es probable que la suscripción falle.

- Al desenlazar una partición de una tabla protegida, la tabla de destino creada automáticamente por DB2 quedará protegida del mismo modo que la tabla fuente.
- Cuando se modifica una tabla de modo que queda protegida con granularidad en el nivel de fila, se invalidan las secciones de SQL dinámico de la antememoria que dependen de esa tabla. Del mismo modo, también se invalidan los paquetes que dependen de esa tabla.
- Cuando se modifica una columna de una tabla, T, de modo que se convierte en una columna protegida, se invalidan las secciones de SQL dinámico de la antememoria que dependen de la tabla T. Del mismo modo, también se invalidan los paquetes que dependen de la tabla T.
- Cuando se modifica una columna de una tabla, T, de modo que se convierte en una columna no protegida, se invalidan las secciones de SQL dinámico de la antememoria que dependen de la tabla T. Del mismo modo, también se invalidan los paquetes que dependen de la tabla T.
- Para las filas existentes en la tabla, el valor de la columna de etiqueta de seguridad se establece por omisión en la etiqueta de seguridad correspondiente al acceso de grabación del ID de autorización de sesión en el momento de ejecutarse la sentencia ALTER que añade una columna de etiqueta de seguridad de fila.
- **Añadir consulta materializada:** Cuando se modifica una tabla base para convertirla en una tabla de consulta materializada, los atributos de seguridad de control de acceso basado en etiquetas (política de seguridad, etiquetas de seguridad de columna, columna de etiqueta de seguridad de fila) se derivan del mismo modo que al crear una tabla de consulta materializada nueva. Si la tabla base modificada ya tiene atributos de seguridad de control de acceso basado en etiquetas, estos atributos intervienen en el proceso de derivación del siguiente modo:
 - Control de acceso a columnas: La etiqueta de seguridad existente para una columna se agrega con la etiqueta de seguridad correspondiente derivada de la consulta que define la tabla de consulta materializada.
 - Control de acceso a filas: Los atributos de control de acceso a filas se establecen exactamente del mismo modo que para una tabla de consulta materializada nueva.
- En DB2 Versión 9.7 FixPack 1 o releases posteriores, los nuevos índices de bloque de tabla de clúster multidimensional (MDC) se particionan. La adición de una partición de datos a una tabla de clúster multidimensional (MDC) particionada de datos crea las correspondientes particiones de índice vacías para la nueva partición, incluidos los índices de bloque MDC. Asimismo, se añade una nueva entrada de partición de índice a SYSCAT.SYSINDEXPARTITIONS para cada índice de bloque MDC y para cada índice particionado.
- Al enlazar una partición de datos con una tabla MDC particionada creada con DB2 V9.7 Fixpack 1 o releases posteriores, la tabla de origen que especifica *partición-enlace* puede ser una tabla MDC no particionada o una única tabla MDC particionada con una sola partición.

ALTER TABLE

- **Si la tabla de origen es una tabla no particionada:** los índices de bloque MDC de la tabla de origen se heredarán y se convertirán en los índices MDC particionados para la nueva partición después de haberse completado la operación ATTACH.
- **Si la tabla de origen es una tabla particionada:** si la tabla de origen es una tabla MDC particionada creada con DB2 V9.7 Fixpack 1 o releases posteriores, los índices de bloque se particionarán. Los índices de bloque se convertirán en los nuevos índices de bloque de la partición.
- Si la tabla MDC particionada fuente se ha creado en un nivel anterior a DB2 V9.7 Fixpack 1, los índices de bloque de la tabla no se particionarán. Durante la operación ATTACH, los índices de bloque se descartan y se crean como índices particionados similares a los otros índices de la tabla de origen.
Será necesario emitir la sentencia SET INTEGRITY en la tabla de destino para establecer la partición enlazada en estado en línea.
Si se especifica la cláusula REQUIRE MATCHING INDEXES y la tabla de destino es una tabla MDC particionada creada en DB2 V9.7 Fixpack 1 o releases posteriores, la sentencia ALTER TABLE ... ATTACH PARTITION no se ejecutará correctamente y devolverá SQL20307N (SQLSTATE 428GE). La eliminación de la cláusula REQUIRE MATCHING INDEXES permitirá que el proceso de enlace pueda continuar.

Si la tabla MDC particionada de destino se ha creado en un nivel anterior a DB2 V9.7 Fixpack 1, los índices de bloque no se particionarán. Los índices de bloque de la tabla MDC de origen se descartarán durante la operación ATTACH. Será necesario emitir la sentencia SET INTEGRITY en la tabla de destino para establecer la partición enlazada en el estado en línea. Se añaden nuevas filas de la partición conectada a los índices de bloque no particionados existentes.

- Al desenlazar una partición de datos de una tabla MDC particionada de datos creada en un nivel anterior a DB2 V9.7 Fixpack 1, los índices de bloque no se particionarán. Se aplican las siguientes restricciones:
 - El acceso a la tabla que acaba de desenlazarse no está permitido en la misma unidad de trabajo que la operación de desenlace.
 - Los índices de bloque de la tabla de destino, creados como parte de la operación de desenlace, vuelven a crearse al tener lugar el primer acceso a la tabla después de haberse confirmado la operación de desenlace. Si la tabla de origen tenía algún índice particionado antes de la operación de desenlace, el objeto de índice de la tabla de destino se marcará como no válido para permitir la recreación de los índices de bloque. Como resultado de ello, el tiempo de acceso se incrementará mientras tiene lugar la recreación de los índices de bloque y de todos los demás índices particionados.

Al desenlazar una partición de una tabla MDC particionada creada mediante la utilización de DB2 V9.7 Fixpack 1 o releases posteriores, los índices de bloque se particionarán, y no tendrán aplicación las restricciones anteriores. El acceso a la tabla que acaba de desenlazarse estará permitido en la misma unidad de trabajo siempre que no exista ningún otro objeto dependiente, como las MQT dependientes. Todos los índices particionados, incluidos los índices de bloque, se convertirán en los índices de la tabla de destino, sin necesidad de recrearlos.

- **Consideraciones para columnas ocultas implícitamente:** Se puede hacer referencia de manera explícita a una columna definida como oculta implícitamente en una sentencia ALTER TABLE. Por ejemplo, se puede modificar o especificar una columna implícitamente oculta como parte de una restricción de referencia, una restricción de comprobación o una definición de tabla de consulta materializada.

La modificación de una tabla para establecer algunas de sus columnas como ocultas implícitamente puede afectar al comportamiento de los programas de utilidad de movimiento de datos que funcionan con la tabla. Cuando una tabla de base de datos contiene columnas ocultas implícitamente, algunos programas de utilidad como IMPORT, INGEST y LOAD requieren que especifique si los datos para las columnas ocultas han de incluirse en la operación. Por ejemplo, esto podría significar que una operación de carga que se ejecutaba correctamente antes de que se modificara la tabla, ahora podría no ejecutarse de forma correcta (SQLCODE SQL2437N). De forma similar, EXPORT requiere que especifique si los datos para las columnas ocultas han de incluirse en la operación.

Los programas de movimiento de datos utilizan la variable de registro DB2_DMU_DEFAULT, o los modificadores de tipo de archivo

implicitlyhiddeninclude o **implicitlyhiddenmissing**, al trabajar con tablas que contienen columnas ocultas implícitamente.

- **Control de acceso a filas que se activa de forma explícita:** la cláusula ACTIVATE ROW ACCESS CONTROL se utiliza para activar el control de acceso a filas para una tabla. Cuando esto sucede, se crea de forma implícita un permiso de fila por omisión y no se permite el acceso a ninguna fila de la tabla, a menos que lo permita un permiso de fila de forma explícita creado por el administrador de seguridad. El permiso de fila por omisión siempre está habilitado.

Cuando se hace referencia a la tabla en una sentencia de manipulación de datos, el gestor de bases de datos aplica de forma implícita todos los permisos de fila habilitados que se hayan creado para la tabla, incluido el permiso de fila por omisión, con el fin de controlar las filas de la tabla. De la aplicación del operador OR lógico a la condición de búsqueda en cada permiso de fila se deriva una condición de búsqueda de control de acceso a filas. Esta condición de búsqueda derivada actúa como filtro para la tabla antes de que se procese cualquier operación especificada por el usuario, como pueden ser predicados, agrupaciones y ordenaciones entre otras. Esta condición de búsqueda derivada permite que los ID de autorización que se especifiquen en las definiciones de permiso accedan a ciertas filas de la tabla.

Cuando se utiliza la cláusula ACTIVATE ROW ACCESS CONTROL, todos los paquetes y las sentencias colocadas en antememoria de forma dinámica que hacen referencia a la tabla quedan invalidados.

El control de acceso a filas permanece en vigor hasta que la cláusula se utiliza la cláusula DEACTIVATE ROW ACCESS CONTROL para que deje de aplicarse.

- **Objeto implícito que se crea cuando se activa el control de acceso a filas para una tabla:** cuando se utiliza la cláusula ACTIVATE ROW ACCESS CONTROL para activar el control de acceso a filas para una tabla, el gestor de bases de datos crea de forma implícita un permiso de fila por omisión para la tabla. El permiso de fila por omisión impide cualquier acceso a la tabla. El permiso de fila creado de forma implícita reside en el mismo esquema de la tabla base y tiene un nombre con el formato SYS_DEFAULT_ROW_PERMISSION__nombre-tabla ... y una longitud máxima de 128 caracteres. Observe que, tras "PERMISSION", se especifican dos caracteres de subrayado. Si este nombre no es exclusivo, los cuatro últimos caracteres se reservan para un número exclusivo 'nnnn', donde 'nnnn' son cuatro caracteres alfanuméricos que empiezan en '0000' y van aumentando de uno en uno cada vez que se encuentra un nombre exclusivo.

El propietario del permiso de fila por omisión es SYSIBM. El permiso de fila por omisión siempre está habilitado. El permiso de fila por omisión se descarta cuando se desactiva el control de acceso a filas o cuando se descarta la tabla.

- **Activación del control de acceso a columnas:** la cláusula ACTIVATE COLUMN ACCESS CONTROL se utiliza para activar el control de acceso de nivel de

columna para una tabla. El acceso a la tabla no se restringe, pero cuando la tabla se referencia en una sentencia de manipulación de datos, todas las máscaras de columna habilitadas que se han creado para la tabla se aplican para enmascarar los valores de columna referenciados en la tabla resultante final.

Cuando se utilizan máscaras de columna para enmascarar valores de columna, determinan los valores de la tabla resultante final. Si una columna tiene una máscara de columna y la columna (específicamente una referencia simple a un nombre de columna o una columna incorporada en una expresión) aparece en la lista de selección más externa, la máscara de columna se aplica a la columna para generar los valores para la tabla de resultados final. Si la columna no aparece en la lista de selección más externa, pero participa en la tabla resultante final, por ejemplo, aparece en una expresión de tabla materializada o en una vista, la máscara de columna se aplica a la columna de manera que el valor enmascarado se incluye en la tabla resultante de la expresión de tabla materializada o la vista para que se pueda utilizar en la tabla resultante final.

La aplicación de las máscaras de columna no interfieren con las operaciones de las demás cláusulas dentro de la sentencia, como puede ser WHERE, GROUP BY, HAVING, SELECT DISTINCT y ORDER BY. Las filas devueltas en la tabla de resultados final permanecen iguales, salvo que puede que las máscaras de columna hayan enmascarado los valores de las filas resultantes. De este modo, si la columna enmascarada también aparece en una *clave-clasificación* ORDER BY, el orden se basa en los valores de columna originales y puede que los valores enmascarados de la tabla de resultados final no reflejen ese orden. Asimismo, es posible que los valores enmascarados no reflejen la exclusividad aplicada por SELECT DISTINCT.

Se aplica una máscara de columna en los contextos siguientes:

- La(s) cláusula(s) SELECT más externa(s) de una sentencia SELECT o SELECT INTO, o si la columna no aparece en la lista de selección más externa, pero participa en la tabla resultante final, las cláusulas SELECT más externas de la expresión de tabla materializada o la vista correspondiente donde aparece la columna.
- La(s) cláusula(s) SELECT más externa(s) de una operación SELECT FROM INSERT, SELECT FROM UPDATE o SELECT FROM DELETE.
- La(s) cláusula(s) SELECT más externa(s) que se utiliza(n) para derivar valores nuevos para una sentencia INSERT, UPDATE o MERGE o una sentencia de asignación de *nombre-variable-transición* SET. Se aplica la misma máscara a una expresión de selección completa escalar que aparece en la(s) cláusula(s) SELECT más exterior(es) de las sentencias mencionadas anteriormente, al lado derecho de una sentencia de asignación de *variable-lenguaje-principal* SET, a la sentencia VALUES INTO o a la sentencia VALUES.

Las máscaras de columna no se aplican cuando la columna enmascarada aparece en los contextos siguientes:

- Cláusulas WHERE.
- Cláusulas GROUP BY.
- Cláusulas HAVING.
- SELECT DISTINCT.
- Cláusulas ORDER BY.
- *El control de acceso a filas y columnas no se aplica cuando se cumplimentan las tablas de EXPLAIN:* El control de acceso a filas y columnas puede aplicarse a las tablas de EXPLAIN. No obstante, los permisos de fila habilitados y las máscaras de columna no se aplican cuando el gestor de bases de datos inserta filas en estas tablas.

- **El control de acceso a filas y columnas no se aplica cuando se cumplimentan las tablas del supervisor de sucesos:** El control de acceso a filas y columnas puede aplicarse a las tablas del supervisor de sucesos. No obstante, los permisos de fila habilitados y las máscaras de columna no se aplican cuando el gestor de bases de datos inserta filas en estas tablas.
- **El control de acceso a filas y columnas no se aplica cuando se cumplimentan las tablas históricas temporales:** El control de acceso a filas y columnas puede aplicarse a las tablas históricas temporales. No obstante, los permisos de fila habilitados y las máscaras de columna no se aplican cuando el gestor de bases de datos accede a esas tablas para las operaciones en las tablas temporales de período de sistema.
- **Detención de la aplicación del control de acceso a filas o columnas:** La cláusula DEACTIVATE ROW ACCESS CONTROL se utiliza para detener la aplicación del control de acceso a filas de una tabla. El permiso de fila por omisión se descarta. En adelante, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, no se aplicarán los permisos de fila creados de forma explícita.

La cláusula DEACTIVATE COLUMN ACCESS CONTROL se utiliza para detener la aplicación del control de acceso a columnas para una tabla. En adelante, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, no se aplicarán las máscaras de columna.

Los permisos de fila o las máscaras de columna creados de forma explícita, si los hay, permanecen pero no tienen efecto.

Todos los paquetes y las sentencias colocadas en antememoria de forma dinámica que hacen referencia a la tabla se invalidan cuando se desactiva el control de acceso a filas o columnas.

- **Activadores seguros para el control de acceso a filas y columnas:** los activadores se utilizan para la integridad de las bases de datos y, por ello, es necesario un equilibrio entre el control de acceso a filas y columnas (seguridad) y la integridad de las bases de datos. Los permisos de fila habilitados y las máscaras de columna no se aplican a los valores iniciales de las variables de transición y las tablas de transición. El control de acceso a filas y columnas aplicado para la tabla de activación también se pasa por alto para las variables de transición o las tablas de transición a las que se hace referencia en el cuerpo del activador. Para que no exista ningún obstáculo de seguridad para las sentencias de SQL en la acción que el activador realiza para acceder a datos confidenciales en las variables de transición y las tablas de transición, el activador debe crearse o modificarse con la opción SECURED. Si un activador no es seguro, no se puede aplicar el control de acceso a filas y columnas para la tabla de activación (SQLSTATE 55019).
- **Funciones seguras definidas por el usuario para el control de acceso a filas y columnas:** si una definición de permiso de fila o máscara de columna hace referencia a una función definida por el usuario, la función debe modificarse con la opción SECURED porque es posible que los datos confidenciales se pasen como argumentos a la función. Cuando se hace referencia a una función definida por el usuario en una sentencia de manipulación de datos donde se hace referencia a una tabla que aplica el control de acceso a filas o columnas y cuando los argumentos de la función hacen referencia a las columnas de esa tabla, si la función no es segura, esto afecta a la selección del plan de acceso y puede provocar que el rendimiento sea bajo. El gestor de bases de datos considera la opción SECURED como una aserción que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. Se presupone que este tipo de procedimiento de

auditoría de control se está aplicando y que este proceso de auditoría revisa todas las sentencias ALTER FUNCTION o cambios posteriores en los paquetes externos.

- **Operaciones de base de datos donde el control de acceso a filas y columnas no es aplicable:** el control de acceso a filas y columnas no debe comprometer la integridad de la base de datos. Las columnas implicadas en las claves primarias, claves exclusivas, índices, restricciones de comprobación e integridad referencial no deben estar sujetas al control de acceso a filas y columnas. Se pueden definir máscaras de columna para esas columnas, pero no se aplican durante el proceso de creación de claves, restricción o aplicación de RI.
- **Definición de una tabla temporal de período de sistema:** Una definición de tabla temporal de período de sistema incluye los aspectos siguientes:
 - Un período de sistema denominado SYSTEM_TIME que se define mediante una columna de inicio de fila y una columna de fin de fila. Consulte las descripciones de AS ROW BEGIN, AS ROW END y de definición de período.
 - Una columna de ID de inicio de transacción. Consulte la descripción de AS TRANSACTION START ID.
 - Una definición de mantenimiento de versiones de datos de período de sistema especificada en una sentencia ALTER TABLE posterior utilizando la acción ADD VERSIONING que incluye el nombre de la tabla histórica asociada. Consulte la descripción de la cláusula ADD VERSIONING de ALTER TABLE.

Para asegurarse de que la tabla histórica no se pueda descartar de forma implícita cuando se descarte una tabla temporal de período del sistema, utilice la cláusula WITH RESTRICT ON DROP en la definición de la tabla histórica.

- **Definición de una tabla temporal de período de aplicación:** Una definición de tabla temporal de período de aplicación incluye un período de aplicación cuyo nombre es BUSINESS_TIME. El período de aplicación se define mediante una columna de inicio y una columna de fin que se caracterizan porque ambas columnas tienen el mismo tipo de datos, DATE o TIMESTAMP(p). Consulte la descripción de definición de período.

Las operaciones de cambio de datos en una tabla temporal de período de aplicación pueden generar una inserción automática de una o dos filas adicionales cuando se actualiza o se suprime una fila. Cuando se especifica una actualización o una supresión de una fila en una tabla temporal de período de aplicación para una parte del período que representa esa fila, la fila se actualiza o se suprime y se insertan una o dos filas de forma automática para representar la parte de la fila que no cambia. Se generan nuevos valores para cada columna generada en una tabla temporal de período de aplicación para cada fila que se inserta de forma automática como resultado de una operación de actualización o supresión en la tabla. Si se define una columna generada como parte de una clave exclusiva o primaria, una clave principal de una restricción referencial o un índice exclusivo, es posible que una inserción automática viole una restricción o índice, en cuyo caso se devolverá un error.

- **Consideraciones sobre las columnas de ID de inicio de transacción:** Una columna de ID de inicio de transacción contiene un valor nulo si la columna soporta valores nulos y existe una columna de inicio de fila y el valor de dicha columna es único entre los valores de las columnas de inicio de fila generadas para otras transacciones. Puesto que la columna puede contener valores nulos, se recomienda que se utilice uno de los métodos siguientes cuando se recupere un valor de la columna:
 - COALESCE (col_ID_inicio_transacción, col_inicio_fila)

- CASE WHEN col_ID_inicio_transacción IS NOT NULL THEN col_ID_inicio_transacción ELSE col_inicio_fila END
- **Consideraciones sobre tablas temporales de período de sistema y control de acceso a filas y a columnas:** El control de acceso a filas y a columnas puede definirse tanto en la tabla temporal de período de sistema como en la tabla histórica asociada.
 - Al acceder a una tabla temporal de período de sistema, cualquier norma de acceso a filas y a columnas definida en la tabla temporal de período de sistema se aplica a todas las filas devueltas desde la tabla temporal de período de sistema, independientemente de si las filas están almacenadas en la tabla temporal de período de sistema o en la tabla histórica. Las normas de acceso a filas y a columnas definidas en la tabla histórica no se aplican.
 - Cuando se accede directamente a la tabla histórica, se aplican las normas de acceso a filas y a columnas definidas en la tabla histórica.

Cuando se define una tabla temporal de período de sistema y se activa el control de acceso a filas o el control de acceso a columnas para la tabla temporal de período de sistema, el gestor de bases de datos activa automáticamente el control de acceso a filas en la tabla histórica y crea un permiso de fila por omisión para la tabla histórica.

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - La palabra clave ADD es opcional para lo siguiente:
 - Restricciones PRIMARY KEY sin nombre
 - Restricciones de referencia sin nombre
 - Restricciones de referencia cuyo nombre aparezca a continuación de la frase FOREIGN KEY
 - La palabra clave CONSTRAINT puede omitirse en una *definición-columna* que defina a una cláusula de referencias
 - El *nombre-restricción* puede especificarse a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT)
 - SET SUMMARY AS puede especificarse en lugar de SET MATERIALIZED QUERY AS
 - SET MATERIALIZED QUERY AS DEFINITION ONLY puede especificarse en lugar de DROP MATERIALIZED QUERY
 - SET MATERIALIZED QUERY AS (selección completa) puede especificarse en lugar de ADD MATERIALIZED QUERY (selección completa)
 - Se puede especificar ADD PARTITIONING KEY en lugar de ADD DISTRIBUTE BY HASH (en este caso también se puede especificar la cláusula USING HASHING opcional)
 - Se puede especificar DROP PARTITIONING KEY en vez de DROP DISTRIBUTION
 - Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos y no son recomendables, en especial para aplicaciones portátiles
 - Puede utilizarse una coma para separar varias opciones en la cláusula *modificación-identidad*
 - Se puede especificar PART en lugar de PARTITION
 - Se puede especificar VALUES en lugar de ENDING AT

ALTER TABLE

- Se puede especificar NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER en lugar de NO MINVALUE, NO MAXVALUE, NO CYCLE, NO CACHE y NO ORDER, respectivamente
- Se puede especificar DROP EXPRESSION en lugar de DROP GENERATED para descartar el atributo de expresión generada de una columna.
- Se puede especificar DROP IDENTITY en lugar de DROP GENERATED para descartar el atributo de identidad de una columna.

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 bytes. 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:* Suponga que hay una tabla denominada EQUIPMENT definida con las columnas siguientes:

Nombre de columna	Tipo de 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 30.000 dólares, por ejemplo.

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT REVENUE
CHECK (SALARY + COMM > 30000)
```

- *Ejemplo 5:* Modifique la tabla EMPLOYEE. Descarte 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 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 de fuente de MEASURE no es BLOB ni un tipo de datos de indicación 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 indicación de fecha y hora.

- *Ejemplo 9:* Se ha definido una tabla denominada CUSTOMERS con las siguientes columnas:

Nombre de columna	Tipo de 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. Para distribuir la tabla, tendrá que crear una clave de distribución para la tabla. La tabla deberá definirse en un espacio de tablas de un grupo de particiones de base de datos de un único nodo. La clave primaria debe ser un superconjunto de las columnas de clave de distribución: como mínimo una de las columnas de la clave primaria debe utilizarse como clave de distribución. Establezca BRANCH_NO como clave de distribución tal como se indica a continuación:

```
ALTER TABLE CUSTOMERS
ADD DISTRIBUTE BY HASH (BRANCH_NO)
```

- *Ejemplo 10:* Se ha creado la tabla remota EMPLOYEE en un sistema federado utilizando un DDL transparente. Altere la tabla remota EMPLOYEE para añadir las columnas PHONE_NO y WORK_DEPT; añada, también, una clave primaria en la columna existente EMP_NO y la nueva columna WORK_DEPT.

```
ALTER TABLE EMPLOYEE
ADD COLUMN PHONE_NO CHAR(4) NOT NULL
ADD COLUMN WORK_DEPT CHAR(3)
ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

ALTER TABLE

- *Ejemplo 11:* Modifique la tabla DEPARTMENT para añadir la dependencia funcional FD1 y descarte la dependencia funcional FD1 de la tabla DEPARTMENT.

```
ALTER TABLE DEPARTMENT
ADD CONSTRAINT FD1
CHECK ( DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED
```

```
ALTER TABLE DEPARTMENT
DROP CHECK FD1
```

- *Ejemplo 12:* Cambie el valor por omisión de la columna WORKDEPT de la tabla EMPLOYEE por 123.

```
ALTER TABLE EMPLOYEE
ALTER COLUMN WORKDEPT
SET DEFAULT '123'
```

- *Ejemplo 13:* Asocie la política de seguridad DATA_ACCESS con la tabla EMPLOYEE.

```
ALTER TABLE EMPLOYEE
ADD SECURITY POLICY DATA_ACCESS
```

- *Ejemplo 14:* Modifique la tabla EMPLOYEE para proteger la columna SALARY.

```
ALTER TABLE EMPLOYEE
ALTER COLUMN SALARY
SECURED WITH EMPLOYEESECLABEL
```

- *Ejemplo 15:* Suponga que tiene una tabla denominada SALARY_DATA que está definida con las columnas siguientes:

Nombre columna	Tipo de datos
-----	-----
EMP_NAME	VARCHAR(50) NOT NULL
EMP_ID	SMALLINT NOT NULL
EMP_POSITION	VARCHAR(100) NOT NULL
SALARY	DECIMAL(5,2)
PROMOTION_DATE	DATE NOT NULL

Cambie esta tabla para que se puedan almacenar los salarios en una columna DECIMAL(6,2), establezca PROMOTION_DATE como campo opcional que se puede definir en un valor nulo y elimine la columna EMP_POSITION.

```
ALTER TABLE SALARY_DATA
ALTER COLUMN SALARY SET DATA TYPE DECIMAL(6,2)
ALTER COLUMN PROMOTION_DATE DROP NOT NULL
DROP COLUMN EMP_POSITION
```

- *Ejemplo 16:* Añada una columna denominada DATE_ADDED a la tabla BOOKS. El valor por omisión de esta columna es la indicación de fecha y hora actual.

```
ALTER TABLE BOOKS
ADD COLUMN DATE_ADDED TIMESTAMP
WITH DEFAULT CURRENT_TIMESTAMP
```

- *Ejemplo 17:* Convierta una tabla con atributos de seguridad de control de acceso basado en etiquetas en una tabla de consulta materializada. Las tablas base tt1 y tt2 existen y se crearon con el siguiente SQL:

```
CREATE TABLE tt1
(c1 INT SECURED WITH C, c2 DB2SECURITYLABEL) SECURITY POLICY P;
CREATE TABLE tt2
(c3 INT SECURED WITH B, c4 DB2SECURITYLABEL) SECURITY POLICY P;
```

La tabla tt2 se puede convertir en una tabla de consulta materializada con el siguiente SQL:

```
ALTER TABLE tt2 ADD (SELECT * FROM tt1 WHERE c1 > 10)
DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

La tabla tt2 se convierte en una tabla de consulta materializada con política de seguridad P. tt2.c3 tiene la etiqueta de seguridad P.B. tt2.c4 tiene la etiqueta de seguridad P.C y también es DB2SECURITYLABEL.

ALTER TABLESPACE

La sentencia ALTER TABLESPACE se utiliza para modificar un espacio de tablas existente.

Un espacio de tablas puede modificarse de las formas siguientes:

- Añadir un contenedor a un espacio de tablas DMS o descartarlo del mismo; es decir, un espacio creado con la opción MANAGED BY DATABASE.
- Modificar el tamaño de un contenedor en un espacio de tablas DMS.
- Reducir la marca de límite superior de un espacio de tablas DMS mediante el traslado de extensiones.
- Añadir un contenedor a un espacio de tablas SMS en una partición de base de datos que actualmente no tiene contenedores.
- 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.
- Modificar la política de almacenamiento en antememoria del sistema de archivos para un espacio de tablas.
- Habilitar o deshabilitar el ajuste de tamaño automático para un espacio de tablas de almacenamiento automático o DMS.
- Volver a equilibrar un espacio de tablas de almacenamiento automático de tamaño normal o grande.
- Modificar el valor DATA TAG para un espacio de tablas.
- Modificar un espacio de tablas DMS a un espacio de tablas de almacenamiento automático.
- Modificar el valor STOGROUP asociado a un espacio de tablas.

Invocación

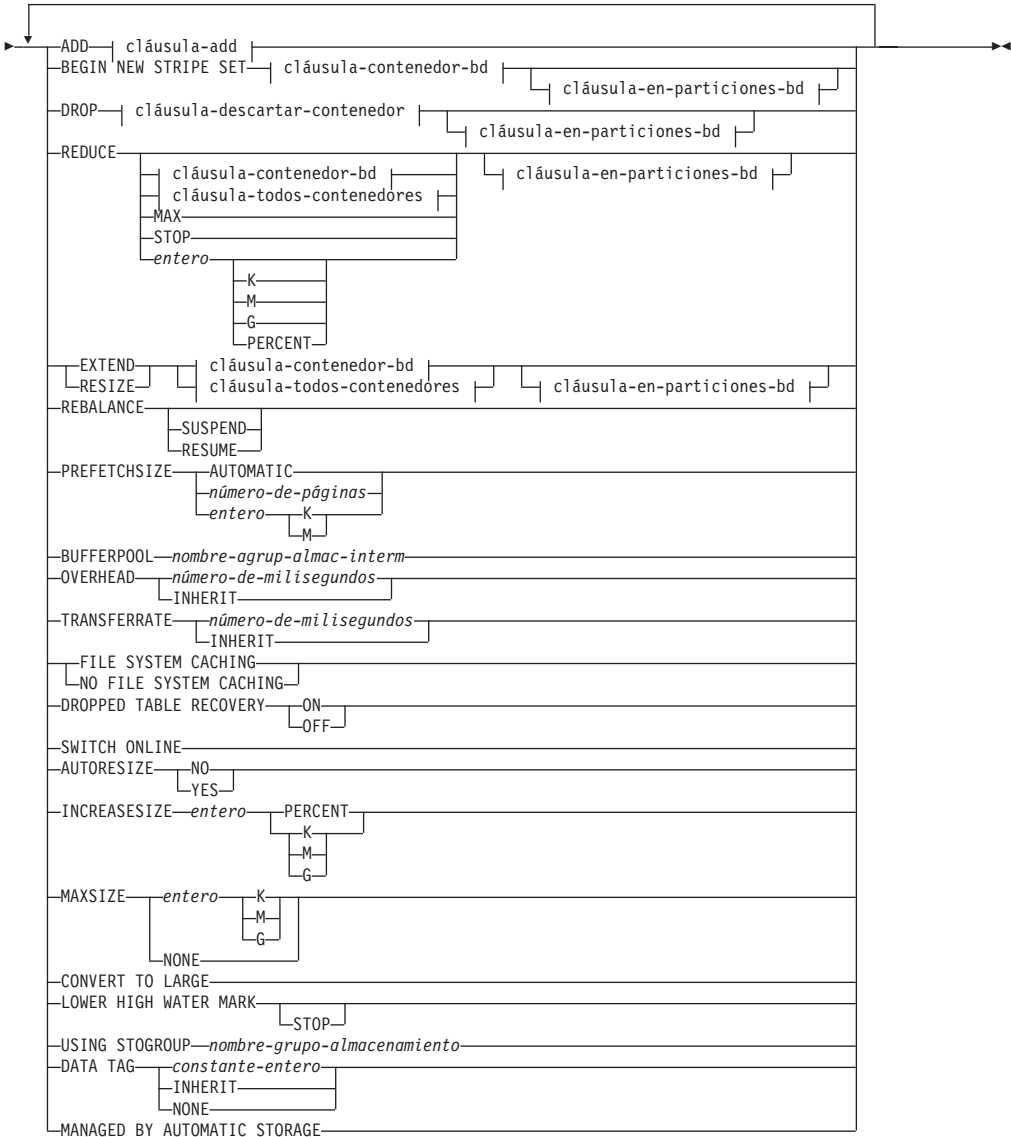
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

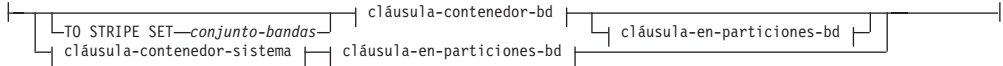
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis

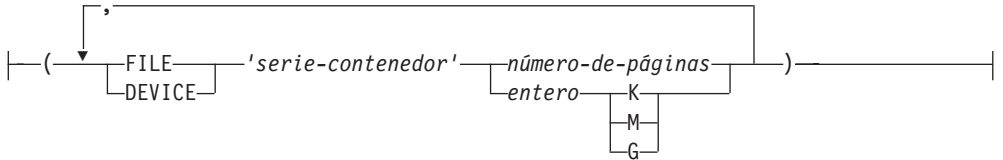
▶▶ALTER TABLESPACE—*nombre-espacio-tablas*—————▶



cláusula-add:

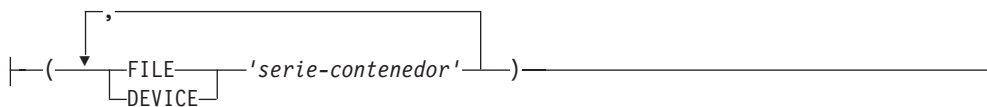


cláusula-contenedor-bd:



cláusula-descartar-contenedor:

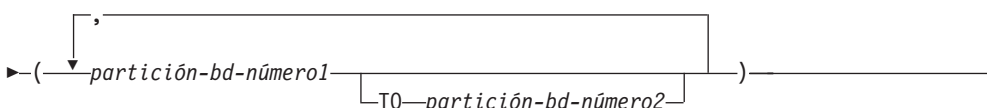
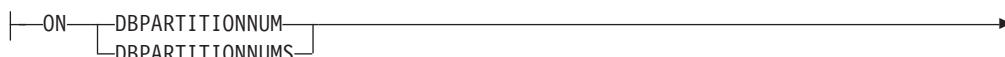
ALTER TABLESPACE



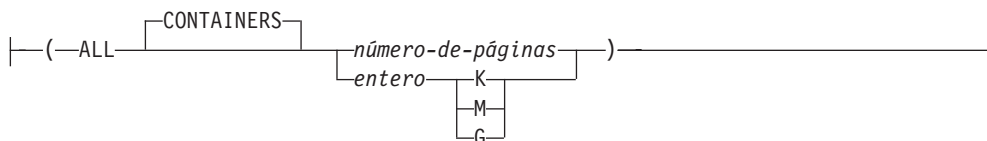
cláusula-contenedor-sistema:



cláusula-en-particiones-bd:



cláusula-todos-contenedores:



Descripción

nombre-espacio-tablas

Nombra el espacio de tablas. Este nombre consta de una sola parte. Es un identificador de SQL largo (ordinario o delimitado).

ADD

Especifica que se deben añadir uno o varios contenedores nuevos al espacio de tablas.

TO STRIPE SET *conjunto-bandas*

Especifica que se añadan uno o más contenedores al espacio de tablas y que se colocarán en el conjunto de bandas especificado.

BEGIN NEW STRIPE SET

Especifica que se debe crear un conjunto de bandas nuevo en el espacio de tablas, y que se deben añadir uno o varios contenedores a este conjunto de bandas nuevo. Los contenedores que se añadan posteriormente mediante la opción ADD se añadirán a este nuevo conjunto de bandas a menos que se especifique TO STRIPE SET.

DROP

Especifica que se deben descartar uno o varios contenedores del espacio de tablas.

REDUCE

Para espacios de tablas de almacenamiento no automático, especifica que va a reducirse el tamaño de los contenedores existentes. El tamaño que se especifica es el tamaño que se reducirá el contenedor existente. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas disminuirán de tamaño. Si la reducción del tamaño da como resultado un tamaño de espacio de tablas que es inferior a la marca de límite superior actual, se intentará reducir la marca de límite superior antes de intentar reducir los contenedores. Para espacios de tablas de almacenamiento no automático, la cláusula REDUCE debe ir seguida de una *cláusula-contenedor-bd* o una *cláusula-todos-contenedores*.

Para espacios de tablas de almacenamiento automático, especifica que, si se puede, hay que reducir la marca de límite superior actual, y que hay que reducir el tamaño del espacio de tablas según la nueva marca de límite superior. Para espacios de tablas de almacenamiento automático, la cláusula REDUCE no debe ir seguida de una *cláusula-contenedor-bd*, una *cláusula-todos-contenedores* o una *cláusula-en-particiones-bd*.

Nota: La opción REDUCE con las cláusulas MAX, valor numérico, PERCENT o STOP y la opción LOWER HIGH WATER MARK con la cláusula STOP sólo están disponibles para espacios de tablas gestionados por base de datos y por almacenamiento automático que contienen el atributo de almacenamiento reclamable. Además, estas opciones se deben especificar y ejecutar de forma independiente al resto de opciones, incluso cada una de ellas con respecto al resto.

La cláusula **MAX**, **STOP**, *entero* [**K** | **M** | **G**] o *entero* **PERCENT** entra en vigor cuando la sentencia se procesa y no se retrotrae si la unidad de trabajo, donde se ejecuta la sentencia, se retrotrae.

cláusula-contenedor-bd

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.

cláusula-todos-contenedores

Amplía, reduce o cambia el tamaño de todos los contenedores de un espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

MAX

Para los espacios de tablas de almacenamiento automático con almacenamiento reclamable, especifica que el número máximo de extensiones debería trasladarse al principio del espacio de tablas para reducir la marca de límite superior. Además, el tamaño del espacio de tablas disminuirá hasta la nueva marca de límite superior. Esto no se aplica a los espacios de tablas de almacenamiento no automático.

STOP

Para los espacios de tablas de almacenamiento automático con almacenamiento reclamable, interrumpe la operación de traslado de extensiones si se está ejecutando. Esta opción no está disponible para espacios de tablas de almacenamiento no automático.

entero [**K** | **M** | **G**] o *entero* **PERCENT**

Para los espacios de tablas de almacenamiento automático con

ALTER TABLESPACE

almacenamiento reclamable, especifica el valor numérico en el que se reducirá el espacio de tablas gracias al traslado de extensiones. El valor se puede expresar de varias formas:

- Un entero especificado sin K, M, G o PERCENT indica que el valor numérico es el número de páginas en el que se va a reducir el espacio de tablas.
- Un entero especificado con K, M o G indica el tamaño de la reducción en kilobytes, megabytes o gigabytes, respectivamente. El valor en bytes se convierte antes al número total de páginas en función del tamaño de página del espacio de tablas.
- Un entero especificado con PERCENT indica el número de extensiones que se van a trasladar, como porcentaje del tamaño actual del espacio de tablas.

Una vez finalizado el traslado de extensiones, el tamaño del espacio de tablas disminuye hasta la nueva marca de límite superior. Esta opción no está disponible para espacios de tablas de almacenamiento no automático.

cláusula-en-particiones-bd

Especifica una o más particiones de base de datos para las operaciones del contenedor correspondiente.

EXTEND

Especifica que va a aumentarse 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-todos-contenedores*, todos los contenedores del espacio de tablas aumentarán de tamaño.

RESIZE

Especifica que va a cambiarse el tamaño de los contenedores existentes. El tamaño especificado es el nuevo tamaño del contenedor. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas se cambiarán a este tamaño. Si la operación afecta a más de un contenedor, deberá aumentarse o disminuirse el tamaño de todos esos contenedores. No es posible aumentar el tamaño de algunos de ellos y, al mismo tiempo, reducir el de otros (SQLSTATE 429BC).

cláusula-contenedor-bd

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.

cláusula-descartar-contenedor

Descarta uno o varios 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.

cláusula-contenedor-sistema

Añade uno o más contenedores a un espacio de tablas SMS de las particiones de base de datos especificadas. 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 de base de datos especificadas para el espacio de tablas (SQLSTATE 42921).

cláusula-en-particiones-bd

Especifica una o más particiones de base de datos para las operaciones del contenedor correspondiente.

cláusula-todos-contenedores

Amplía, reduce o cambia el tamaño de todos los contenedores de un espacio

de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

REBALANCE

Para los espacios de tablas de almacenamiento automático de tamaño normal o grande, inicia la creación de contenedores en las vías de almacenamiento que se acaban de agregar, el descarte de los contenedores ubicados en las vías de acceso de almacenamiento que tienen el estado "Descarte pendiente" o ambas acciones. Durante el reequilibrio, los datos se trasladan a contenedores de vías de acceso nuevas y se sacan de los contenedores de las vías de acceso descartadas. El reequilibrio se ejecuta de forma asíncrona en segundo plano y no influye en la disponibilidad de los datos.

Nota: La cláusula **SUSPEND** o **RESUME** entra en vigor cuando la sentencia se procesa y no se retrotrae si la unidad de trabajo, donde se ejecuta la sentencia, se retrotrae.

SUSPEND

Suspende la operación de reequilibrio activa en el espacio de tablas especificado. Si no existe una operación de reequilibrio activa, no se emprende acción alguna y se devuelve una ejecución satisfactoria. El estado de suspensión es persistente y si la base de datos se desactiva mientras el reequilibrio está suspendido, tras la activación de la base de datos, la operación de reequilibrio se reinicia desde el estado suspendido. Suspende una operación de reequilibrio cuando ya está suspendida no tiene ningún efecto y se devuelve una ejecución satisfactoria.

RESUME

Reanuda una operación de reequilibrio suspendida anteriormente. Si no existe una operación de reequilibrio activa, no se emprende acción alguna y se devuelve una ejecución satisfactoria. Si el reequilibrio está en estado PAUSED (en pausa) a causa de una operación de copia de seguridad en línea, el reequilibrio del espacio de tablas sale del estado suspendido, pero permanece en pausa hasta que finaliza la copia de seguridad en línea.

PREFETCHSIZE

Especifica que han de leerse los datos que una consulta necesita antes de que la consulta haga referencia a éstos con el fin de que la consulta no tenga que esperar a que se produzca la E/S.

AUTOMATIC

Especifica que el tamaño de captación previa de un espacio de tablas debe actualizarse automáticamente, es decir, el tamaño de captación previa se gestionará a través del gestor de bases de datos de DB2.

Una base de datos DB2 actualizará el tamaño de captación previa automáticamente siempre que el número de contenedores de un espacio de tablas cambie (seguido por una ejecución satisfactoria de una sentencia ALTER TABLESPACE que añada o descarte uno o más contenedores). El tamaño de captación previa se actualiza al arrancar la base de datos.

La actualización automática del tamaño de captación previa puede desactivarse especificando un valor numérico en la cláusula PREFETCHSIZE.

número-de-páginas

Especifica el número de páginas PAGESIZE del espacio de tablas que se leerán cuando se realice la captación previa de datos. El valor máximo es 32767.

ALTER TABLESPACE

entero K | M

Especifica los valores de tamaño de captación previa como valor de entero seguido de K (de kilobytes) o de M (de megabytes). Si se especifica de este modo, se utiliza el límite inferior del número de bytes dividido por el tamaño de página para determinar el valor correspondiente al número de páginas para el tamaño de captación previa.

BUFFERPOOL *nombre-agrup-almac-interm*

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 (SQLSTATE 42704). Debe haberse definido el grupo de particiones de base de datos del espacio de tablas para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos* u **OVERHEAD INHERIT**

Especifica la actividad general del controlador de E/S, la búsqueda de disco y el tiempo de latencia. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

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. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y OVERHEAD se hereda de forma dinámica del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42858). Si OVERHEAD se establece en indefinido para el grupo de almacenamiento y OVERHEAD se establece en INHERIT, se utilizará el valor por omisión de creación de base de datos.

En el caso de una base de datos que se ha creado en DB2 V10.1 o posterior, la actividad general del controlador de E/S por omisión y el tiempo de búsqueda y latencia de disco es 6,725 milisegundos.

En el caso de una base de datos que se ha actualizado de una versión anterior de DB2 a DB2 V10.1 o posterior, la actividad general del controlador de E/S y el tiempo de búsqueda y latencia de disco por omisión son los siguientes:

- 7,5 milisegundos para una base de datos creada en DB2 versión 9.1 o posterior
- 12,67 milisegundos para las bases de datos creadas entre DB2 versión 8.2 y DB2 versión 9.1
- 24,1 milisegundos para las versiones DB2 anteriores a 8.2

TRANSFERRATE *número-de-milisegundos* o **TRANSFERRATE INHERIT**

Especifica el tiempo necesario para leer una página en memoria. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

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. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y TRANSFERRATE se hereda de forma dinámica del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42858). Si la DEVICE READ RATE (velocidad de lectura del dispositivo) del grupo de almacenamiento se establece en un valor indefinido y el usuario establece TRANSFERRATE en INHERIT, se utilizará el valor por omisión de creación de base de datos.

Cuando un espacio de tablas de almacenamiento automático hereda el valor TRANSFERRATE del grupo de almacenamiento que utiliza, la DEVICE READ RATE del grupo de almacenamiento, que se expresa en megabytes por segundo, se convierte en la cuenta de milisegundos por página leída para el valor PAGESIZE del espacio de tablas. La fórmula de conversión es la siguiente:

$$\text{TRANSFERRATE} = (1 / \text{DEVICE READ RATE}) * 1000 / 1024000 * \text{PAGESIZE}$$

En el caso de una base de datos que se ha creado en DB2 V10.1 o posterior, el tiempo por omisión para leer una página en memoria para el espacio de tablas PAGESIZE de 4 KB es 0,04 milisegundos.

En el caso de una base de datos que se ha actualizado de una versión anterior de DB2 a DB2 V10.1 o posterior, el tiempo por omisión para leer una página en memoria es el siguiente:

- 0,06 milisegundos para una base de datos creada en DB2 versión 9.1 o posterior
- 0,18 milisegundos para las bases de datos creadas entre DB2 versión 8.2 o versión 9.1
- 0,9 milisegundos para las versiones de DB2 anteriores a la 8.2

FILE SYSTEM CACHING o NO FILE SYSTEM CACHING

Especifica si las operaciones de E/S se almacenarán en antememoria a nivel del sistema de archivos o no. Las conexiones con la base de datos deben terminar para que surta efecto una nueva política de almacenamiento en antememoria. Tenga en cuenta que el acceso de E/S es demasiado largo o que los datos LOB se almacenan de forma intermedia para los contenedores SMS y DMS.

FILE SYSTEM CACHING

Todas las operaciones de E/S del espacio de tablas de destino se almacenarán en antememoria en el nivel del sistema de archivos.

NO FILE SYSTEM CACHING

Todas las operaciones de E/S evitarán el almacenamiento en antememoria a nivel del sistema de archivos.

DROPPED TABLE RECOVERY

Especifica si las tablas que se han descartado del *nombre-espacio-tablas* pueden recuperarse o no utilizando la opción **RECOVER DROPPED TABLE ON** del mandato **ROLLFORWARD DATABASE**. Para tablas particionadas, la recuperación de tablas descartadas siempre está activa, incluso cuando dicha recuperación esté desactivada para las tablas sin particiones en uno o más espacios de tablas.

ON Especifica que las tablas descartadas pueden recuperarse.

OFF

Especifica que las tablas descartadas no pueden recuperarse.

ALTER TABLESPACE

SWITCH ONLINE

Especifica que los espacios de tablas en estado OFFLINE deben ponerse en línea si sus contenedores han pasado a ser accesibles. Si los contenedores no son accesibles, se devolverá un error (SQLSTATE 57048).

AUTORESIZE

Especifica si debe habilitarse o no la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas gestionado por la base de datos (DMS) o de un espacio de tablas de almacenamiento automático. Los espacios de tablas de redimensionamiento automático aumentan automáticamente de tamaño cuando se llenan.

NO Especifica que debe inhabilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático. Si se inhabilita la posibilidad de cambiar automáticamente el tamaño, no se conservará ningún valor que se haya especificado anteriormente para INCREASESIZE o MAXSIZE.

YES

Especifica que debe habilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

INCREASESIZE *entero* PERCENT o INCREASESIZE *entero* K | M | G

Especifica la cantidad, por partición de base de datos, en la que aumentará automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente, cuando se llene el espacio de tablas y se haya efectuado una petición de espacio. El valor entero debe ir seguido de:

- PERCENT para especificar la cantidad como porcentaje del tamaño de espacio de tablas en el momento en el que se efectúe una petición de espacio. Cuando se especifique PERCENT, el valor entero debe estar entre 0 y 100 (SQLSTATE 42615).
- K (de kilobytes), M (de megabytes) o G (de gigabytes) para especificar la cantidad en bytes.

Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior o superior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

MAXSIZE *entero* K | M | G o MAXSIZE NONE

Especifica el tamaño máximo hasta el que se puede aumentar automáticamente un espacio de tablas habilitado para redimensionamiento automático.

entero

Especifica un límite fijo sobre el tamaño, por partición de base de datos, hasta el cual puede aumentar automáticamente un espacio de tablas DMS o un espacio de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

NONE

Especifica que debe permitirse incrementar el espacio de tablas hasta la capacidad del sistema de archivos o hasta el tamaño máximo del espacio de tablas (descrito en "Límites de SQL y XML").

CONVERT TO LARGE

Modifica un espacio de tablas DMS normal existente para que sea un espacio de tablas DMS grande. El espacio de tablas y su contenido se bloquean durante

la conversión. Esta opción sólo puede utilizarse en espacios de tablas DMS normales. Si se especifica un espacio de tablas SMS, un espacio de tablas temporal o un espacio de tablas de catálogo del sistema, se devuelve un error (SQLSTATE 560CF). No es posible convertir un espacio de tablas que contiene una partición de datos de una tabla particionada que tiene particiones de datos en otro espacio de tablas (SQLSTATE 560CF). La conversión no puede invertirse una vez que se ha confirmado. Si las tablas del espacio de tablas están definidas con DATA CAPTURE CHANGES, tenga en cuenta los límites de almacenamiento y de capacidad de la tabla de destino y del espacio de tablas.

LOWER HIGH WATER MARK

Para los espacios de tablas de almacenamiento automático y no automático con almacenamiento reclamable, activa la operación de traslado de extensiones para mover el número máximo de extensiones a una posición inferior del espacio de tablas. Aunque la marca de límite superior disminuye, el tamaño del espacio de tablas no se reduce. A continuación, debe realizarse un ALTER TABLESPACE REDUCE en los espacios de tablas de almacenamiento automático, o un ALTER TABLESPACE REDUCE con la *cláusula-contenedor-bd* o *cláusula-todos-contenedores* en los espacios de tablas de almacenamiento no automático.

Nota: La opción LOWER HIGH WATER MARK con la cláusula STOP y la opción REDUCE con las cláusulas MAX, valor numérico, PERCENT, o STOP sólo están disponibles para los espacios de tablas gestionados por base de datos y por almacenamiento automático que contienen el atributo de almacenamiento reclamable. Además, estas opciones se deben especificar y ejecutar de forma independiente al resto de opciones, incluso cada una de ellas con respecto al resto.

Nota: Esta cláusula entra en vigor cuando la sentencia se procesa y no se retrotrae si la unidad de trabajo, donde se ejecuta la sentencia, se retrotrae.

STOP

Para los espacios de tablas de almacenamiento automático y no automático con almacenamiento reclamable, interrumpe la operación de traslado de extensiones si se está ejecutando.

USING STOGROUP

Asocia un espacio de tablas con otro grupo de almacenamiento. Los datos asociados con el espacio de tablas se moverán de su grupo de almacenamiento actual al grupo de almacenamiento especificado. Esta cláusula sólo se aplica a espacios de tablas de almacenamiento automático a menos que se especifique con la cláusula MANAGED BY AUTOMATIC STORAGE (SQLSTATE 42858).

En el caso de espacios de tablas de almacenamiento automático, se reinicia un REBALANCE implícito durante la confirmación. En el caso de un espacio de tablas gestionado por base de datos que se va a convertir en gestionado por almacenamiento automático, es necesaria una sentencia REBALANCE explícita.

En un entorno de base de datos particionada, para modificar la asociación de grupos de almacenamiento de un espacio de tablas, este debe definirse mediante almacenamiento automático en todas las particiones de bases de datos. Si el espacio de tablas o cualquier partición de base de datos no se define mediante almacenamiento automático, este mandato fallará a menos que se especifique con la cláusula MANAGED BY AUTOMATIC STORAGE (SQLSTATE 42858). Sin embargo, no es necesario que un espacio de tablas tenga la misma asociación de grupo de almacenamiento en todas las

ALTER TABLESPACE

particiones de la base de datos para que este mandato consiga mover el espacio de tablas en todas las particiones de la base de datos.

nombre-grupo-almacenamiento

Identifica el grupo de almacenamiento en el que se almacenarán los datos de espacio de tablas. *nombre-grupo-almacenamiento* debe identificar un grupo de almacenamiento que exista en el servidor actual (SQLSTATE 42704).

Este nombre consta de una sola parte.

DATA TAG *constante-entero*, DATA TAG INHERIT o DATA TAG NONE

Especifica un código para los datos en el espacio de tablas. Este valor se puede utilizar como parte de una configuración de WLM en una definición de clase de trabajo o se puede referenciar en una definición de umbral; para obtener más información, consulte las sentencias CREATE WORK CLASS SET, ALTER WORK CLASS SET, CREATE THRESHOLD y ALTER THRESHOLD. Esta cláusula no se puede especificar para los espacios de tablas USER o SYSTEM TEMPORARY o para el espacio de tablas de catálogo (SQLSTATE 42858).

constante-entera

Los valores válidos para *constante-entera* son los enteros del 1 al 9. Si se especifica una *constante-entera* y existe un grupo de almacenamiento asociado, el código de datos especificado para el espacio de tablas alterará temporalmente cualquier valor de código de datos que se especifique para el grupo de almacenamiento asociado.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y DATA TAG se hereda de forma dinámica del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42858).

NONE

Si se especifica NONE, no hay ningún código de datos.

MANAGED BY AUTOMATIC STORAGE

Permite el almacenamiento automático para un espacio de tablas gestionado por la base de datos (DMS). Cuando el almacenamiento automático está habilitado, no se pueden ejecutar más operaciones de contenedor en el espacio de tablas. El espacio de tablas que se convierte no puede utilizar contenedores RAW (DEVICE).

Si la cláusula USING STOGROUP no se incluye al realizar una conversión de un espacio de tablas DMS a un espacio de tablas de almacenamiento automático, se especifica el grupo de almacenamiento por omisión.

Normas

- La cláusula BEGIN NEW STRIPE SET no puede especificarse en la misma sentencia que ADD, DROP, EXTEND, REDUCE ni RESIZE, a menos que dichas cláusulas se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- El valor de conjunto de bandas especificado con la cláusula TO STRIPE SET se debe encontrar dentro del rango válido para el espacio de tablas que se está modificando (SQLSTATE 42615).
- Al añadir o eliminar espacio del espacio de tablas, se deben seguir las normas siguientes:
 - EXTEND y RESIZE pueden utilizarse en la misma sentencia, siempre que se aumente el tamaño de cada contenedor (SQLSTATE 429BC).

- REDUCE y RESIZE pueden utilizarse en la misma sentencia, siempre que se reduzca el tamaño de cada contenedor (SQLSTATE 429BC).
- EXTEND y REDUCE no pueden utilizarse en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- ADD no puede utilizarse con REDUCE ni DROP en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- DROP no puede utilizarse con EXTEND ni ADD en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- La cláusula AUTORESIZE, INCREASESIZE o MAXSIZE no puede especificarse para los espacios de tablas gestionados por el sistema (SMS), los espacios de tablas temporales creados utilizando el almacenamiento automático ni los espacios de tablas DMS definidos para utilizar contenedores de dispositivo sin procesar (SQLSTATE 42601).
- La cláusula INCREASESIZE o MAXSIZE no puede especificarse si el espacio de tablas no permite cambiar el tamaño automáticamente (SQLSTATE 42601).
- Cuando se especifique un nuevo tamaño máximo para un espacio de tablas, el valor debe ser superior al tamaño actual en cada partición de base de datos (SQLSTATE 560B0).
- Las operaciones con contenedores (ADD, EXTEND, RESIZE, DROP o BEGIN NEW STRIPE SET) no pueden realizarse en espacios de tablas de almacenamiento automático porque el gestor de bases de datos controla la gestión del espacio de dichos espacios de tablas (SQLSTATE 42858).
- No pueden añadirse contenedores de dispositivo sin procesar a un espacio de tablas DMS que permita cambiar el tamaño automáticamente (SQLSTATE 42601).
- La cláusula CONVERT TO LARGE no puede especificarse en la misma sentencia que ninguna otra cláusula (SQLSTATE 429BC).
- La cláusula REBALANCE no puede especificarse con ninguna otra cláusula (SQLSTATE 429BC).
- La cláusula REBALANCE sólo es válida en los espacios de tablas de almacenamiento automático de tamaño normal y grande (SQLSTATE 42601). Los espacios de tablas de almacenamiento automático temporales deben descartarse y recrearse para aprovechar las vías de acceso de almacenamiento añadidas recientemente o para que se puedan eliminar sus contenedores ubicados en las vías de acceso de almacenamiento que se están descartando.
- Las operaciones de contenedor y la cláusula REBALANCE no se pueden especificar si el espacio de tablas tiene el estado “Reequilibrador DMS activo” (SQLSTATE 55041).
- No se puede especificar la cláusula USING STOGROUP para los espacios de tablas temporales (SQLSTATE 42858).
- Las cláusulas siguientes no están soportadas en entornos DB2 pureScale:
 - ADD *cláusula-contenedor-bd*
 - BEGIN NEW STRIPE SET *cláusula-contenedor-bd*
 - DROP *cláusula-contenedor-bd*
 - LOWER HIGH WATER MARK
 - LOWER HIGH WATER MARK STOP
 - REDUCE, a menos que se especifique sin ninguno de los elementos opcionales
 - RESIZE *cláusula-contenedor-bd*

ALTER TABLESPACE

– USING STOGROUP

- Las cláusulas ADD, DROP, RESIZE, EXTEND, REDUCE, LOWER HIGH WATER MARK y BEGIN_STRIPE_SET no se pueden usar en combinación con la cláusula MANAGED BY AUTOMATIC STORAGE o USING STOGROUP (SQLSTATE 429BC).
- La cláusula USING STOGROUP no se puede especificar si el espacio de tablas está en el estado del tipo "el reequilibrador está activo" (SQLSTATE 55041).
- *Límites de tamaño de contenedor*: En espacios de tablas DMS, un contenedor debe tener al menos el doble del tamaño de extensión en páginas de longitud (SQLSTATE 54039). El tamaño máximo de un contenedor depende del sistema operativo.
- *Límite de longitud de definición de contenedor*: Cada definición de contenedor requiere 53 bytes más el número de bytes necesario para almacenar el nombre de contenedor. La longitud combinada de todas las definiciones de contenedores para el espacio de tablas no puede superar los 208 kilobytes (SQLSTATE 54034).

Notas

- Las operaciones de contenedor por omisión son operaciones de contenedor que se han especificado en la sentencia ALTER TABLESPACE, pero que no se han dirigido de forma explícita a una partición de base de datos específica. Estas operaciones de contenedor se envían a cualquier partición de base de datos que no aparezca en la lista de la sentencia. Si estas operaciones de contenedor por omisión no se envían a ninguna partición de base de datos porque todas las particiones de base de datos se han mencionado explícitamente para una operación de contenedor, se devuelve un mensaje de aviso (SQLSTATE 01589).
- Tras añadir o eliminar espacio de un espacio de tablas y confirmar la transacción, se puede reequilibrar el contenido del espacio de tablas entre los contenedores. El acceso al espacio de tablas no está restringido durante el reequilibrio.
- Si el espacio de tablas se encuentra en estado OFFLINE y los contenedores han pasado a ser accesibles, el usuario puede desconectar todas las aplicaciones y volverlas a conectar a la base de datos para que el espacio de tablas salga del estado OFFLINE. De forma alternativa, la opción SWITCH ONLINE puede activar el espacio de tablas (cambiar su estado OFFLINE) mientras el resto de la base de datos sigue activa y se utiliza.
- Si se añade más de un contenedor a un espacio de tablas, se recomienda añadirlos en la misma sentencia para que el reequilibrio sólo deba efectuarse una vez. Si se intenta añadir contenedores al mismo espacio de tablas en sentencias ALTER TABLESPACE diferentes dentro de una sola transacción, se producirá un error (SQLSTATE 55041).
- Si se intenta ampliar, reducir, cambiar el tamaño o descartar contenedores que no existen, se generará un error (SQLSTATE 428B2).
- Cuando se amplía, se reduce o se cambia el tamaño de un contenedor, el tipo de contenedor debe coincidir con el tipo que se ha utilizando durante la creación del contenedor (SQLSTATE 428B2).
- Si se intenta cambiar los tamaños de los contenedores de un mismo espacio de tablas, utilizando sentencias ALTER TABLESPACE diferentes pero en una sola transacción, se producirá un error (SQLSTATE 55041).
- En una base de datos particionada, si más de una partición de base de datos reside en el mismo nodo físico, no podrá especificarse el mismo dispositivo o vía de acceso específica para tales particiones de base de datos (SQLSTATE 42730). Para este entorno, especifique una *serie-contenedor* específica para cada partición o bien utilice un nombre de vía de acceso relativa.

- Aunque la definición del espacio de tablas sea transaccional y los cambios en la definición se reflejen en las tablas del catálogo tras su confirmación, la agrupación de almacenamientos intermedios con la nueva definición no se podrá utilizar 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.
- La opción REDUCE, RESIZE o DROP intenta liberar, si es necesario, extensiones no utilizadas para espacios de tablas DMS y la opción REDUCE intenta liberar extensiones no utilizadas para espacios de tablas de almacenamiento automático. La eliminación de las extensiones no utilizadas permite que la marca de límite superior de espacio de tablas se reduzca al valor que fielmente representa la cantidad de espacio utilizado lo que, a su vez, permite mayores reducciones del tamaño del espacio de tablas.
- *Conversión a espacios de tablas DMS grandes:* tras la conversión, se recomienda emitir la sentencia COMMIT y luego aumentar la capacidad de almacenamiento del espacio de tablas.
 - Si el espacio de tablas está habilitado para el redimensionamiento automático, el valor del atributo de espacio de tablas MAXSIZE debe aumentarse, a menos que ya esté establecido en NONE.
 - Si el espacio de tablas no está habilitado para el redimensionamiento automático:
 - Habilite el redimensionamiento automático emitiendo una sentencia ALTER TABLESPACE con la opción AUTORESIZE YES, o bien
 - Añada más capacidad de almacenamiento añadiendo conjuntos de bandas, ampliando el tamaño de los contenedores existentes, o realizando ambas acciones.

Los índices de las tablas de un espacio de tablas convertido deben reorganizarse o generarse de nuevo para que puedan dar soporte a identificadores de registros de gran tamaño (RID).

- Los índices se pueden volver a crear mediante el mandato **REORG INDEXES ALL** con la opción REBUILD. Especifique la opción **ALLOW NO ACCESS** para las tablas particionadas.
- Las tablas también pueden reorganizarse (no INPLACE), lo que dará lugar a que se generen de nuevo todos los índices y se habiliten las tablas para dar soporte a más de 255 filas por tabla.

Para determinar qué tablas no dan soporte todavía a RID de gran tamaño, utilice la función de tabla ADMIN_GET_TAB_INFO.

- El reequilibrio del espacio de tablas de almacenamiento automático que tiene contenedores en una vía de acceso de almacenamiento con estado “Descarte pendiente” descartará estos contenedores. Podría ser necesario crear contenedores nuevos para contener los datos que se están retirando de los contenedores descartados. Debe haber espacio libre suficiente en las otras vías de acceso de almacenamiento de la base de datos para que puedan crearse dichos contenedores; en caso contrario se devuelve un error SQLSTATE 57011. La cantidad real de espacio libre requerida depende de muchos factores, incluida la ubicación de la extensión de marca de límite superior y los conjuntos de bandas que se están modificando. Sin embargo, para garantizar que la operación se complete satisfactoriamente, la cantidad mínima de espacio libre suficiente en las vías de acceso de almacenamiento restantes debería ser igual que el espacio que están consumiendo los contenedores que se están descartando.

ALTER TABLESPACE

- Si se ha especificado la cláusula REBALANCE pero el servidor de datos determina que no es necesario crear contenedores nuevos o descartar los existentes, no se ejecuta el reequilibrio y la sentencia resulta satisfactoria con un aviso (SQLSTATE 01690).
- Además de añadir contenedores a vías de acceso creadas recientemente, también puede utilizarse la operación REBALANCE para añadir contenedores a las vías de acceso de almacenamiento existentes. Se examinan todos los conjuntos de bandas del espacio de tablas y se identifican las vías de acceso de almacenamiento que no está utilizando un conjunto de bandas concreto. Se creará un contenedor nuevo para cada vía de acceso de almacenamiento identificada, si el espacio libre en ella es suficiente. El tamaño del contenedor será igual al del resto de los contenedores del conjunto de bandas. Esta situación resultaría beneficiosa en el caso de que una vía de acceso de almacenamiento dada se quedara sin espacio, los espacios de tablas dejaran de utilizarla (mediante la creación de conjuntos de bandas en las otras vías de acceso) y se otorgara más almacenamiento a la vía de acceso. En este caso no se han añadido vías de acceso nuevas, pero el reequilibrio intentará incluir esta vía de acceso de almacenamiento en los conjuntos de bandas en los que no se había incluido previamente.
- Pese a todo, el ajuste de tamaño automático puede producirse mientras se está ejecutando el reequilibrio de un espacio de tablas de almacenamiento automático.
- Cuando la cláusula MANAGED BY AUTOMATIC STORAGE habilita un espacio de tablas DMS para el almacenamiento automático, dicho espacio de tablas dispondrá de uno o varios conjuntos de bandas de contenedores de almacenamiento automático y de uno o varios conjuntos de bandas de contenedores (de almacenamiento no automático) definidos por el usuario. Al volver a equilibrar el espacio de tablas (con la cláusula REBALANCE) se eliminan todos los contenedores definidos por el usuario. El gestor de bases de datos podría extender los contenedores de almacenamiento automático existentes o crear otros nuevos para contener los datos que se están moviendo desde los contenedores definidos por el usuario.
- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODE puede especificarse en lugar de DBPARTITIONNUM
 - NODES puede especificarse en lugar de DBPARTITIONNUMS

Ejemplos

- *Ejemplo 1:* Añada un dispositivo al espacio de tablas PAYROLL.

```
ALTER TABLESPACE PAYROLL
ADD (DEVICE '/dev/rhdisk9' 10000)
```

- *Ejemplo 2:* Cambie el tamaño de captación previa y la actividad general de E/S para el espacio de tablas ACCOUNTING.

```
ALTER TABLESPACE ACCOUNTING
PREFETCHSIZE 64
OVERHEAD 19.3
```

- *Ejemplo 3:* Cree el espacio de tablas TS1, luego cambie el tamaño de todos los contenedores a 2000 páginas. (Se muestran tres sentencias ALTER TABLESPACE diferentes que efectuarán 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 todos los contenedores del espacio de tablas DATA_TS en 1000 páginas.

```

ALTER TABLESPACE DATA_TS
  EXTEND (ALL 1000)

```

- *Ejemplo 5:* Cambie el tamaño de todos los contenedores del espacio de tablas INDEX_TS a 100 megabytes (MB).

```

ALTER TABLESPACE INDEX_TS
  RESIZE (ALL 100 M)

```

- *Ejemplo 6:* Añada tres nuevos contenedores. Amplíe el primer contenedor y cambie el tamaño del segundo.

```

ALTER TABLESPACE TS0
  ADD (FILE 'cont2' 2000, FILE 'cont3' 2000)
  ADD (FILE 'cont4' 2000)
  EXTEND (FILE 'cont0' 100)
  RESIZE (FILE 'cont1' 3000)

```

- *Ejemplo 7:* El espacio de tablas TSO existe en las particiones de base de datos 0, 1 y 2. Añada un contenedor nuevo a la partición de base de datos 0. Amplíe todos los contenedores de la partición de base de datos 1. Cambie el tamaño de un contenedor en todas las particiones de bases de datos, excepto en las especificadas de forma explícita (es decir, las particiones de bases de datos 0 y 1).

```

ALTER TABLESPACE TSO
  ADD (FILE 'A' 200) ON DBPARTITIONNUM (0)
  EXTEND (ALL 200) ON DBPARTITIONNUM (1)
  RESIZE (FILE 'B' 500)

```

La cláusula RESIZE es la cláusula de contenedor por omisión en este ejemplo, y se ejecutará en la partición de base de datos 2, ya que otras particiones se envían de forma explícita a las particiones de base de datos 0 y 1. No obstante, si sólo hubieran existido estas dos particiones de base de datos, la sentencia se habría ejecutado correctamente, pero habría devuelto una advertencia (SQL1758W) indicando que los contenedores por omisión se especifican pero no se utilizan.

- *Ejemplo 8:* Habilite la opción de cambiar el tamaño automáticamente para el espacio de tablas DMS_TS1 y establezca su tamaño máximo en 256 megabytes.

```

ALTER TABLESPACE DMS_TS1
  AUTORESIZE YES MAXSIZE 256 M

```

- *Ejemplo 9:* Habilite la opción de cambiar el tamaño automáticamente para el espacio de tablas AUTOSTORE1 y cambie su índice de incremento por el de un 5%.

```

ALTER TABLESPACE AUTOSTORE1
  AUTORESIZE YES INCREASESIZE 5 PERCENT

```

ALTER TABLESPACE

- *Ejemplo 10:* Cambie el índice de incremento por el de 512 kilobytes para un espacio de tablas denominado MY_TS que permite cambiar el tamaño automáticamente, y establezca su tamaño máximo de forma que sea el mayor posible.

```
ALTER TABLESPACE MY_TS  
INCREASESIZE 512 K MAXSIZE NONE
```

- *Ejemplo 11:* habilitar el almacenamiento automático para el espacio de tablas gestionado por una base de datos DMS_TS10 y que utilice el grupo de almacenamiento sg_3.

```
ALTER TABLESPACE DMS_TS10  
MANAGED BY AUTOMATIC STORAGE  
USING STOGROUP sg_3
```

- *Ejemplo 12:* Una sentencia ALTER DATABASE ha suprimido de la base de datos conectada actualmente las vías de acceso /db2/filesystem1 y /db2/filesystem2. Los espacios de tablas denominados PRODTS1, PRODTS2 y PRODTS3 eran los únicos espacios de tablas que utilizaban las vías de acceso suprimidas. Reequilibrar estos espacios de tablas. Se deben utilizar tres sentencias ALTER TABLESPACE.

```
ALTER TABLESPACE PRODTS1 REBALANCE  
ALTER TABLESPACE PRODTS2 REBALANCE  
ALTER TABLESPACE PRODTS3 REBALANCE
```

- *Ejemplo 13:* habilitar el almacenamiento automático para el espacio de tablas gestionado por la base de datos DATA1 y eliminar todos los contenedores de almacenamiento no automático existentes del espacio de tablas. Debe confirmarse la primera sentencia para poder ejecutar la segunda.

```
ALTER TABLESPACE DATA1 MANAGED BY AUTOMATIC STORAGE  
ALTER TABLESPACE DATA1 REBALANCE
```

- *Ejemplo 14:* activar el traslado de extensiones para un espacio de tablas de almacenamiento automático con el atributo de almacenamiento reclamable, para reducir el tamaño de los contenedores en 10 MB.

```
ALTER TABLESPACE DMS_TS1 REDUCE 10 M
```

- *Ejemplo 15:* activar el traslado de extensiones para un espacio de tablas de almacenamiento no automático con el atributo de almacenamiento reclamable y reducir posteriormente el tamaño de cada contenedor en 10 MB.

```
ALTER TABLESPACE TBSP1 LOWER HIGH WATER MARK  
ALTER TABLESPACE TBSP1 REDUCE (ALL CONTAINERS 10 M)
```


ALTER THRESHOLD

La sentencia ALTER THRESHOLD modifica la definición de un umbral.

Invocación

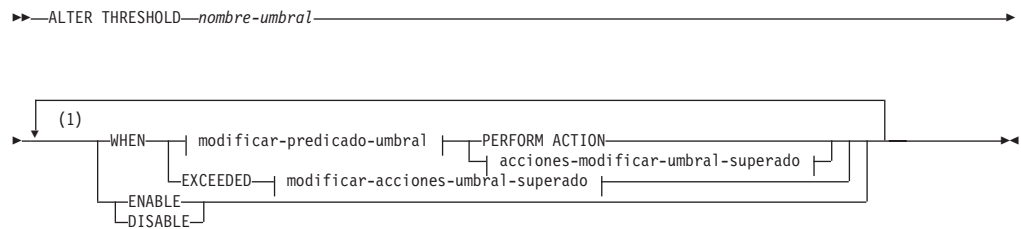
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

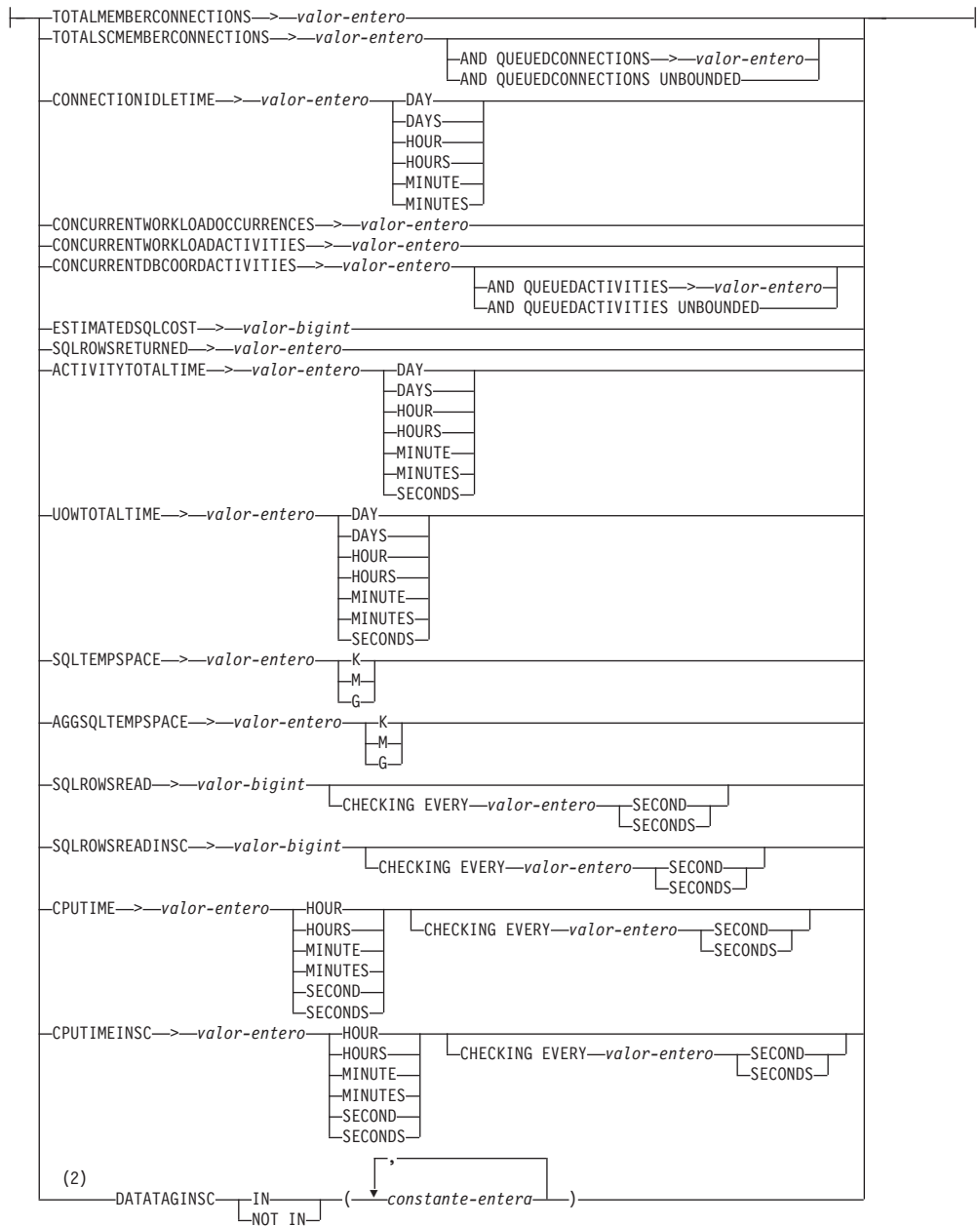
- Autoridad SQLADM, únicamente si todas las cláusulas de modificación son cláusulas COLLECT.
- Autorización WLMADM
- Autorización DBADM

Sintaxis

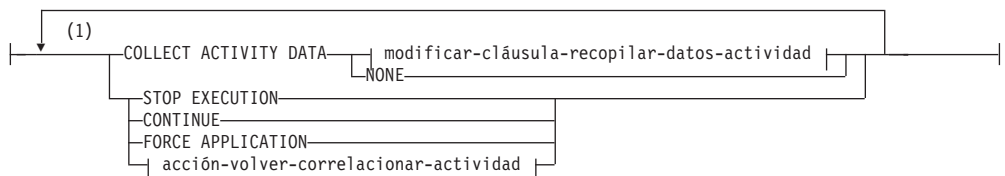


modificar-predicado-umbral:

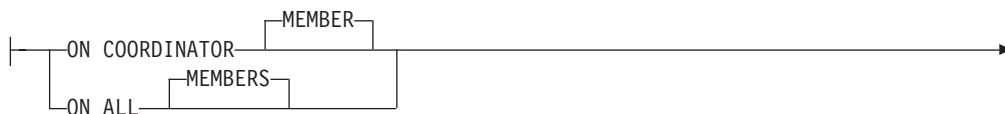
ALTER THRESHOLD

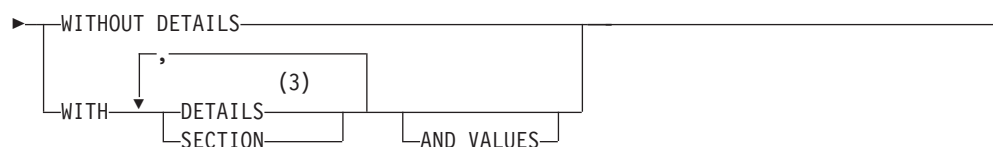


acciones-modificar-umbral-superado:



modificar-cláusula-recopilar-datos-actividad:





acción-volver-correlacionar-actividad:



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.
- 2 Cada valor de etiqueta de datos sólo se puede especificar una vez.
- 3 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-umbral

Identifica el umbral que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre debe identificar exclusivamente un umbral existente en el servidor actual (SQLSTATE 42704).

WHEN *modificar-predicado-umbral* o **WHEN EXCEEDED**

Sustituye el valor de vinculación superior en la condición de predicado de umbral por un nuevo valor de vinculación superior. La condición del umbral no puede cambiarse por una condición diferente.

PERFORM ACTION

Cuando se modifica el valor de la condición de predicado de umbral, especifica que la acción que ha superado el umbral no ha cambiado.

EXCEEDED

Especifica que debe conservarse el mismo predicado de umbral que se especificó en un principio para este umbral modificado.

modificar-predicado-umbral

TOTALMEMBERCONNECTIONS > *valor-entero*

Esta condición define un límite superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en un miembro. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá la conexión de cualquier conexión de coordinador nueva. Proseguirán todas las conexiones en cola o en ejecución en la actualidad.

TOTALSCMEMBERCONNECTIONS > *valor-entero*

Esta condición define un límite superior en el número de conexiones de coordinador que pueden ejecutarse simultáneamente en un miembro de una superclase de servicio específica. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá que cualquier conexión nueva se una a la clase de servicio. Proseguirán todas las conexiones en cola o en ejecución en la actualidad.

ALTER THRESHOLD

AND QUEUEDCONNECTIONS > *valor-entero* o **AND QUEUEDCONNECTIONS UNBOUNDED**

Especifica un tamaño de cola para el momento en que se supere el número máximo de conexiones de coordinador. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna conexión de coordinador. Especificar UNBOUNDED pondrá en cola todas las conexiones que superen el número máximo especificado de conexiones de coordinador y las *acciones-umbral-superado* nunca se ejecutarán.

CONNECTIONIDLETIME > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES

Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá a una conexión permanecer desocupada. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. Esta condición se aplica en el miembro coordinador.

Si se especifica la acción STOP EXECUTION con umbrales CONNECTIONIDLETIME, la conexión para la aplicación se descarta cuando se supera el umbral. Cualquier intento posterior por parte de la aplicación para acceder al servidor de datos no recibirá SQLSTATE 5U026 dado que la aplicación ya no está conectada con el servidor de datos.

El valor máximo para este umbral es 2.147.483.640 segundos. Cualquier valor que se especifique que tenga un número de segundos equivalente a 2.147.483.640 se establecerá en este número de segundos.

CONCURRENTWORKLOADOCCURRENCES > *valor-entero*

Esta condición define un límite superior en el número de actividades de coordinador simultáneas y actividades anidadas para la carga de trabajo en cada miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

CONCURRENTWORKLOADACTIVITIES > *valor-entero*

Esta condición define un límite superior en el número de actividades de coordinador simultáneas y actividades anidadas para la carga de trabajo en cada miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Cada actividad anidada debe satisfacer las condiciones siguientes:

- Debe ser una actividad de coordinador reconocida. No se contará ninguna actividad de coordinador anidada que no caiga dentro de los tipos de actividades reconocidos. De modo análogo, no se contará ninguna actividad de subagente anidado, como por ejemplo las peticiones de nodo remoto.
- Debe invocarse directamente desde la lógica de usuario, como por ejemplo un procedimiento escrito por el usuario que emita sentencias de SQL.

En consecuencia, las actividades coordinadoras anidadas que se han iniciado automáticamente bajo la invocación de un programa de utilidad de DB2 o rutinas de los esquemas SYSIBM, SYSFUN o SYSPROC no cuentan con relación a la vinculación superior especificada mediante este umbral.

Este umbral no cuenta las actividades SQL internas, como por ejemplo las generadas estableciendo una restricción o renovando una tabla de consulta

materializada, ya que son iniciadas por medio del gestor de base de datos y no invocadas directamente mediante la lógica de usuario.

CONCURRENTDBCOORDACTIVITIES > *valor-entero*

Esta condición define un límite superior sobre el número de actividades de coordinador de base de datos reconocidas que pueden ejecutarse simultáneamente en todos los miembros del dominio especificado. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá la ejecución de las actividades de cualquier coordinador de base de datos nueva. Proseguirán todas las actividades de coordinador de base de datos que estén en cola o en ejecución en la actualidad. Esta condición hace el seguimiento de todas las actividades, salvo de los elementos siguientes:

- Este umbral no controla las sentencias CALL, pero todas las actividades hijo anidadas iniciadas dentro de la rutina llamada se someten al control de este umbral. Los bloques anónimos y las rutinas autónomas se clasifican como sentencias CALL.
- Este umbral controla las funciones definidas por el usuario, pero las actividades hijo anidadas en una función definida por el usuario no se controlan. Si se llama a una rutina autónoma desde dentro de una función definida, ni la rutina autónoma ni las actividades hijo de la rutina autónoma están bajo el control del umbral.
- Este umbral no controla las acciones de activador que invocan sentencias CALL y las actividades hijo de esas sentencias CALL. Las sentencias INSERT, UPDATE o DELETE que pueden hacer que un activador se active siguen estando sometidas al control del umbral.

Importante: Antes de utilizar umbrales CONCURRENTDBCOORDACTIVITIES, asegúrese de haberse familiarizado con los efectos que pueden tener en el sistema de bases de datos. Para obtener más información, consulte el tema "Umbral CONCURRENTDBCOORDACTIVITIES".

AND QUEUEDACTIVITIES > *valor-entero* o **AND QUEUEDACTIVITIES UNBOUNDED**

Especifica un tamaño de cola para el momento en que se supere el número máximo de actividades de coordinador de base de datos. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna actividad de coordinador de base de datos. Especificar UNBOUNDED pondrá en cola todas las actividades de coordinador de base de datos que superen el número máximo especificado de actividades de coordinador de base de datos y las *acciones-umbral-superado* nunca se ejecutarán.

Nota: Si se especifica la acción de umbral CONTINUE para un umbral de cola, el tamaño de la cola no queda acotado, independientemente de los valores fijos que se hayan especificado para el tamaño de la cola.

ESTIMATEDSQLCOST > *valor-bigint*

Esta condición define una vinculación superior para el coste asignado de optimizador (en activaciones de temporizador) de una actividad. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). Esta condición se aplica en el miembro coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.

ALTER THRESHOLD

- Las actividades de DML anidado que se invocan desde la lógica del usuario. En consecuencia, esta condición no hace el seguimiento de las actividades de DML (a menos que su coste se incluya en la estimación del padre, en cuyo caso se hace un seguimiento indirecto de las mismas) que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

SQLROWSRETURNED > *valor-entero*

Esta condición define una vinculación superior para el número de filas devueltas a una aplicación cliente desde el servidor de la aplicación. Este valor puede ser cualquier entero positivo (no cero) (SQLSTATE 42820). Esta condición se implanta en el miembro coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML.
- Actividades de DML anidado que derivan de la lógica del usuario. Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

Los conjuntos de resultados devueltos desde un procedimiento se tratan como actividades individuales por separado. Las filas que devuelve el propio procedimiento no se agregan.

ACTIVITYTOTALTIME > *integer-value* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECONDS

Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá que se ejecute una actividad, incluyendo el tiempo que la actividad estuvo en cola. Las actividades que este umbral cubre incluyen la ejecución de sentencias de SQL, sin incluir el tiempo de compilación y el programa de utilidad de carga. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. Esta condición se aplica en el miembro coordinador.

Si la unidad de tiempo especificada es SECONDS, el valor debe ser un múltiplo de 10 (SQLSTATE 42615). El valor máximo que se puede especificar para este umbral es 2.147.483.640 segundos. Cualquier valor especificado (mediante la utilización de la unidad de tiempo DAY, HOUR, MINUTE o SECONDS) que tenga un equivalente en segundos mayor que 2.147.483.640 segundos se truncará en este número de segundos.

UOWTOTALTIME > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECONDS

Esta condición define un límite superior para la cantidad de tiempo que el gestor de bases de datos concederá a la ejecución de una unidad de trabajo. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. Si la unidad de tiempo especificada es SECONDS, el valor debe ser un múltiplo de 10 (SQLSTATE 42615). Esta condición se aplica en el miembro coordinador.

El valor máximo que se puede especificar para este umbral es 2.147.483.640 segundos. Si cualquier valor especificado (mediante la unidad de tiempo DAY, HOUR o MINUTE) tiene un equivalente en segundos superior al valor máximo, se devuelve un error (SQLSTATE 42615).

SQLTEMPSPACE > *valor-entero* K | M | G

Esta condición define la cantidad máxima de espacio temporal del sistema

que una sentencia de SQL puede consumir en un miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Si se especifica *valor-entero* K (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero* M, el tamaño máximo es *valor entero* multiplicado por 1 048 576. Si se especifica *valor-entero* G, el tamaño máximo es *valor entero* multiplicado por 1 073 741 824.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

AGGSQLTEMPSPACE > *valor-entero* K | M | G

Esta condición define la cantidad máxima de espacio temporal del sistema que puede consumir un conjunto de sentencias en una clase de servicio en un miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Si se especifica *valor-entero* K (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero* M, el tamaño máximo es *valor entero* multiplicado por 1.048.576. Si se especifica *valor-entero* G, el tamaño máximo es *valor entero* multiplicado por 1.073.741.824.

Las actividades que contribuyen al agregado cuyo seguimiento efectúa esta condición son:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente, como la ejecución de subsección.
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente, como la ejecución de subsección. Las actividades iniciadas por medio del gestor de bases de datos mediante un programa de utilidad, procedimiento o sentencia de SQL interna no resultan afectadas por esta condición.

SQLROWSREAD > *valor-bigint*

Esta condición define un límite superior en el número de filas que puede leer una actividad durante su vida útil en un miembro concreto. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). Tenga en cuenta que el número de filas leídas es distinto del número de filas devueltas, que se controla mediante la condición SQLROWSRETURNED.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.

ALTER THRESHOLD

- Este umbral tampoco efectúa el seguimiento de actividades SQL internas como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. El umbral se comprueba al final de cada petición (como una operación de captación, por ejemplo) y en el intervalo definido por la cláusula CHECKING. La cláusula CHECKING define una vinculación superior sobre el tiempo durante el que una infracción de umbral puede estar sin detectarse. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

SQLROWSREADINSC >*valor-bigint*

Esta condición define un límite superior en el número de filas que puede leer una actividad en un miembro concreto mientras se ejecuta en una subclase de servicio. Las filas leídas antes de ejecutarse en la subclase de servicio especificada no cuentan. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). Tenga en cuenta que el número de filas leídas es distinto del número de filas devueltas, que se controla mediante la condición SQLROWSRETURNED.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. El umbral se comprueba al final de cada petición (como una operación de captación, por ejemplo) y en el intervalo definido por la cláusula CHECKING. La cláusula CHECKING define una vinculación superior sobre el tiempo durante el que una infracción de umbral puede estar sin detectarse. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

CPUTIME > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS

Esta condición define un límite superior para la cantidad de tiempo de procesador que una actividad puede consumir durante su vida útil en un miembro. El tiempo de procesador cuyo seguimiento realiza este umbral se

mide desde el momento en que la actividad empieza a ejecutarse. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas, como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.
- Actividades de tipo CALL. En el caso de actividades CALL, el tiempo de procesador cuyo seguimiento se haya efectuado para el procedimiento no incluye el tiempo de procesador utilizado por cualquier actividad hija o por cualquier proceso de modalidad no delimitada. La condición de umbral sólo se comprobará cuando la devuelva la lógica de usuario al motor de la base de datos. Por ejemplo: durante la ejecución de una rutina fiable, la condición de umbral sólo se comprobará cuando la rutina emita una petición al motor de la base de datos.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. La granularidad del umbral CPUTIME es aproximadamente este número multiplicado por el grado de paralelismo de la actividad. Por ejemplo: si el umbral se comprueba cada 60 segundos y el grado de paralelismo es 2, la actividad puede utilizar 2 minutos más de tiempo de procesador, en lugar de 1 minuto antes de que se detecte la infracción de umbral. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

CPUTIMEINSC > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS

Esta condición define un límite superior para la cantidad de tiempo de procesador que una actividad puede consumir en un miembro concreto mientras se está ejecutando en una subclase de servicio. El tiempo de procesador cuyo seguimiento realiza este umbral se mide desde el momento en que la actividad empieza a ejecutarse en la subclase de servicio identificada en el dominio de umbral. El tiempo de procesador utilizado con anterioridad a ese punto no cuenta con relación al límite que este umbral impone. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos

mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.

- Este umbral tampoco efectúa el seguimiento de actividades SQL internas, como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.
- Actividades de tipo CALL. En el caso de actividades CALL, el tiempo de procesador cuyo seguimiento se haya efectuado para el procedimiento no incluye el tiempo de procesador utilizado por cualquier actividad hija o por cualquier proceso de modalidad no delimitada. La condición de umbral sólo se comprobará cuando la devuelva la lógica de usuario al motor de la base de datos. Por ejemplo: durante la ejecución de una rutina fiable, la condición de umbral sólo se comprobará cuando la rutina emita una petición al motor de la base de datos.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. La granularidad del umbral CPUTIMEINSC es aproximadamente este número multiplicado por el grado de paralelismo de la actividad. Por ejemplo: si el umbral se comprueba cada 60 segundos y el grado de paralelismo es 2, la actividad puede utilizar 2 minutos más de tiempo de procesador, en lugar de 1 minuto antes de que se detecte la infracción de umbral. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

DATATAGINSC IN (*constante-entero*, ...)

Esta condición define uno o más valores de etiqueta de datos especificados en un espacio de tablas que la actividad toca. La etiqueta de datos de un espacio de tablas, o su grupo de almacenamiento subyacente (donde proceda) puede no establecerse o establecerse en un valor del 1 al 9. Si la actividad toca un espacio de tablas que no tiene ninguna etiqueta de datos establecida (en el nivel de espacio de tablas o de grupo de almacenamiento), este umbral no tendrá ningún efecto en esa actividad. El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037). Esta condición se implanta independientemente en cada partición de base de datos.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.
- Las actividades de DML anidado que se invocan directamente desde la lógica del usuario.

Esta condición no hace el seguimiento de las actividades de DML que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

Este umbral sólo se comprueba cuando se abre una exploración en una tabla o cuando se realiza una inserción en una tabla. La recuperación de datos de una tabla después de que se haya abierto una exploración no superará el umbral.

DATATAGINSC NOT IN (*constante-entero, ...*)

Esta condición define uno o más valores de etiqueta de datos no especificados en un espacio de tablas que la actividad toca. La etiqueta de datos de un espacio de tablas, o su grupo de almacenamiento subyacente (donde proceda) puede no establecerse o establecerse en un valor del 1 al 9. Si la actividad toca un espacio de tablas que no tiene ninguna etiqueta de datos establecida (en el nivel de espacio de tablas o de grupo de almacenamiento), este umbral no tendrá ningún efecto en esa actividad. El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037). Esta condición se implanta independientemente en cada partición de base de datos.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.
- Las actividades de DML anidado que se invocan directamente desde la lógica del usuario.

Esta condición no hace el seguimiento de las actividades de DML que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

Este umbral sólo se comprueba cuando se abre una exploración en una tabla o cuando se realiza una inserción en una tabla. La recuperación de datos de una tabla después de que se haya abierto una exploración no superará el umbral.

acciones-modificar-umbral-superado

Especifica la acción que va a adoptarse cuando se supera una condición. Cada vez que se supera una condición, se registra un suceso en todos los supervisores de sucesos de infracciones de umbral.

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad que ha superado el umbral deben enviarse a los supervisores de sucesos de actividades activas cuando la actividad finalice. El valor COLLECT ACTIVITY DATA no se aplica a los umbrales que no son de actividad, como CONNECTIONIDLETIME, TOTALDBPARTITIONCONNECTIONS, TOTALSCPARTITIONCONNECTIONS, CONCURRENTWORKLOADOCCURRENCES o UOWTOTALTIME.

*modificar-cláusula-recopilar-datos-actividad***ON COORDINATOR MEMBER**

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Para umbrales de predicción, la información sobre actividades sólo se recopila en todos los miembros si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL MEMBERS no tiene efecto y la información sobre actividades siempre se recopila únicamente en el miembro

ALTER THRESHOLD

coordinador. Para umbrales de predicción y de reacción, los valores de datos de entrada, la información de sección o los valores de actividades únicamente se recopilarán en el miembro coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la actividad finaliza su ejecución. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. En el caso de umbrales predictivos, los datos reales de sección se recopilarán en cualquier miembro donde se recopilen los datos de actividad. En el caso de umbrales reactivos, los datos reales de sección se recopilarán sólo en el miembro coordinador.

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

NONE

Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que supera el umbral.

STOP EXECUTION

La ejecución de la actividad se detiene y se devuelve un error (SQLSTATE 5U026). En el caso del umbral UOWTOTALTIME, la unidad de trabajo se retrotrae.

CONTINUE

La ejecución de la actividad no se detiene. Cuando la condición también tiene una cola, esta opción hace que la acción de poner en cola se extienda más allá del tamaño de la cola.

FORCE APPLICATION

Se fuerza que la aplicación salga del sistema (SQLSTATE 55032). Esta acción sólo puede especificarse para el umbral UOWTOTALTIME.

acción-volver-correlacionar-actividad

REMAP ACTIVITY TO *nombre-subclase-servicio*

La actividad se correlaciona con *nombre-subclase-servicio*. La ejecución de la actividad no se detiene. Esta acción sólo es válida para umbrales de clase en servicio como los umbrales CPUTIMEINSC, SQLROWSREADINSC, DATATAGINSC IN y DATATAGINSC NOT IN (SQLSTATE 5U037). El

nombre-subclase-servicio debe identificar una subclase de servicio existente en la misma superclase que tenga asociada el umbral (SQLSTATE 5U037). El *nombre-subclase-servicio* no puede ser igual que la subclase de servicio asociada del umbral (SQLSTATE 5U037).

NO EVENT MONITOR RECORD

Especifica que no se grabará ningún registro de infracción de umbral.

LOG EVENT MONITOR RECORD

Especifica que si existe un supervisor de sucesos THRESHOLD VIOLATIONS y está activo, se grabará en él un registro de infracción de umbral.

ENABLE o DISABLE

Especifica si el umbral está habilitado o no para que lo utilice el gestor de la base de datos.

ENABLE

El gestor de base de datos utiliza el umbral para restringir la ejecución de actividades de base de datos. Las actividades de base de datos que están en ejecución en la actualidad seguirán ejecutándose sin las restricciones de este umbral.

DISABLE

El gestor de base de datos no utiliza el umbral para restringir la ejecución de actividades de base de datos. Este umbral no restringirá actividades de base de datos nuevas. Los umbrales con una cola, por ejemplo, TOTALSCMEMBERCONNECTIONS o CONCURRENTDBCOORDACTIVITIES, deben inhabilitarse antes de que puedan descartarse.

Notas

- Los umbrales pueden definirse basándose en distintos aspectos del comportamiento de la base de datos para supervisar y controlar ese comportamiento. Cuando se define un umbral basado en actividades, a menos que se especifique algo distinto, éste sólo se aplicará durante la ejecución real de las sentencias de SQL, sin incluir el tiempo de compilación y el programa de utilidad de carga.
- El umbral CONCURRENTWORKLOADOCCURRENCES y el umbral CONCURRENTWORKLOADACTIVITIES son distintos en cuanto al ámbito. CONCURRENTWORKLOADOCCURRENCES controla cuántas conexiones pueden correlacionarse con una definición de carga de trabajo simultáneamente, y CONCURRENTWORKLOADACTIVITIES controla cuántas actividades puede someter simultáneamente cada correlación que se correlaciona con la definición de carga de trabajo.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- El valor nuevo para un umbral sólo afecta a las actividades de DB2 que comienzan a ejecutarse después de que se confirme la operación de modificación.

ALTER THRESHOLD

- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse TOTALDBPARTITIONCONNECTIONS en lugar de TOTALMEMBERCONNECTIONS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse TOTALSCPARTITIONCONNECTIONS en lugar de TOTALSCMEMBERCONNECTIONS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplo

Modifique el umbral MAXBIGQUERIESCONCURRENCY por un máximo de tres actividades en vez de dos.

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY  
  WHEN CONCURRENTDBCOORDACTIVITIES > 3  
  STOP EXECUTION
```

Debido a que este es un umbral con una cola, el umbral no puede descartarse a menos que se inhabilite, del siguiente modo:

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY DISABLE
```

ALTER TRIGGER

La sentencia ALTER TRIGGER cambia la descripción de un activador del servidor actual.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES se especifica implícita o explícitamente.

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SECADM
- Autorización CREATE_SECURE_OBJECT

Sintaxis

```
▶▶ ALTER TRIGGER nombre-activador [SECURED | NOT SECURED] ▶▶
```

Descripción

nombre-activador

Identifica el activador que se debe alterar. El *nombre-activador* debe identificar un activador que exista en el servidor actual (SQLSTATE 42704).

NOT SECURED o SECURED

Especifica si el activador se considera seguro.

SECURED

Especifica que el activador se considera seguro. SECURED debe especificarse para un activador cuya tabla sujeto sea una tabla en la que el control de acceso de nivel de fila o de nivel de columna se haya activado (SQLSTATE 428H8). Asimismo, SECURED debe especificarse para un activador que se cree en una vista y una o más de las tablas subyacentes de esa definición de vista tenga el control de acceso de nivel de fila o columna activado (SQLSTATE 428H8).

NOT SECURED

Especifica que el activador no se considera seguro. No se puede cambiar un activador de seguro a no seguro si está definido en una tabla para la cual está activado el control de acceso de nivel de fila o columna (SQLSTATE 428H8). Asimismo, no se puede cambiar un activador de seguro a no seguro si se ha definido en una vista y una o más de las tablas subyacentes de la definición de la vista tienen activado el control de acceso de nivel de fila o columna (SQLSTATE 428H8).

Ejemplos

- *Ejemplo 1:* cambiar activador de TRIGGER1 a SECURED.
ALTER TRIGGER TRIGGER1 SECURED
- *Ejemplo 2:* cambiar activador de TRIGGER1 a NOT SECURED.
ALTER TRIGGER TRIGGER1 NOT SECURED

ALTER TRUSTED CONTEXT

La sentencia ALTER TRUSTED CONTEXT modifica la definición de un contexto fiable en el servidor actual.

Invocación

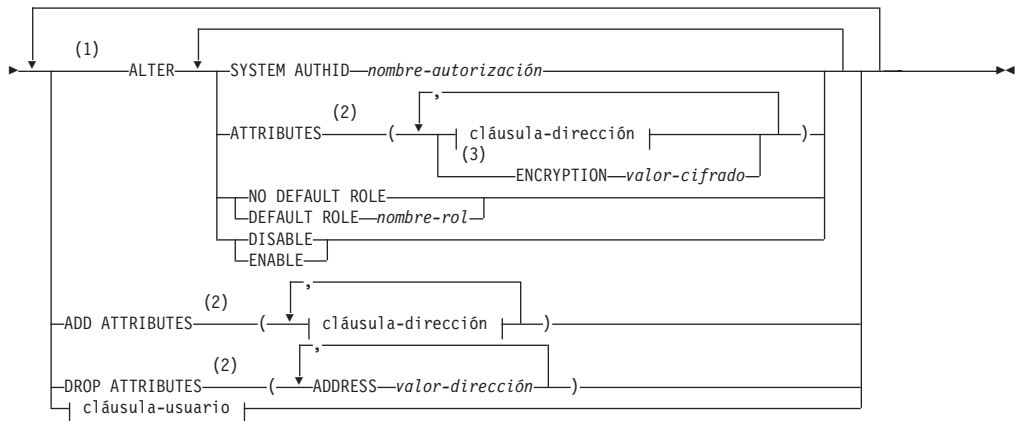
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

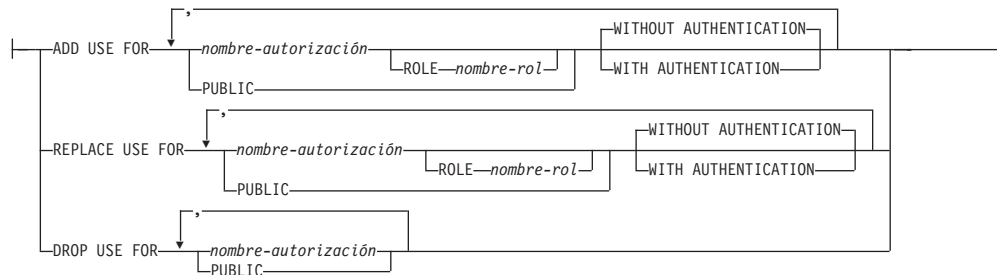
►►ALTER TRUSTED CONTEXT—*nombre-contexto*—►►



cláusula-dirección:



cláusula-usuario:



Notas:

- 1 Cada una de las cláusulas ATTRIBUTES, DEFAULT ROLE, ENABLE y WITH USE puede especificarse una vez como máximo (SQLSTATE 42614).
- 2 Cada nombre de atributo y su valor correspondiente deben ser exclusivos (SQLSTATE 4274D).
- 3 ENCRYPTION no se puede especificar más de una vez (SQLSTATE 42614); sin embargo, WITH ENCRYPTION puede especificarse para cada una de las veces que se especifique ADDRESS.

Descripción*nombre-contexto*

Identifica el contexto fiable que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704).

ALTER

Modifica las opciones y atributos de un contexto fiable.

SYSTEM AUTHID *nombre-autorización*

Especifica que el contexto es una conexión establecida por medio del ID de autorización del sistema *nombre-autorización*, que no debe asociarse con un contexto fiable existente (SQLSTATE 428GL). No puede ser el ID de autorización de la sentencia (SQLSTATE 42502).

ATTRIBUTES (...)

Especifica una lista de uno o más atributos de confianza de conexión, sobre la que se define el contexto fiable, que van a modificarse. Los valores existentes para los atributos especificados se sustituyen por los valores nuevos. Si un atributo no forma parte actualmente de la definición de contexto fiable, se devuelve un error (SQLSTATE 4274C). Los atributos que no se especifiquen retendrán sus valores anteriores.

ADDRESS *valor-dirección*

Especifica la dirección de comunicación real que utiliza el cliente para comunicar con el servidor de bases de datos. El único protocolo soportado es TCP/IP. Se eliminarán los valores anteriores ADDRESS para el contexto fiable especificado. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su

representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.

- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de contexto fiable si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio. Al crear un objeto de contexto fiable, resulta ventajoso proporcionar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP). Con DHCP, un dispositivo puede disponer de una dirección de IP diferente cada vez que se conecte a la red. Por tanto, si se proporciona una dirección IP estática para el atributo ADDRESS de un objeto de contexto fiable, es posible que algún dispositivo adquiriera alguna conexión fiable de modo no intencional. Proporcionar nombres de dominio para el atributo ADDRESS de un objeto de contexto fiable evita este problema en entornos DHCP.

WITH ENCRYPTION *valor-cifrado*

Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser uno de los valores siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autenticación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse un cifrado SSL (Secure Sockets Layer) para la comunicación de datos entre el cliente de

DB2 y el servidor de DB2 si una conexión entrante debe coincidir con el valor de cifrado para esta dirección específica

ENCRYPTION *valor-cifrado*

Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red. El valor por omisión es NONE.

valor-cifrado

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser uno de los valores siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico para una conexión de entrada que coincida con el atributo ENCRYPTION de este objeto de contexto fiable
- LOW, se necesita un cifrado ligero mínimo; el tipo de autenticación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable
- HIGH, debe utilizarse un cifrado SSL (Secure Sockets Layer) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 si una conexión de entrada debe coincidir con el atributo ENCRYPTION de este objeto de contexto fiable

Para obtener más detalles sobre el atributo fiable ENCRYPTION, consulte "CREATE TRUSTED CONTEXT".

NO DEFAULT ROLE o **DEFAULT ROLE** *nombre-rol*

Especifica si se asocia o no un rol por omisión con una conexión fiable basada en este contexto fiable. Si está activa una conexión fiable para este contexto, el cambio pasa a estar vigente en la siguiente petición de usuario de conmutador o una petición de conexión nueva.

NO DEFAULT ROLE

Especifica que el contexto fiable no tiene un rol por omisión.

DEFAULT ROLE *nombre-rol*

Especifica que *nombre-rol* es el rol por omisión para el contexto fiable. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). Este rol se utiliza con el usuario de una conexión fiable, basada en este contexto fiable, cuando el usuario no tenga definido un rol específico de usuario como parte de la definición del contexto fiable.

ENABLE o **DISABLE**

Especifica si el contexto fiable está habilitado o inhabilitado.

ENABLE

Especifica que el contexto fiable está habilitado.

DISABLE

Especifica que el contexto fiable está inhabilitado. Un contexto fiable que esté inhabilitado no se tomará en consideración cuando se establezca una conexión fiable.

ADD ATTRIBUTES

Especifica una lista de uno o más atributos fiables adicionales sobre la que se define el contexto fiable.

ADDRESS *valor-dirección*

Especifica la dirección de comunicación real que utiliza el cliente para

ALTER TRUSTED CONTEXT

comunicar con el servidor de bases de datos. El único protocolo soportado es TCP/IP. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.
- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com.

WITH ENCRYPTION *valor-cifrado*

Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser uno de los valores siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autenticación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse un cifrado SSL (Secure Sockets Layer) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 si una conexión de entrada debe coincidir con el atributo ENCRYPTION de este objeto de contexto fiable

DROP ATTRIBUTES

Especifica que se van a descartar uno o más atributos de la definición del contexto fiable. Si el atributo y el par de valores de atributo no forman parte actualmente de la definición de contexto fiable, se devuelve un error (SQLSTATE 4274C).

ADDRESS *valor-dirección*

Especifica que se va a eliminar de la definición del contexto fiable la dirección de comunicaciones identificada. El *valor-dirección* especifica una constante de serie que contiene el valor de un atributo fiable de ADDRESS existente.

ADD USE FOR

Especifica usuarios adicionales que pueden utilizar una conexión fiable basada en este contexto fiable. Si la definición de un contexto fiable permite el acceso por parte de PUBLIC y una lista de usuarios, las especificaciones para un usuario alterarán temporalmente las especificaciones para PUBLIC.

nombre-autorización

Especifica que la conexión fiable puede utilizarse por medio del *nombre-autorización* especificado. El *nombre-autorización* no debe identificar un ID de autorización que ya esté definido para la utilización del contexto fiable y no debe especificarse más de una vez en la cláusula ADD USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502).

ROLE *nombre-rol*

Especifica que *nombre-rol* es el rol que ha de utilizarse para el usuario. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). El rol explícitamente especificado para el usuario altera temporalmente los roles por omisión asociados con el contexto fiable.

PUBLIC

Especifica que cualquier usuario puede utilizar una conexión fiable basada en este contexto fiable. PUBLIC no debe estar previamente definido para la utilización del contexto fiable y no debe especificarse más de una vez en la cláusula ADD USE FOR (SQLSTATE 428GM).

WITHOUT AUTHENTICATION o WITH AUTHENTICATION

Especifica si conmutar el usuario actual en una conexión fiable basada en este contexto fiable requiere autenticación o no.

WITHOUT AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario no requiere autenticación.

WITH AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario requiere autenticación.

REPLACE USE FOR

Especifica la forma en la que va a cambiar un usuario concreto o PUBLIC utiliza el contexto fiable.

nombre-autorización

Especifica el *nombre-autorización* del usuario cuya utilización de la conexión fiable va a cambiarse. El contexto fiable debe estar ya definido para que lo utilice el *nombre-autorización* (SQLSTATE 428GN) y el *nombre-autorización* no

ALTER TRUSTED CONTEXT

debe especificarse más de una vez en la cláusula REPLACE USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502).

ROLE *nombre-rol*

Especifica que *nombre-rol* es el rol para el usuario. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). El rol explícitamente especificado para el usuario altera temporalmente los roles por omisión asociados con el contexto fiable.

PUBLIC

Especifica que van a cambiar los atributos para utilizar la conexión fiable por medio de PUBLIC. El contexto fiable debe estar ya definido para permitir que lo utilice PUBLIC (SQLSTATE 428GN) y PUBLIC no debe especificarse más de una vez en la cláusula REPLACE USE FOR (SQLSTATE 428GM).

WITHOUT AUTHENTICATION o **WITH AUTHENTICATION**

Especifica si conmutar el usuario actual en una conexión fiable basada en este contexto fiable requiere autenticación o no.

WITHOUT AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario no requiere autenticación.

WITH AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario requiere autenticación.

DROP USE FOR

Especifica la(s) persona(s) que ya no puede(n) utilizar el contexto fiable. Los usuarios eliminados de la definición del contexto fiable son los usuarios a los que se les permite en la actualidad utilizar el contexto fiable. En el caso de que se pueda eliminar de la definición del contexto fiable uno o más usuarios, pero no la totalidad de los mismos, se eliminarán los usuarios especificados y se devolverá un aviso (SQLSTATE 01682). En el caso de que no se pueda eliminar de la definición del contexto fiable ninguno de los usuarios especificados, se devolverá un aviso (SQLSTATE 428GN).

nombre-autorización

Elimina la posibilidad de que el ID de autorización especificado utilice este contexto fiable.

PUBLIC

Elimina la posibilidad de que todos los usuarios (excepto el ID de autorización del sistema y los ID de autorización individual que se hayan habilitado explícitamente) utilicen este contexto fiable.

Normas

- Una sentencia de SQL exclusiva del contexto fiable debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas del contexto fiable son:
 - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT o DROP (TRUSTED CONTEXT)

- Una sentencia de SQL exclusiva del contexto fiable no puede emitirse en una transacción global; por ejemplo, una transacción XA o una transacción global que se inicie como parte de una confirmación en dos fases para las transacciones federadas (SQLSTATE 51041).

Notas

- Al proporcionar una dirección IP como parte de una definición de contexto fiable, la dirección debe estar en el formato que esté vigente para la red. Por ejemplo, proporcionar una dirección en un formato IPv6 cuando la red sea IPv4 no dará como resultado una coincidencia. En un entorno mixto, resulta ventajoso especificar las representaciones IPv4 e IPv6 de la dirección, o mejor aún, especificar un nombre de dominio seguro (por ejemplo, corona.torolab.ibm.com), que oculta los detalles de formato de dirección.
- Sólo se permite una sentencia de SQL exclusiva del contexto fiable sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva del contexto fiable sin confirmar, las siguientes sentencias de SQL exclusivas del contexto fiable esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva del contexto fiable actual.
- Los cambios se graban en el catálogo del sistema pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- **Orden de operaciones:** El orden de las operaciones en una sentencia ALTER TRUSTED CONTEXT es:
 - DROP
 - ALTER
 - ADD ATTRIBUTES
 - ADD USE FOR
 - REPLACE USE FOR
- **Efecto de los cambios sobre las conexiones fiables existentes:** Si existen conexiones fiables para el contexto fiable que están modificándose, las conexiones seguirán siendo fiables con la definición que estaba vigente antes de la sentencia ALTER TRUSTED CONTEXT hasta que termine la siguiente conexión o petición de conmutación de usuario. Si se inhabilita el contexto fiable mientras están activas las conexiones fiables para este contexto, las conexiones seguirán siendo fiables >hasta que termine la siguiente conexión o petición de usuario de conmutador. Si se cambian los atributos fiables con la sentencia ALTER TRUSTED CONTEXT, se permite que continúen las conexiones fiables que existan mientras la sentencia ALTER TRUSTED CONTEXT utiliza el contexto fiable.
- **Privilegios de rol:** Si no hay ningún rol asociado al usuario o al contexto fiable, sólo son de aplicación los privilegios asociados con el usuario. Es igual que no estar en un contexto fiable.

Ejemplos

- *Ejemplo 1:* Suponga que existe el contexto fiable APPSERVER y que está habilitado. Emita una sentencia ALTER TRUSTED CONTEXT para permitir a Bill utilizar el contexto fiable APPSERVER, pero coloque el contexto fiable en estado de inhabilitado.

```
ALTER TRUSTED CONTEXT APPSERVER
DISABLE
ADD USE FOR BILL
```

- *Ejemplo 2:* Suponga que existe el contexto fiable SECUREROLE. Emita una sentencia ALTER TRUSTED CONTEXT para modificar el usuario existente Joe

ALTER TRUSTED CONTEXT

para que utilice el contexto fiable con la autenticación y para añadir a todos los demás para que utilicen el contexto fiable sin autenticación.

```
ALTER TRUSTED CONTEXT SECUREROLE  
  REPLACE USE FOR JOE WITH AUTHENTICATION  
  ADD USE FOR PUBLIC WITHOUT AUTHENTICATION
```

- *Ejemplo 3:* Suponga que hay un contexto fiable SECUREROLEENCRYPT con los valores de atributo ADDRESS '9.13.55.100' y '9.12.30.112' y un valor de atributo de ENCRYPTION 'NONE'. Emita una sentencia ALTER para modificar los valores de atributo ADDRESS y el atributo de cifrado a 'LOW'.

```
ALTER TRUSTED CONTEXT SECUREROLEENCRYPT  
  ALTER ATTRIBUTES (ADDRESS '9.12.155.200',  
  ENCRYPTION 'LOW')
```


ALTER TYPE (estructurado)

La sentencia ALTER TYPE se utiliza para añadir o descartar especificaciones de atributo o de método de un tipo estructurado definido por el usuario. También pueden modificarse las propiedades de los métodos existentes.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema del tipo
- Propietario del tipo, según se indica en la columna OWNER de la vista de catálogo SYSCAT.DATATYPES
- Autorización DBADM

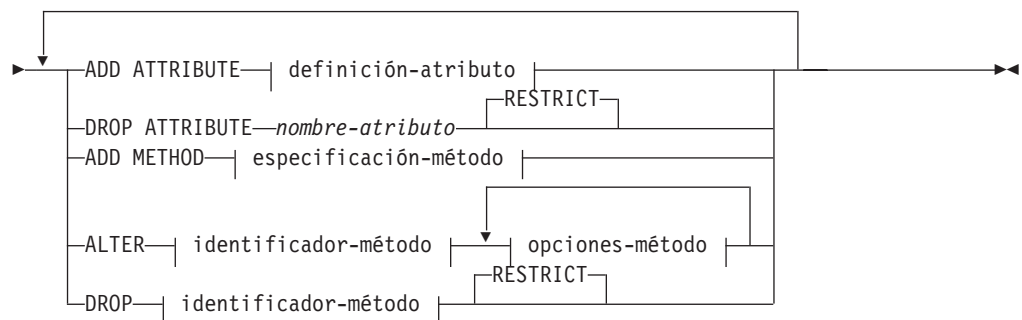
Para modificar un método con el fin de que no esté delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

Para modificar un método con el fin de que esté delimitado, no se necesita ninguna autorización ni privilegio adicional.

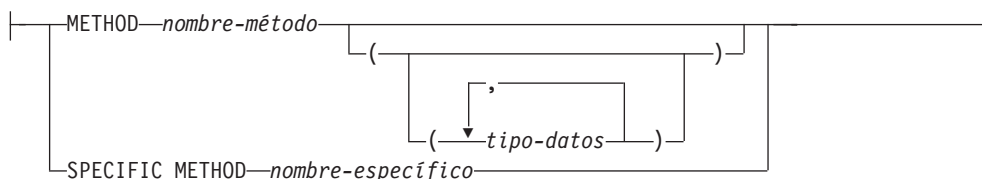
Sintaxis

►► ALTER TYPE *nombre-tipo* →



identificador-método:

ALTER TYPE (estructurado)



opciones-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 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.

ADD ATTRIBUTE

Añade un atributo después del último atributo del tipo estructurado existente.

definición-atributo

Define los atributos del tipo estructurado.

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 que se listan en CREATE TABLE, que no sean XML (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. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". Si el tipo de datos del atributo es un tipo de referencia, el tipo de destino de la referencia debe ser un tipo estructurado que exista (SQLSTATE 42704).

Para evitar las definiciones de tipo que, durante la ejecución, pueden permitir que una instancia del tipo contengan, directa o indirectamente, otra instancia del mismo tipo o uno de sus subtipos, existe una

restricción que establece que un tipo no puede definirse de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP).

opciones-lob

Especifica las opciones asociadas a tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de las opciones-lob, consulte "CREATE TABLE".

DROP ATTRIBUTE

Descarta un atributo del tipo estructurado existente.

nombre-atributo

El nombre del atributo. El atributo debe existir como un atributo del tipo (SQLSTATE 42703).

RESTRICT

Aplica la norma que establece que ningún atributo puede descartarse si se utiliza *nombre-tipo* como tipo de una tabla, una vista o una columna existente, de un atributo anidado en el tipo de una columna o de una extensión de índice.

ADD METHOD especificación-método

Añade una especificación de método al tipo que *nombre-tipo* identifica. Para poder utilizar el método es necesario darle un cuerpo mediante una sentencia CREATE METHOD separada. Para obtener más información acerca de la *especificación-método*, consulte "CREATE TYPE (Estructurado)".

ALTER identificador-método

Identifica de forma exclusiva una instancia de un método que va a modificarse. El método especificado puede o no tener un cuerpo de método existente. Los métodos declarados como LANGUAGE SQL no pueden modificarse (SQLSTATE 42917).

*identificador-método***METHOD nombre-método**

Identifica un método en particular y sólo es válido si existe exactamente una instancia de método con el nombre *nombre-método* para el tipo *nombre-tipo*. Para el método identificado puede definirse cualquier número de parámetros. Si no existe ningún método con este nombre para el tipo, se generará un error (SQLSTATE 42704). Si existe más de una instancia del método para el tipo, se generará un error (SQLSTATE 42725).

METHOD nombre-método (tipo-datos,...)

Proporciona la signatura del método, que identifica de forma exclusiva al método. No se utiliza el algoritmo de resolución de método.

nombre-método

Especifica el nombre del método para el tipo *nombre-tipo*.

(tipo-datos,...)

Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) en la sentencia CREATE TYPE. Para identificar la instancia de método específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de

ALTER TYPE (estructurado)

acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE).

Si se codifica la longitud, la precisión o la escala, el valor debe coincidir exactamente con el valor especificado en la sentencia CREATE TYPE.

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha 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 con la signatura especificada para el tipo en el esquema indicado o implícito, se generará un error (SQLSTATE 42883).

SPECIFIC METHOD *nombre-específico*

Identifica un método en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del método. 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. El *nombre-específico* debe identificar una instancia de método específica en el esquema indicado o implícito; en caso contrario, se genera un error (SQLSTATE 42704).

opciones-método

Especifica las opciones que van a modificarse para el método.

FENCED o NOT FENCED

Especifica si se considera que el método es seguro para ejecutarse en el espacio de dirección o en el proceso del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED). La mayoría de los métodos tienen la opción de ejecutarse como FENCED o NOT FENCED.

Si un método se altera para que sea FENCED, el gestor de bases de datos aísla sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que el método no pueda acceder a ellos. En general, un método que se ejecute como FENCED no funcionará tan bien como otro método similar que se ejecute como NOT FENCED.

PRECAUCIÓN:

El uso de métodos NOT FENCED que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 toman algunas precauciones ante varios de los tipos comunes de anomalías inadvertidas que puedan producirse, pero no pueden garantizar la integridad completa si se utilizan métodos NOT FENCED.

Un método declarado como NOT THREADSAFE no puede modificarse para que sea NOT FENCED (SQLSTATE 42613).

Si un método tiene algún parámetro cuya definición sea AS LOCATOR y se ha definido con la opción NO SQL, el método no puede modificarse para que sea FENCED (SQLSTATE 42613).

Esta opción no puede modificarse para los métodos LANGUAGE OLE (SQLSTATE 42849).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que un método es seguro para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si el método se ha definido con un LANGUAGE distinto de OLE:

- Si el método se ha definido como THREADSAFE, el gestor de bases de datos puede invocar el método en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, un método no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Los métodos FENCED y NOT FENCED pueden ser THREADSAFE. Si el método se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).
- Si el método se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará el método en el mismo proceso que otra rutina. Sólo un método FENCED puede ser NOT THREADSAFE (SQLSTATE 42613).

DROP *identificador-método*

Identifica de forma exclusiva una instancia de un método que va a descartarse. El método especificado no debe tener un cuerpo de método existente (SQLSTATE 428ER). Utilice la sentencia DROP METHOD para descartar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD. Los métodos generados implícitamente por la sentencia CREATE TYPE (tales como mutadores y observadores) no se pueden descartar (SQLSTATE 42917).

RESTRICT

Indica que el método especificado no puede tener un cuerpo de método existente. Utilice la sentencia DROP METHOD para descartar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD.

Normas

- La adición o el descarte de un atributo no está permitido para el tipo *nombre-tipo* (SQLSTATE 55043) si:
 - 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 el *nombre-tipo*. Los términos *utiliza directamente* y *utiliza indirectamente* se definen en “Tipos estructurados”.
 - El tipo o uno de sus subtipos se utiliza en una extensión de índice.
- Un tipo no puede modificarse añadiendo atributos de tal modo que el número total de atributos para el tipo, o cualquiera de sus subtipos, sea superior a 4082 (SQLSTATE 54050).
- Opción ADD ATTRIBUTE:
 - ADD ATTRIBUTE genera métodos de observador y mutador para el nuevo atributo. Estos métodos son similares a los que se generan durante la creación de un tipo estructurado (véase “CREATE TYPE (Estructurado)”). Si estos

ALTER TYPE (estructurado)

métodos entran en conflicto con otros métodos u otras funciones existentes, la sentencia ALTER TYPE da un error (SQLSTATE 42745).

- Si el usuario ha especificado de forma explícita el valor de INLINE LENGTH para el tipo (o cualquiera de sus subtipos) con un valor inferior a 292, y los atributos añadidos dan lugar a que la longitud en línea especificada sea inferior que el tamaño del resultado de la función de construcción para el tipo modificado (32 bytes más 10 bytes por atributo), se genera un error (SQLSTATE 42611).
- Opción DROP ATTRIBUTE:
 - Un atributo que se hereda de un supertipo existente no puede descartarse (SQLSTATE 428DJ).
 - DROP ATTRIBUTE descarta los métodos de mutador y observador de los atributos eliminados y comprueba si hay dependencias respecto a esos métodos descartados.
- Opción DROP METHOD:
 - No se puede descartar un método original que otros métodos alteran temporalmente (SQLSTATE 42893).

Notas

- No es posible modificar un método que esté en el esquema SYSIBM, SYSPROC o SYSPROC (SQLSTATE 42832).
- Cuando se altera un tipo, mediante la adición o el descarte 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 descarta 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 (de manera recursiva) 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 el descarte 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.
- En un entorno de base de datos particionada, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- **Privilegios:** el privilegio EXECUTE no se otorgará para ninguno de los métodos explícitamente especificados en la sentencia ALTER TYPE hasta que se haya definido un cuerpo de método mediante la utilización de la sentencia CREATE METHOD. El propietario del tipo definido por el usuario tiene la capacidad de descartar la especificación del método utilizando una sentencia ALTER TYPE.

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 descartar 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. Puesto que la longitud en línea no se ha especificado en la sentencia CREATE TYPE original, la base de datos DB2 calcula de nuevo la longitud en línea añadiendo 13 (10 bytes para el atributo nuevo + 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 da por supuesto que el tipo EMP incluye un atributo denominado SALARY, el siguiente ejemplo muestra una definición de cuerpo de método.

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP
RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

ALTER USAGE LIST

La sentencia ALTER USAGE LIST modifica la definición de una lista de uso.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

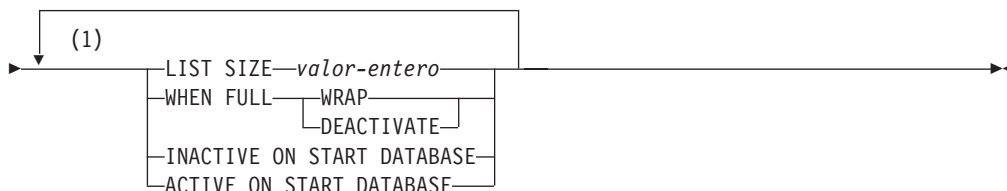
Autorización

El ID de autorización de la sentencia debe tener uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SQLADM

Sintaxis

►► ALTER USAGE LIST *nombre-lista-uso* ►►



Notas:

- 1 No se debe especificar la misma cláusula más de una vez.

Descripción

nombre-lista-uso

Identifica la lista de uso que se debe alterar. El *nombre-lista-uso* debe identificar una lista de uso que exista en el servidor actual (SQLSTATE 42704).

LIST SIZE *valor-entero*

Especifica que el tamaño de esta lista son *valor-entero* entradas. El tamaño mínimo que se puede especificar es de 10 y el máximo de 5000 (SQLSTATE 428B7).

WHEN FULL

Especifica la acción que ha de realizarse cuando una lista de uso activa se llena.

WRAP

Especifica que la lista de uso se ajusta y sustituye las entradas mas antiguas.

DEACTIVATE

Especifica que la lista de uso se desactiva.

INACTIVE ON START DATABASE

Especifica que la lista de uso no está activada para la supervisión siempre que

la base de datos está activada. La colección se debe iniciar de forma explícita utilizando la sentencia SET USAGE LIST.

ACTIVE ON START DATABASE

Especifica que la lista de uso se activa automáticamente para la supervisión siempre que la base de datos está activada. En un entorno de entorno de base de datos particionada o DB2 pureScale, la colección se inicia de forma automática cuando se activa el miembro de la base de datos.

Notas

- *Cuando los cambios entran en vigor:* si el estado actual de una lista de uso es activo, las modificaciones no entrarán en vigor cuando se procese la sentencia ni cuando se confirmen los cambios. Los cambios en la lista de uso entrarán en vigor la siguiente vez que el estado de la lista de uso se establezca en activo. En un entorno de base de datos particionada o en un entorno DB2 pureScale, las alteraciones entrarán en vigor la siguiente vez que la lista de uso se active en un miembro.

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

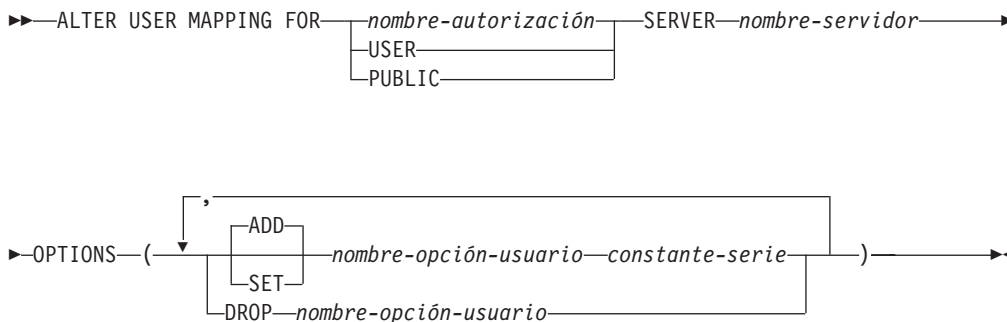
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de la autorización que se correlaciona a la fuente de datos, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o autorizaciones.

Cuando se modifica una correlación de usuarios públicos, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM.

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 ha especificado en la opción de usuario REMOTE_AUTHID.

PUBLIC

Especifica que cualquier ID de autorización válido de la base de datos federada local se correlacione con el ID de autorización de la fuente de datos especificado 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 deben habilitar, restaurar o descartar para la correlación que se está modificando.

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 debe habilitar o restaurar.

constante-serie

Especifica el valor para *nombre-opción-usuario* como una constante de serie de caracteres.

DROP *nombre-opción-usuario*

Descarta 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 descarta una opción de usuario, no afecta a ninguna otra opción de usuario que se esté utilizando.
- Una sentencia ALTER USER MAPPING de una unidad de trabajo (UOW) determinada no puede procesarse (SQLSTATE 55007) si la UOW ya incluye uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de la fuente de datos que se debe incluir en la correlación
 - Un cursor abierto en un apodo para una tabla o una vista de la fuente de datos que se debe incluir en la correlación
 - Una sentencia INSERT, DELETE o UPDATE emitida para un apodo de una tabla o una vista de la fuente de datos que se debe incluir en la correlación.
- No pueden coexistir en el mismo servidor federado correlaciones de usuarios públicos y correlaciones de usuarios no públicos. Esto significa que, si ha creado correlaciones de usuarios públicos, no podrá crear correlaciones de usuarios no públicos en el mismo servidor federado. Asimismo, si ha creado correlaciones de usuarios no públicos, no podrá crear correlaciones de usuarios públicos en el mismo servidor federado.

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 para z/OS llamada DORADO. Utiliza un ID de autorización para acceder a DB2 y otro para acceder a DORADO y ha creado una correlación entre

ALTER USER MAPPING

los dos ID. Ha utilizado la misma contraseña con ambos ID, pero ahora decide utilizar una contraseña diferente, 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. La sentencia ALTER VIEW también habilita o inhabilita una vista para su utilización en la optimización de la consulta.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

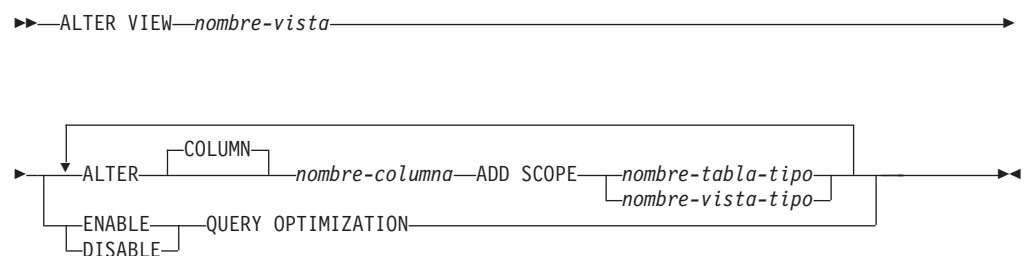
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTERIN para el esquema de la vista
- Propietario de la vista que se debe modificar
- Privilegio CONTROL para la vista a modificar
- Autorización DBADM

Para habilitar o inhabilitar una vista para utilizarla en la optimización de la consulta, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para cada una de las tablas o de las tablas subyacentes de las vistas a las que se hace referencia en la cláusula FROM de la selección completa de la vista:

- Privilegio ALTER para la tabla
- Privilegio ALTERIN para el esquema de la tabla
- Autorización DBADM

Sintaxis



Descripción

nombre-vista

Especifica la vista que debe cambiarse. Debe ser una vista que esté descrita en el catálogo.

ALTER COLUMN *nombre-columna*

Especifica el nombre de la columna que se debe modificar. 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 no tenga

ALTER VIEW

todavía ningún ámbito definido (SQLSTATE 428DK). La columna no debe ser herencia de una supervista (SQLSTATE 428DJ).

nombre-tabla-tipo

Especifica 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 ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en el *nombre-tabla-tipo*.

nombre-vista-tipo

Especifica 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*.

ENABLE QUERY OPTIMIZATION o DISABLE QUERY OPTIMIZATION

Especifica si la vista y las estadísticas asociadas se deben utilizar para mejorar la optimización de las consultas. **DISABLE QUERY OPTIMIZATION** es el valor por omisión cuando se crea una vista.

ENABLE QUERY OPTIMIZATION

Especifica que la vista incluye estadísticas que pueden utilizarse para mejorar la optimización de las consultas que implican esta vista o las consultas que incluyen subconsultas parecidas a la selección completa de esta vista.

DISABLE QUERY OPTIMIZATION

Especifica que la vista y las estadísticas asociadas no deben utilizarse para mejorar la optimización de las consultas.

Normas

- Una vista no puede habilitarse para la optimización de una consulta si:
 - La vista hace referencia directa o indirectamente a una tabla de consulta materializada (MQT). Tenga en cuenta que una MQT o una vista de estadísticas puede establecer una referencia a una vista de estadísticas.
 - La vista hace referencia directa o indirectamente a una tabla de catálogo.
 - Es una vista con tipo

Notas

- Para que puede considerarse para la optimización de una consulta, una vista:
 - No puede contener una operación diferenciada ni de agregación
 - No puede contener una operación de unión, excepción o intersección
 - No puede contener una especificación OLAP
- Si una vista se modifica para inhabilitar la optimización de consultas, los planes de consultas en antememoria que utilizaban la vista para la optimización de consultas se invalidarán. Si una vista se modifica para habilitar la optimización de consultas, los planes de consulta en antememoria se invalidarán si hacen referencia a las mismas tablas que las nuevas referencias de vista habilitadas, ya sea directa o indirectamente a través de otras vistas. La invalidación de estos planes de consulta en antememoria da lugar a una revalidación implícita que tiene en cuenta la propiedad de optimización de consulta modificada de la vista. La propiedad de optimización de consulta para una vista no tiene impacto alguno en las sentencias de SQL integradas estáticas.

ALTER WORK ACTION SET

La sentencia ALTER WORK ACTION SET modifica un conjunto de acciones de trabajo añadiendo, modificando o descartando acciones de trabajo en el conjunto de acciones de trabajo.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

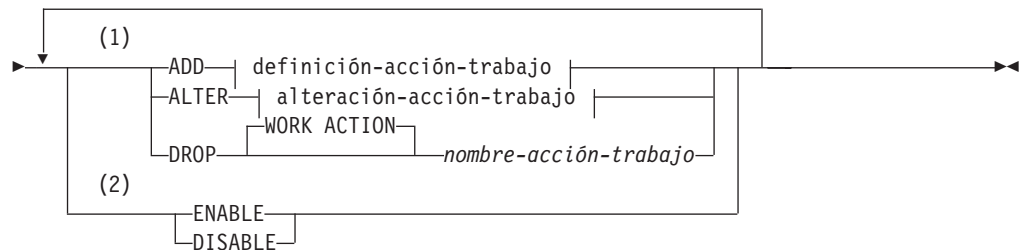
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autoridad SQLADM, únicamente si todas las cláusulas de modificación son cláusulas COLLECT.
- Autorización WLMADM
- Autorización DBADM

Sintaxis

►► ALTER WORK ACTION SET *nombre-conjunto-acciones-trabajo* ►►



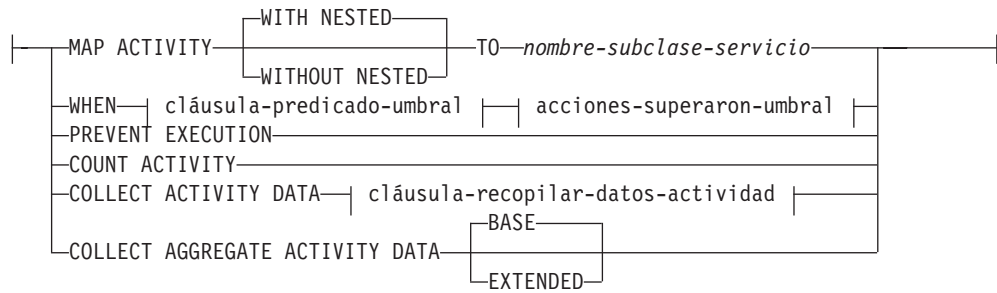
definición-acción-trabajo:

WORK ACTION *nombre-acción-trabajo* ON WORK CLASS *nombre-clase-trabajo* ►►

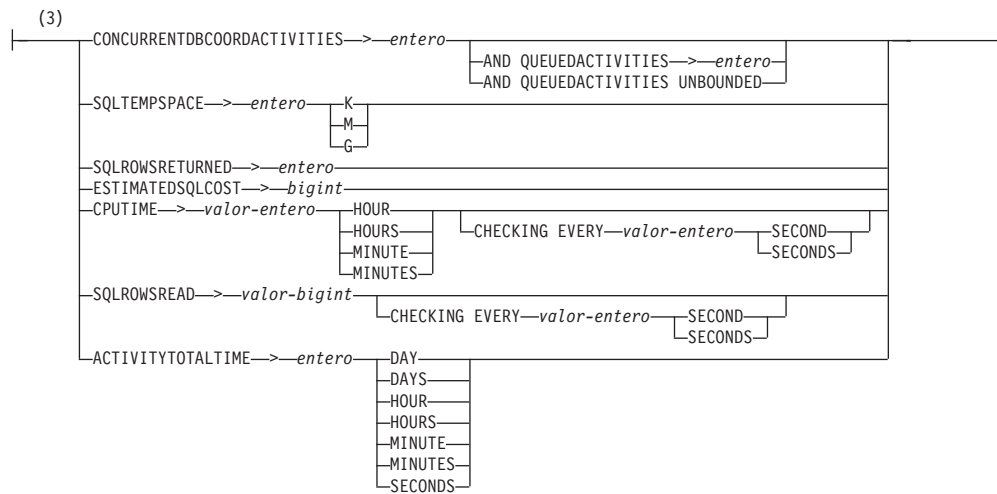
►► cláusula-tipos-acción | cláusula-plantilla-histograma | [ENABLE] | [DISABLE] ►►

cláusula-tipos-acción:

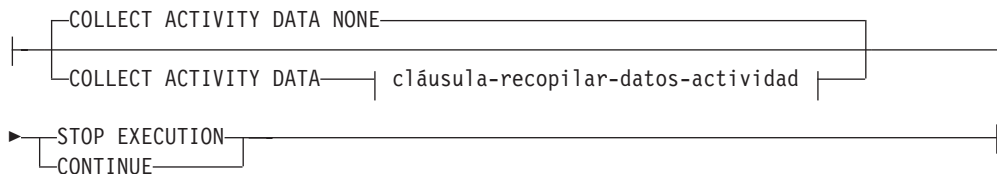
ALTER WORK ACTION SET



cláusula-predicado-umbral:

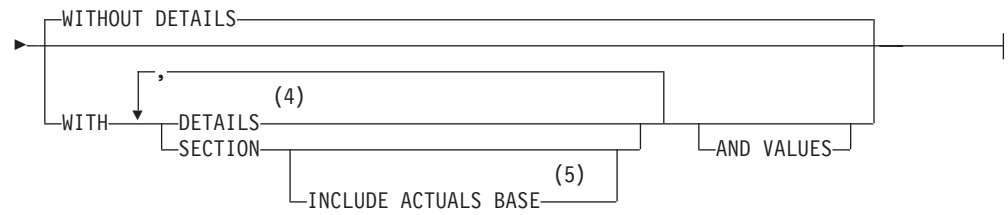


acciones-superaron-umbral:

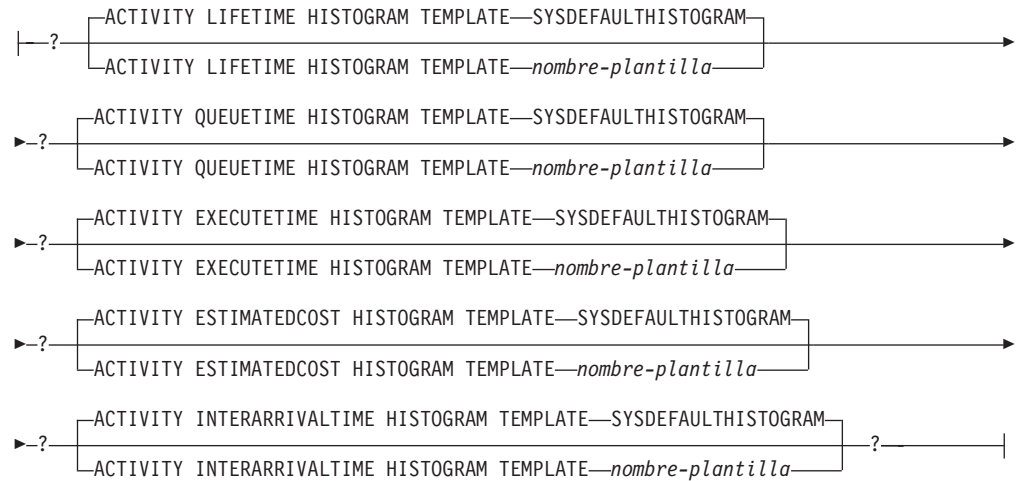


cláusula-recopilar-datos-actividad:

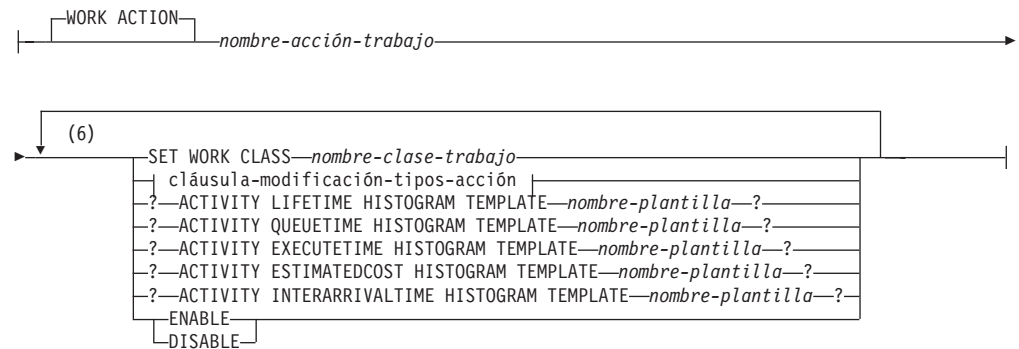




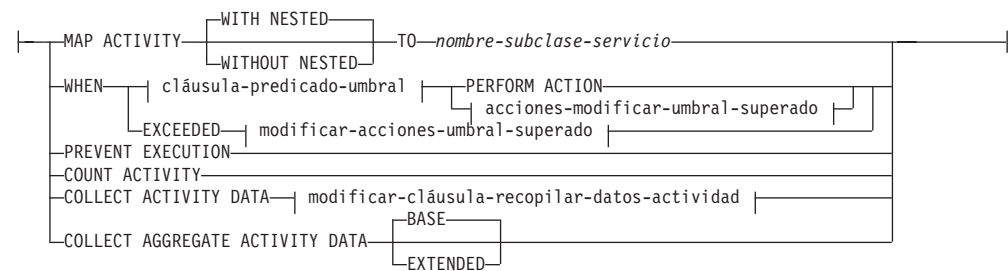
cláusula-plantilla-histograma:



alteración-acción-trabajo:

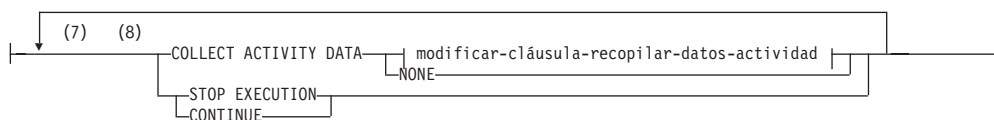


cláusula-modificación-tipos-acción:

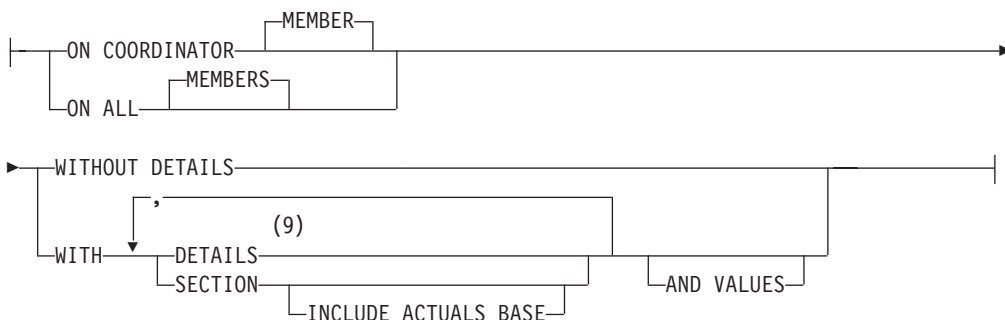


ALTER WORK ACTION SET

acciones-modificar-umbral-superado:



modificar-cláusula-recopilar-datos-actividad:



Notas:

- 1 Las cláusulas ADD, ALTER y DROP se procesan en el orden en que se especifican.
- 2 La cláusula ENABLE o DISABLE sólo puede especificarse una vez en la misma sentencia.
- 3 Sólo puede aplicarse una acción de trabajo del mismo tipo de umbral a una única clase de trabajo a la vez. Cuando se modifica una acción de trabajo de umbral, no se puede cambiar el predicado de umbral.
- 4 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.
- 5 Esta cláusula no se aplica a los umbrales.
- 6 Una misma cláusula no se debe especificar más de una vez.
- 7 Una misma cláusula no se debe especificar más de una vez.
- 8 Si una acción de trabajo existente no tiene una acción de umbral superado definida y se va a modificar para que pase a ser una acción de trabajo de umbral, debe especificarse STOP EXECUTION o CONTINUE, y si no se especifica COLLECT ACTIVITY DATA, COLLECT ACTIVITY DATA NONE es el valor por omisión.
- 9 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-conjunto-acciones-trabajo

Identifica el conjunto de acciones de trabajo que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704).

ADD

Añade una acción de trabajo al conjunto de acciones de trabajo.

WORK ACTION *nombre-acción-trabajo*

Da nombre a la acción de trabajo. El *nombre-acción-trabajo* no debe identificar una acción de trabajo que ya exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42710). El *nombre-acción-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

ON WORK CLASS *nombre-clase-trabajo*

Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY

Especifica una acción de trabajo de correlación de la actividad. Esta acción sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED

Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED

Todas las actividades de base de datos que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de datos anidadas bajo esta actividad, se correlacionarán a la subclase de servicio; es decir, las actividades con un nivel de anidamiento superior a cero se ejecutarán en la misma clase de servicio que las actividades con un nivel de anidamiento de cero.

WITHOUT NESTED

Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO *nombre-subclase-servicio*

Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN

Especifica el umbral que se aplicará a la actividad de base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo. Un umbral sólo puede especificarse si el objeto de gestor de base de datos para el que se ha definido esta acción de trabajo está definido como base de datos (SQLSTATE 5U034). Ninguno de estos umbrales se aplican a las actividades de base de datos internas iniciadas por medio del gestor de base de datos o a las actividades de base de datos generadas por medio de rutinas de SQL administrativas.

ALTER WORK ACTION SET

cláusula-predicado-umbral

Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia "CREATE THRESHOLD".

acciones-superaron-umbral

Para obtener una descripción de las acciones que superaron el umbral válidas, consulte la sentencia "CREATE THRESHOLD".

PREVENT EXECUTION

Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY

Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida, deben enviarse a cualquier supervisor de sucesos de actividades activas cuando se completa la actividad.

cláusula-recopilar-datos-actividad

ON COORDINATOR MEMBER

Especifica que los datos de actividad se deben recopilar en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todos los miembros si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL MEMBERS no tiene efecto y la información sobre actividades siempre se recopila únicamente en el miembro coordinador. Para umbrales de predicción y de reacción, los valores de datos de entrada, la información de sección o los valores de actividades únicamente se recopilarán en el miembro coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la actividad finaliza su ejecución. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que la sentencia, los datos del entorno de compilación y los del entorno de sección deben enviarse a

cualquier supervisor de sucesos de actividades activas que dispongan de los mismos. Se debe especificar DETAILS si se especifica SECTION.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actuals**.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula INCLUDE ACTUALS, el parámetro de configuración de base de datos **section_actuals** y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del parámetro de configuración de la base de datos **section_actuals** es NONE y <collectsectionactuals> se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

BASE especifica que han de habilitarse y recopilarse los datos reales siguientes durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

NONE

Especifica que los datos de actividad no se deben recopilar para cada actividad que esté asociada con la clase de trabajo para la que se define esta acción de trabajo.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE

Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado

ALTER WORK ACTION SET

- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal. Sólo las actividades a las que se ha aplicado un umbral SQLTEMPSPACE participan en esta marca de límite superior.
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

ENABLE o DISABLE

Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

ENABLE

Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE

Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometan actividades de base de datos.

cláusula-plantilla-histograma

Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades asociadas a la clase de trabajo a la que se asigna esta acción de trabajo. Los datos de actividades agregadas sólo se recopilan para la clase de trabajo cuando el tipo de acción de trabajo es COLLECT AGGREGATE ACTIVITY DATA.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos de las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada miembro en el que se ejecuta la actividad. En el miembro coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En los miembros que no son coordinadores, este es el tiempo que estos miembros pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en un miembro no coordinador más de una vez y cada vez el miembro no coordinador recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en un miembro. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ALTER

Modifica la definición de la acción de trabajo. Puede cambiar la clase de trabajo a la que se aplica esta acción de trabajo y la acción que va a aplicarse a la actividad de base de trabajo que caiga dentro de la clase de trabajo.

WORK ACTION *nombre-acción-trabajo*

Identifica la acción de trabajo. El *nombre-acción-trabajo* debe identificar una acción de trabajo que exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42704).

SET WORK CLASS *nombre-clase-trabajo*

Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY

Especifica una acción de trabajo de correlación de la actividad. Esta acción

ALTER WORK ACTION SET

sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED

Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED

Todas las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de trabajo anidadas bajo esta actividad se correlacionarán con la subclase de servicio.

WITHOUT NESTED

Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO *nombre-subclase-servicio*

Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN

Especifica el umbral que se modificará para la actividad de la base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo.

cláusula-predicado-umbral

Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia "CREATE THRESHOLD".

PERFORM ACTION

Cuando se modifica el valor de la condición de predicado de umbral, especifica que la acción que ha superado el umbral no ha cambiado. La acción de trabajo debe ser un umbral (SQLSTATE 42613).

acciones-modificar-umbral-superado

Para obtener una descripción de modificar-acciones-superaron-umbral válida, consulte las acciones-superaron-umbral de la sentencia "CREATE THRESHOLD".

EXCEEDED

Especifica que debe conservarse el mismo predicado de umbral que se especificó en un principio para este umbral modificado. La acción de trabajo debe ser un umbral (SQLSTATE 42613).

PREVENT EXECUTION

Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY

Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida, deben enviarse a cualquier supervisor de sucesos de actividades activas cuando se completa la actividad.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la actividad finaliza su ejecución. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH**DETAILS**

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. Los datos reales de sección sólo se recopilarán en los miembros donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actuals**.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula INCLUDE ACTUALS, el parámetro de configuración de base de datos **section_actuals** y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del parámetro de

configuración de la base de datos **section_actuals** es NONE y `<collectsectionactuals>` se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

BASE especifica que han de habilitarse y recopilarse los datos reales siguientes durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

NONE

Especifica que los datos de actividad no se deben recopilar para cada actividad que esté asociada con la clase de trabajo para la que se define esta acción de trabajo.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE

Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado de DML de la actividad
- Histograma de tiempo de llegada de DML de actividad

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos de las actividades de

DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada miembro en el que se ejecuta la actividad. En el miembro coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En los miembros que no son coordinadores, este es el tiempo que estos miembros pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en un miembro no coordinador más de una vez y cada vez el miembro no coordinador recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en un miembro. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades de lenguaje de manipulación de datos (DML) asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ENABLE o DISABLE

Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos.

ALTER WORK ACTION SET

ENABLE

Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometán actividades de base de datos.

DISABLE

Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometán actividades de base de datos.

DROP *nombre-acción-trabajo*

Descarta una acción de trabajo del conjunto de acciones de trabajo. El *nombre-acción-trabajo* debe identificar una acción de trabajo que exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42704).

Un umbral creado como parte de un conjunto de acciones de trabajo no se puede manipular directamente. Debe inhabilitar en primer lugar la acción de trabajo para inhabilitar el umbral. A continuación, puede descartar la acción de trabajo una vez que el umbral no se utilice. Para obtener más información, consulte la sección sobre cómo descartar una acción de trabajo en la publicación *Guía y consulta de DB2 Workload Management*.

ENABLE o DISABLE

Especifica si el conjunto de acciones de trabajo va a tomarse en consideración o no cuando se sometán actividades de base de datos.

ENABLE

Especifica que el conjunto de acciones de trabajo está habilitado y va a tomarse en consideración cuando se sometán actividades de base de datos.

DISABLE

Especifica que el conjunto de acciones de trabajo está inhabilitado y no va a tomarse en consideración cuando se sometán actividades de base de datos.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (plantilla de histograma)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (clase de servicio)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (umbral)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (conjunto de acciones de trabajo)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (conjunto de clases de trabajo)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (carga de trabajo)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

- Los umbrales con una cola, por ejemplo, CONCURRENTDBCOORDACTIVITIES, deben inhabilitarse antes de que puedan descartarse.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* Modifique el conjunto de acciones de trabajo DATABASE_ACTIONS y añada dos acciones de trabajo utilizando la clase de trabajo LARGE_SELECTS. Para la acción de trabajo ONE_CONCURRENT_SELECT, aplique un umbral de simultaneidad de 1 para controlar el número de actividades que puede ejecutarse a la vez y permitir que como máximo haya 3 en cola. Para la acción de trabajo BIG_ROWS_RETURNED, limite el número de filas que puedan devolver las actividades de base de trabajo que caigan en dicha clase a 1 000 000.

```
ALTER WORK ACTION SET DATABASE_ACTIONS
ADD WORK ACTION ONE_CONCURRENT_SELECT ON WORK CLASS LARGE_SELECTS
  WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 3 STOP EXECUTION
ADD WORK ACTION BIG_ROWS_RETURNED ON WORK CLASS LARGE_SELECTS
  WHEN SQLROWSRETURNED > 1000000 STOP EXECUTION
```

- *Ejemplo 2:* Modifique el conjunto de acciones de trabajo ADMIN_APPS_ACTIONS al objeto de modificar la acción de trabajo MAP_SELECTS para correlacionar todas las actividades que se ejecuten en la clase de super servicio ADMIN_APPS bajo la clase de trabajo SELECT_CLASS con la subclase de servicio ALL_SELECTS. Añada asimismo una acción de trabajo nueva denominada MAP_UPDATES que correlacione todas las actividades que se ejecuten en la clase de trabajo UPDATE_CLASS con la subclase de servicio ALL_SELECTS.

```
ALTER WORK ACTION SET ADMIN_APPS_ACTIONS
ALTER WORK ACTION MAP_SELECTS MAP ACTIVITY TO ALL_SELECTS
ADD WORK ACTION MAP_UPDATES ON WORK CLASS UPDATE_CLASS
  MAP ACTIVITY TO ALL_SELECTS
```

ALTER WORK CLASS SET

La sentencia ALTER WORK CLASS SET añade, modifica o descarta clases de trabajo dentro de un conjunto de clases de trabajo.

Invocación

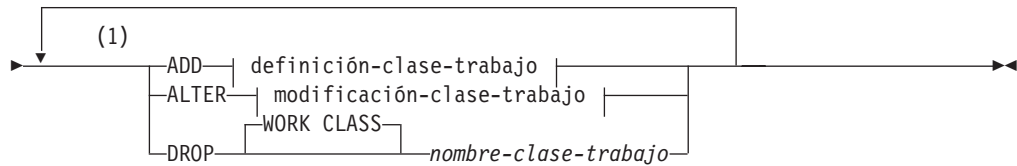
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

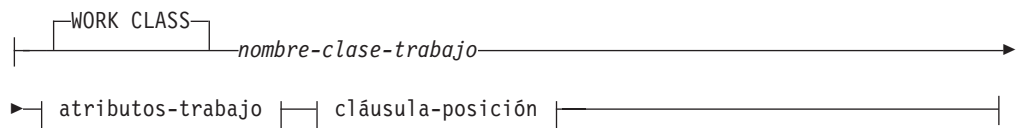
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Sintaxis

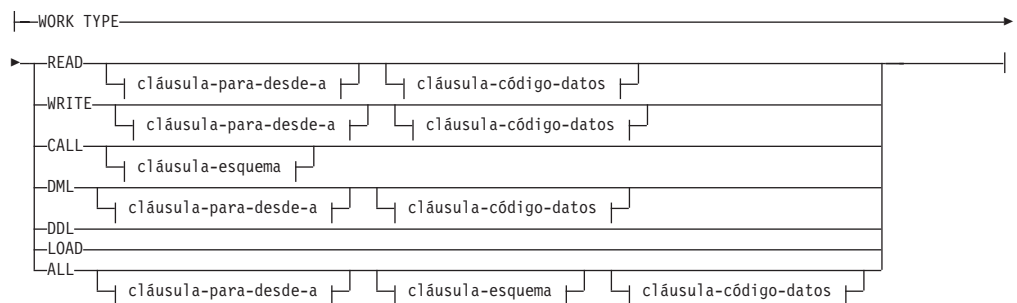
► ALTER WORK CLASS SET *nombre-conjunto-clases-trabajo* →

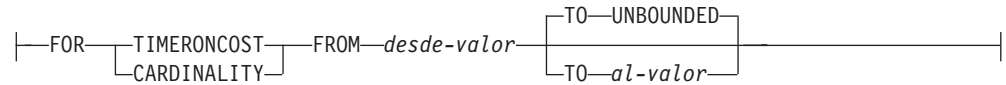
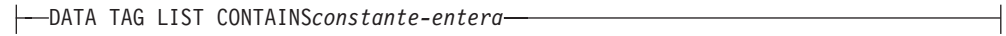
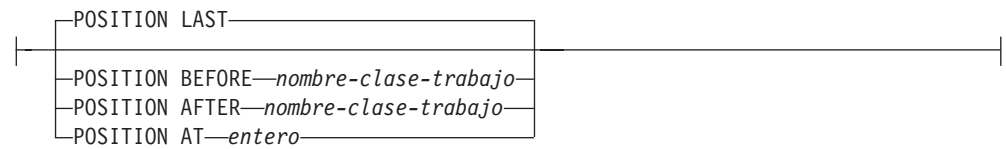
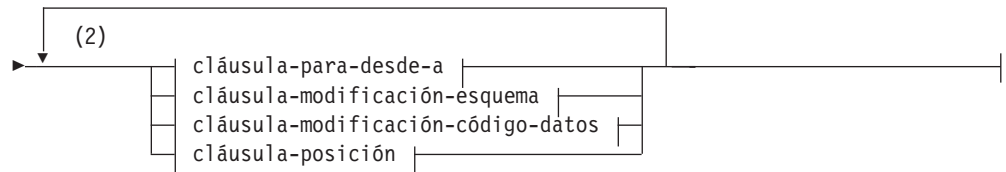
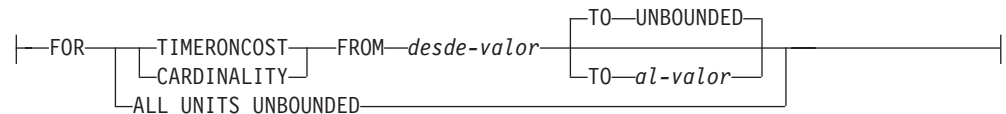


definición-clase-trabajo:



atributos-trabajo:



cláusula-para-desde-a:**cláusula-código-datos:****cláusula-esquema:****cláusula-posición:****modificación-clase-trabajo:****cláusula-para-desde-a:****cláusula-modificación-esquema:****cláusula-modificación-código-datos:**

ALTER WORK CLASS SET

Notas:

- 1 Las cláusulas ADD, ALTER y DROP se procesan en el orden en que se especifican.
- 2 Una misma cláusula no se debe especificar más de una vez.

Descripción

nombre-conjunto-clases-trabajo

Identifica el conjunto de clases de trabajo que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704).

ADD

Añade una clase de trabajo al conjunto de clases de trabajo. Para obtener detalles, consulte "CREATE WORK CLASS SET".

ALTER

Modifica los atributos de actividad de base de datos y la posición de una clase de trabajo específica en el conjunto de clases de trabajo.

WORK CLASS *nombre-clase-trabajo*

Identifica la clase de trabajo se va a modificar. El *nombre-clase-trabajo* debe identificar una clase de trabajo que exista en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42704).

DROP

Descarta la clase de trabajo del conjunto de clases de trabajo.

WORK CLASS *nombre-clase-trabajo*

Identifica la clase de trabajo se va a descartar. El *nombre-clase-trabajo* debe identificar una clase de trabajo que exista en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42704). Una clase de trabajo no se puede descartar si hay una acción de trabajo en cualquiera de los conjuntos de acciones de trabajo asociados con este conjunto de clases de trabajo que dependa de la misma (SQLSTATE 42893).

cláusula-para-desde-a

FOR

Indica el tipo de información que está especificándose en la cláusula FROM *desde-valor* TO *al-valor*. La cláusula FOR sólo se utiliza para los siguientes tipos de trabajo:

- ALL
- DML
- READ
- WRITE

TIMERONCOST

El coste estimado del trabajo, en acciones de temporización. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

CARDINALITY

La cardinalidad estimada del trabajo. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

FROM desde-valor TO UNBOUNDED o **FROM desde-valor TO al-valor**

Especifica el rango de valor de activación de temporizador (para el coste estimado) o cardinalidad en la que debe caer la actividad de la base de datos si va a formar parte de esta clase de trabajo. El rango incluye *desde-valor* y *al-valor*. Este rango sólo se utiliza para los siguientes tipos de trabajo:

- ALL
- DML
- READ
- WRITE

FROM desde-valor TO UNBOUNDED

desde-valor debe ser cero o un valor DOUBLE positivo (SQLSTATE 5U019). El rango no tiene vinculación superior.

FROM desde-valor TO al-valor

desde-valor debe ser cero o un valor DOUBLE positivo y *al-valor* debe ser un valor DOUBLE positivo. *desde-valor* debe ser igual o más pequeño que *al-valor* (SQLSTATE 5U019).

ALL UNITS UNBOUNDED

Indica que no ha de especificarse ningún rango en la cláusula FROM *desde-valor* TO *al-valor* y que ha de incluirse todo el trabajo caiga dentro del tipo de trabajo especificado.

*cláusula-modificación-esquema***ROUTINES**

Esta cláusula sólo se utiliza si el tipo de trabajo es CALL o ALL y la actividad de base de datos es una sentencia CALL.

IN SCHEMA nombre-esquema

Especifica el nombre de esquema del procedimiento que estará llamando la sentencia CALL.

ALL

Especifica que se incluyan todos los esquemas.

*cláusula-modificación-código-datos***DATA TAG LIST CONTAINS constante-entera**

Especifica el valor del código que se proporciona a cualquier dato que puede tocar la actividad de la base de datos si va a formar parte de esta clase de trabajo. Si no se especifica la cláusula para la clase de trabajo, todo el trabajo que se inscribe dentro del tipo de trabajo especificado, independientemente de los datos que pueda tocar, se incluirá (es decir, el valor por omisión es ignorar el código de datos). Esta cláusula sólo se utiliza si el tipo de trabajo es READ, WRITE, DML o ALL y la actividad de base de datos es una sentencia DML. Los valores válidos para *constante-entero* son enteros del 1 al 9.

DATA TAG LIST CONTAINS ANY

Indica que cualquier valor de código de datos, incluso que no haya código de datos, es válido para la clase de trabajo. Todo el trabajo que se inscribe dentro del tipo de trabajo especificado debe incluirse, independientemente del código de datos.

*cláusula-posición***POSITION**

Especifica el lugar en el que va a colocarse esta clase de trabajo dentro

ALTER WORK CLASS SET

del conjunto de clases de trabajo, lo cual determina el orden en el que se evalúan estas clases de trabajo. Al efectuar la asignación de clases de trabajo en tiempo de ejecución, el gestor de base de datos determina en primer lugar el conjunto de clases de trabajo asociado al objeto, la base de datos o una superclase de servicio. Se selecciona la primera clase de trabajo coincidente en dicho conjunto de clases de trabajo. Si no se especifica esta palabra clave, la clase de trabajo se ubica en la última posición.

LAST

Especifica que la clase de trabajo va a colocarse en última posición en la lista ordenada de las clases de trabajo del conjunto de clases de trabajo.

BEFORE *nombre-clase-trabajo*

Especifica que la clase de trabajo va a colocarse antes de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AFTER *nombre-clase-trabajo*

Especifica que la clase de trabajo va a colocarse detrás de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AT *posición*

Especifica la posición absoluta en la que la carga de trabajo va a colocarse en el conjunto de clases de trabajo en la lista ordenada de clases de trabajo. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42615). Si *posición* es mayor que el número de clases de trabajo existentes más una, la clase de trabajo se sitúa en la última posición del conjunto de clases de trabajo.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.

Ejemplos

- *Ejemplo 1:* Modifique el conjunto de clases de trabajo LARGE_QUERIES y establezca que las dos clases de trabajo existentes tengan un rango que comience en 100 000, conservando el rango sin vincular. Añada una tercera clase de trabajo para todas las sentencias SELECT que tengan un coste de actividades de temporización estimado que sea igual o superior a 10 000 y sitúe esta clase de trabajo para que tenga prioridad sobre las dos clases de trabajo existentes.

```
ALTER WORK CLASS SET LARGE_QUERIES
ALTER WORK CLASS LARGE_ESTIMATED_COST
FOR TIMERONCOST FROM 100000 TO UNBOUNDED
ALTER WORK CLASS LARGE_CARDINALITY
FOR CARDINALITY FROM 100000 TO UNBOUNDED
ADD WORK CLASS LARGE_SELECTS WORK TYPE READ
FOR TIMERONCOST FROM 10000 TO UNBOUNDED POSITION AT 1
```

- *Ejemplo 2:* Modifique un conjunto de clases de trabajo denominado DML_STATEMENTS para añadir una clase de trabajo que represente todas las sentencias DML SELECT que contengan una sentencia DELETE, INSERT, MERGE o UPDATE.

```
ALTER WORK CLASS SET DML_STATEMENTS
ADD WORK CLASS UPDATE_CLASS WORK TYPE WRITE
```

ALTER WORKLOAD

La sentencia ALTER WORKLOAD modifica una carga de trabajo.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

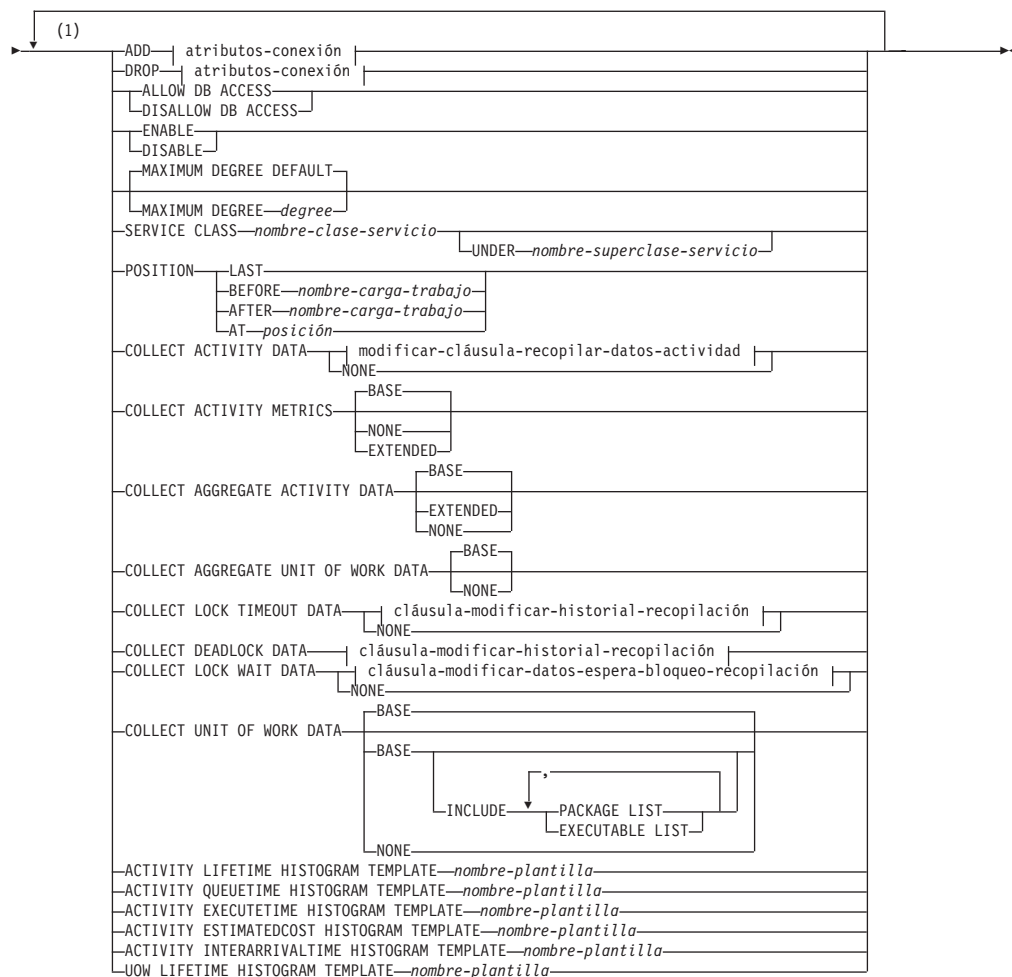
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autoridad SQLADM, únicamente si todas las cláusulas de modificación son cláusulas COLLECT.
- Autorización WLMADM
- Autorización DBADM

Para especificar una cláusula que no sea COLLECT, el ID de autorización de la sentencia debe incluir la autorización DBADM o WLMADM.

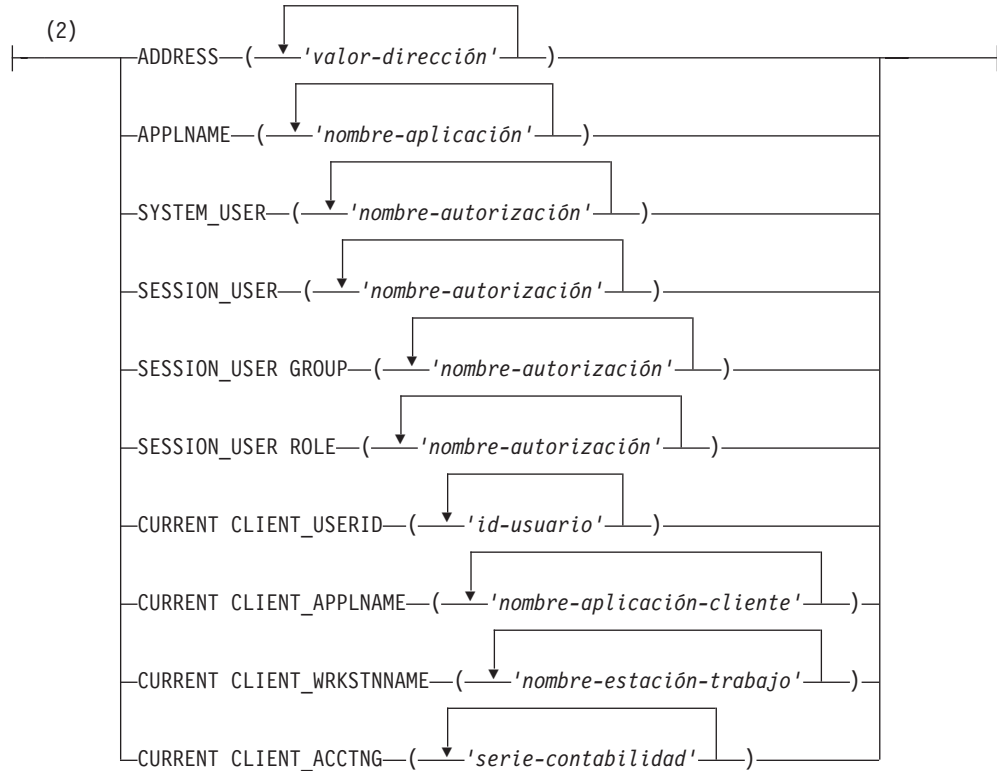
Sintaxis

▶▶ALTER WORKLOAD—*nombre-carga-trabajo*————▶

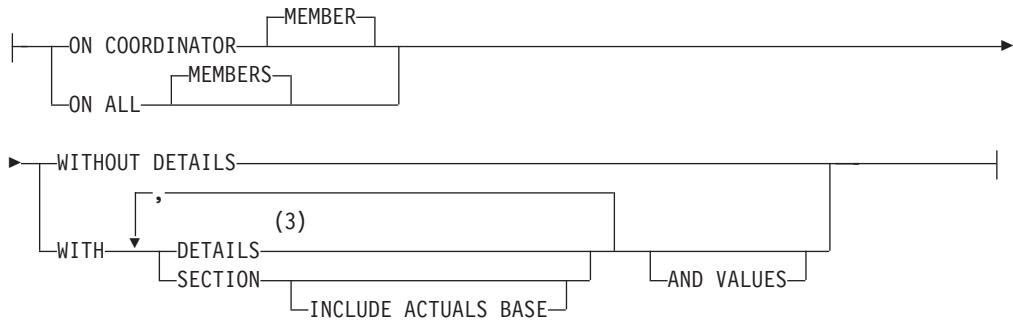


atributos-conexión:

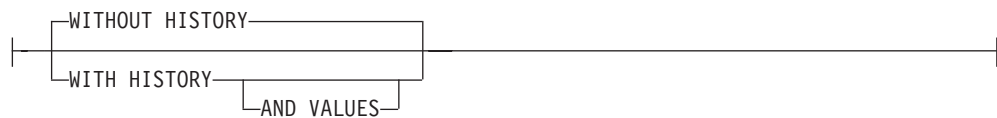
ALTER WORKLOAD



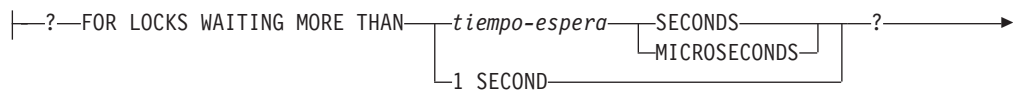
modificar-cláusula-recopilar-datos-actividad:



cláusula-modificar-historial-recopilación:



cláusula-modificar-datos-espera-bloqueo-recopilación:



► | cláusula-modificar-historial-recopilación | ? |

Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.
- 2 Cada cláusula de atributo de conexión sólo puede especificarse una vez.
- 3 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-carga-trabajo

Identifica la carga de trabajo que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704).

ADD *atributos-conexión*

Añade uno o más valores de atributo de conexión a la definición de la carga de trabajo. Los valores de atributo de conexión especificados ya no deben estar definidos para la carga de trabajo (SQLSTATE 5U039). La opción ADD no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

DROP *atributos-conexión*

Descarta uno o más valores de atributo de conexión de la definición de la carga de trabajo. Los valores de atributo de conexión especificados deben estar definidos para la carga de trabajo (SQLSTATE 5U040). La opción DROP no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832). Debe haber como mínimo un valor de atributo de conexión definida. No puede descartarse el último valor de atributo de conexión (SQLSTATE 5U022).

atributos-conexión

Especifica los valores de atributo de conexión para la carga de trabajo. Todos los atributos de conexión distinguen entre mayúsculas y minúsculas, excepto ADDRESS.

ADDRESS ('*valor-dirección*', ...)

Especifica una o más direcciones IPv4, direcciones IPv6 o nombres de dominio seguros para el atributo de conexión ADDRESS. Un valor de dirección no puede aparecer más de una vez en la lista (SQLSTATE 42713). Cada valor de dirección debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor localhost o su representación equivalente 127.0.0.1 no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal. Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no darán como resultado una coincidencia. De modo análogo, localhost o su representación abreviada de IPv6 ::1 no dará como resultado una coincidencia. Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una

dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es `corona.torolab.ibm.com`. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de carga de trabajo si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio.

Al crear un objeto de carga de trabajo, debe especificar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP) donde un dispositivo puede tener una dirección IP cada vez que se conecta con la red.

APPLNAME ('nombre-aplicación', ...)

Especifica una o más aplicaciones para el atributo de conexión de APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si el *nombre-aplicación* no contiene un solo carácter de asterisco (*), equivale al valor que se muestra en el campo "Nombre de la aplicación" en la salida del supervisor del sistema y en la salida del mandato LIST APPLICATIONS. Si *nombre-aplicación* no contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de nombres de aplicación, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la aplicación, utilice una secuencia de dos caracteres de asterisco (**).

SYSTEM_USER ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SYSTEM_USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER GROUP ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER GROUP. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ROLE ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER ROLE. Los roles de un ID de autorización de sesión en este contexto hacen referencia a todos los roles a disposición del ID de autorización de sesión, sin tener en cuenta el modo en que se obtuvieron los roles. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT_CLIENT_USERID ('id-usuario', ...)

Especifica uno o más ID de usuario cliente para el atributo de conexión CURRENT_CLIENT_USERID. Un ID de usuario cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *ID-usuario* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de ID de usuario, donde el asterisco (*) representa

una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el ID de usuario, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT CLIENT_APPLNAME ('nombre-aplicación-cliente', ...)

Especifica una o más aplicaciones para el atributo de conexión de CURRENT CLIENT_APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si el *nombre-aplicación-cliente* no contiene un solo carácter de asterisco (*), equivale al valor que se muestra en el campo “Nombre de aplicación cliente TP Monitor” en la salida del supervisor del sistema. Si *nombre-aplicación-cliente* no contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de nombres de aplicación, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la aplicación, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT CLIENT_WRKSTNNAME ('nombre-estación-trabajo', ...)

Especifica uno o más nombres de estación de trabajo cliente para el atributo de conexión CURRENT CLIENT_WRKSTNNAME. Un nombre de estación de trabajo cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *nombre-estación-trabajo* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de nombres de estación de trabajo, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la estación de trabajo, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT CLIENT_ACCTNG ('serie-contabilidad', ...)

Especifica uno o más series de contabilidad cliente para el atributo de conexión CURRENT CLIENT_ACCTNG. Una serie de contabilidad cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *serie-contabilidad* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de series de contabilidad, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en la serie de contabilidad, utilice una secuencia de dos caracteres de asterisco (**).

ALLOW DB ACCESS o DISALLOW DB ACCESS

Especifica si se permitirá o no a la aparición de carga de trabajo asociada a esta carga de trabajo acceder a la base de datos.

ALLOW DB ACCESS

Especifica que se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos.

DISALLOW DB ACCESS

Especifica que no se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos. La siguiente unidad de trabajo asociada a esta carga de trabajo se rechazará (SQLSTATE 5U020). Se permite que se completen las apariciones de carga de trabajo que ya se estén ejecutando. Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

ENABLE o DISABLE

Especifica si esta carga de trabajo se tomará o no en consideración cuando se seleccione una carga de trabajo.

ALTER WORKLOAD

ENABLE

Especifica que la carga de trabajo se habilitará y se tomará en consideración cuando se seleccione una carga de trabajo.

DISABLE

Especifica que la carga de trabajo se inhabilitará y no se tomará en consideración cuando se seleccione una carga de trabajo. Esta opción no puede especificarse si *nombre-carga-trabajo* es SYSDEFAULTUSERWORKLOAD o SYSDEFAULTADMWORKLOAD (SQLSTATE 42832).

MAXIMUM DEGREE

Especifica el grado de ejecución máximo de paralelismo para esta carga de trabajo. El atributo MAXIMUM DEGREE no se puede modificar si *nombre-carga-trabajo* es SYSDEFAULTADMWORKLOAD.

DEFAULT

Especifica que esta carga de trabajo hereda el valor de paralelismo intrapartición del parámetro de configuración del gestor de bases de datos **intra_parallel**. Cuando **intra_parallel** se establece en NO, la carga de trabajo se ejecuta con el paralelismo intrapartición inhabilitado. Cuando **intra_parallel** se establece en YES, la carga de trabajo se ejecuta con el paralelismo intrapartición habilitado. La carga de trabajo no especifica un grado de tiempo de ejecución máximo para las aplicaciones asignadas. Por tanto, el grado de tiempo de ejecución real se determina como el más bajo de los valores del parámetro de configuración **max_querydegree**, el valor establecido por el comando SET RUNTIME DEGREE y el grado de compilación de la sentencia de SQL.

grado

Especifica el grado máximo de paralelismo para esta carga de trabajo. Los valores válidos son 1 a 32.767. Con valor 1, las peticiones asociadas se ejecutan con el paralelismo intrapartición inhabilitado. Con un valor de 2 a 32.767, las peticiones asociadas se ejecutan con el paralelismo intrapartición habilitado. El grado de tiempo de ejecución real se determina como el más bajo de este *grado*, el valor del parámetro de configuración **max_querydegree**, el valor establecido por el comando SET RUNTIME DEGREE y el grado de compilación de la sentencia de SQL.

SERVICE CLASS *nombre-clase-servicio*

Especifica que las peticiones asociadas a esta carga de trabajo se ejecuten en la clase de servicio *nombre-clase-servicio*. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-clase-servicio* no puede ser 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032). Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

UNDER *nombre-superclase-servicio*

La cláusula se utiliza al especificar una subclase de servicio. El *nombre-superclase-servicio* identifica la superclase de servicio de *nombre-superclase-servicio*. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no puede ser 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032).

POSITION

Especifica el lugar en el que va a situarse esta carga de trabajo en la lista ordenada de cargas de trabajo. En tiempo de ejecución, se examina esta lista

por orden para buscar la primera carga de trabajo que coincida con los atributos de conexión necesarios. Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

LAST

Especifica que la carga de trabajo va a ser la última de la lista, antes de las cargas de trabajo por omisión SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

BEFORE *nombre-carga-trabajo-relativa*

Especifica que la carga de trabajo va a colocarse antes de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción BEFORE no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AFTER *nombre-carga-trabajo-relativa*

Especifica que la carga de trabajo va a colocarse detrás de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción AFTER no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AT *posición*

Especifica la posición absoluta en la que la carga de trabajo va a colocarse en la lista. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42615). Si *posición* es mayor que el número de cargas de trabajo existentes más una, la carga de trabajo se sitúa en la última posición justo antes de SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad asociada a esta carga de trabajo se enviarán al supervisor de sucesos de actividades activas cuando la actividad finalice.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada con esta carga de trabajo se deben enviar a cualquier supervisor de actividades activas, cuando la actividad finalice su ejecución. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. Los datos reales de sección sólo se recopilarán en cualquier miembro donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actuals**.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula INCLUDE ACTUALS, el parámetro de configuración de base de datos **section_actuals** y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del parámetro de configuración de la base de datos **section_actuals** es NONE y <collectsectionactuals> se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

BASE especifica que han de habilitarse y recopilarse los datos reales siguientes durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

NONE

Especifica que los datos de actividad no se recopilan para cada actividad que se asocie con esta carga de trabajo.

COLLECT ACTIVITY METRICS

Especifica que la métrica del supervisor debe recopilarse para una actividad enviada por una ocurrencia de la carga de trabajo. El valor por omisión es COLLECT ACTIVITY METRICS NONE.

La configuración efectiva de la recopilación de métrica de actividad es la combinación del atributo especificado por la cláusula COLLECT ACTIVITY METRICS de la carga de trabajo que envía la petición y el parámetro de configuración de base de datos MON_ACT_METRICS. Si ni el atributo de carga de trabajo ni el parámetro de configuración tienen un valor distinto de NONE, se recopilará la métrica para la actividad.

NONE

Especifica que no se recopilará ninguna métrica para cualquier actividad enviada por una ocurrencia de la carga de trabajo.

BASE

Especifica que la métrica básica se recopilará para cualquier actividad enviada por una ocurrencia de la carga de trabajo.

EXTENDED

Especifica que la métrica básica se recopilará para cualquier actividad enviada por una ocurrencia de la carga de trabajo. Asimismo, especifica que los valores de los siguientes elementos de supervisor deben determinarse con granularidad adicional:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividades agregadas sobre las actividades asociadas con esta carga de trabajo deben enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. Cuando se especifica COLLECT AGGREGATE ACTIVITY DATA, el valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE.

BASE

Especifica que los datos de actividades agregadas básicas sobre las actividades asociadas con esta carga de trabajo se deben enviar al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal. Sólo las actividades a las que se ha aplicado un umbral SQLTEMPSPACE participan en esta marca de límite superior.
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregadas sobre las actividades asociadas con esta carga de trabajo deben enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

NONE

Especifica que no debe recopilarse ningún dato de actividad agregada para esta carga de trabajo.

ALTER WORKLOAD

COLLECT AGGREGATE UNIT OF WORK DATA

Especifica que se enviarán al supervisor de sucesos estadísticos, si hay alguno activo, los datos agregados de unidad de trabajo correspondientes a las unidades de trabajo asociadas a esta carga de trabajo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos `wlm_collect_int`. El valor por omisión cuando se especifica COLLECT AGGREGATE UNIT OF WORK DATA es COLLECT AGGREGATE UNIT OF WORK DATA BASE.

BASE

Especifica que se enviarán al supervisor de sucesos estadísticos, si hay alguno activo, los datos agregados básicos de unidad de trabajo correspondientes a las unidades de trabajo asociadas a esta carga de trabajo. Los datos agregados básicos de unidad de trabajo incluyen:

- Unidad de histograma de ciclo de vida de trabajo

NONE

Especifica que no debe recopilarse ningún dato agregado de unidad de trabajo para esta carga de trabajo.

COLLECT LOCK TIMEOUT DATA

Especifica que los datos sobre los sucesos de tiempo de espera de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. Los datos de tiempo de espera de bloqueo se recopilan en todos los miembros. Este valor funciona conjuntamente con el valor de configuración de base de datos `MON_LOCKWAIT`. Se respeta la configuración que genera la salida más detallada.

cláusula-modificar-historial-recopilación

WITHOUT HISTORY

Especifica que los datos sobre los sucesos de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. El historial de actividades anteriores y los valores de entrada no se envían al supervisor de sucesos.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos los sucesos de bloqueo de este tipo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro `DB2_MAX_INACT_STMTS` para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB, datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación

REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

NONE

Especifica que los datos de tiempo de espera de bloqueo para la carga de trabajo no se recopilan en ningún miembro.

COLLECT DEADLOCK DATA

Especifica que los datos sobre los sucesos de punto muerto que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. Los datos de tiempo de espera de bloqueo se recopilan en todos los miembros. Este valor sólo se respetará si el parámetro de configuración de base de datos **MON_DEADLOCK** está establecido en NONE.

cláusula-modificar-historial-recopilación**WITHOUT HISTORY**

Especifica que los datos sobre los sucesos de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. El historial de actividades anteriores y los valores de entrada no se envían al supervisor de sucesos.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos estos tipos de sucesos de bloqueo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro **DB2_MAX_INACT_STMTS** para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB, datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

COLLECT LOCK WAIT DATA

Especifica que los datos sobre los sucesos de espera de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando el bloqueo no se haya adquirido en el *tiempo-espera*. Este valor funciona conjuntamente con el valor del parámetro de configuración de base de datos **mon_lockwait** y **mon_lw_thresh**. Se respeta la configuración que genera la salida más detallada.

cláusula-modificar-datos-espera-bloqueo-recopilación

FOR LOCKS WAITING MORE THAN *tiempo-espera* SECONDS | MICROSECONDS) | 1 SECOND

Especifica que los datos sobre los sucesos de espera de bloqueo que se produzcan en esta carga de trabajo se envían al supervisor de sucesos aplicable cuando el bloqueo no se haya adquirido en el *tiempo-espera*.

Este valor puede ser cualquier entero no negativo. Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *tiempo-espera*. El valor válido mínimo para el parámetro *tiempo-espera* es 1000 microsegundos.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos los sucesos de bloqueo de este tipo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro DB2_MAX_INACT_STMTS para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB, datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

NONE

Especifica que el suceso de espera de bloqueo para la carga de trabajo no se recopila en ningún miembro.

COLLECT UNIT OF WORK DATA

Especifica que los datos sobre cada unidad de trabajo, también conocida como transacción, asociada con esta carga de trabajo deben enviarse a los supervisores de sucesos de la unidad de trabajo, si hay alguno activo, cuando finalice la unidad de trabajo. El valor por omisión es COLLECT UNIT OF WORK BASE. Si el parámetro de configuración de base de datos **mon_uow_data** se ha establecido en BASE, este tiene prioridad respecto al parámetro COLLECT UNIT OF WORK DATA. El valor NONE para **mon_uow_data** indica que se utilizan los parámetros COLLECT UNIT OF WORK DATA de cargas de trabajo individuales.

BASE

Especifica que se envíe el nivel básico de datos para las transacciones asociadas con esta carga de trabajo al supervisor de sucesos de unidad de trabajo.

Parte de la información notificada en un suceso de unidad de trabajo corresponde a métricas de petición de nivel del sistema. La recopilación de estas métricas se controla independientemente de la recopilación de datos

de la unidad de trabajo. Las métricas de petición se controlan con la cláusula COLLECT REQUEST METRICS de la superclase o bien mediante la utilización del parámetro de configuración de base de datos **mon_req_metrics**. La superclase de servicio con la que está asociada la carga de trabajo, o la superclase de servicio de la subclase de servicio con la que está asociada la carga de trabajo, debe tener habilitada la recopilación de métricas de petición para que las métricas de petición estén presentes en el suceso de unidad de trabajo. Si la recopilación de métricas de petición no está habilitada, el valor de las métricas de petición será cero.

INCLUDE PACKAGE LIST

Especifica que han de enviarse el nivel básico de datos y la lista de paquetes de las transacciones que se asocian a esta carga de trabajo al supervisor de sucesos de unidad de trabajo.

El tamaño de la lista de paquetes recopilados lo determina el valor del parámetro de configuración de base de datos **mon_pkglist_sz**. Si este valor es 0, la lista de paquetes no se recopilará, aunque se haya especificado la opción PACKAGE LIST.

En un entorno de base de datos particionada, la lista de paquetes sólo está disponible en el miembro coordinador. El nivel BASE se recopilará en los miembros remotos.

Parte de la información notificada en un suceso de unidad de trabajo corresponde a métricas de petición de nivel del sistema. La recopilación de estas métricas se controla independientemente de la recopilación de datos de la unidad de trabajo. Las métricas de petición se controlan con la cláusula COLLECT REQUEST METRICS de la superclase o bien mediante la utilización del parámetro de configuración de base de datos **mon_req_metrics**. La superclase de servicio con la que está asociada la carga de trabajo, o la superclase de servicio de la subclase de servicio con la que está asociada la carga de trabajo, debe tener habilitada la recopilación de métricas de petición para que las métricas de petición estén presentes en el suceso de unidad de trabajo. Si la recopilación de métricas de petición no está habilitada, el valor de las métricas de petición será cero.

INCLUDE EXECUTABLE LIST

Especifica que se recopilará la lista de ID ejecutables para una unidad de trabajo junto con el nivel básico de datos y se enviarán al supervisor de sucesos de la unidad de trabajo.

NONE

Especifica que ninguna unidad de datos de trabajo para las transacciones asociadas con esta carga de trabajo se envía al supervisor de sucesos de unidad de trabajo.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos, de las actividades de DB2 en ejecución en la carga de trabajo durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la carga de trabajo están en cola durante un intervalo

ALTER WORKLOAD

específico. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la carga de trabajo están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en el miembro coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML en ejecución en la carga de trabajo. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML a esta carga de trabajo y la llegada de la siguiente actividad DML a esta carga de trabajo. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

UOW LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma que se utiliza para recopilar datos estadísticos sobre la duración, en milisegundos, de unidades de trabajo que se ejecutan en la carga de trabajo durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE UNIT OF WORK DATA con la opción BASE.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)

- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia. Para apariciones de cargas de trabajo recién emitidas, los cambios surten efecto después de confirmar la sentencia ALTER WORKLOAD. Para las apariciones de carga de trabajo activas, los cambios surtirán efecto al principio de la siguiente unidad de trabajo.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Si se especifica la opción DISABLE, la carga de de trabajo se inhabilitará después de las confirmaciones de sentencia. La carga de trabajo no se tendrá en cuenta la próxima vez que se seleccione una carga de trabajo. Si hay una aparición de carga de trabajo activa asociada con esta carga de trabajo cuando se confirma la sentencia ALTER WORKLOAD, sigue ejecutándose hasta el final de la unidad de trabajo actual. Al principio de la siguiente unidad de trabajo, tendrá lugar una reevaluación de carga de trabajo y la conexión pasará a estar asociada a una carga de trabajo diferente.
- *Privilegios*: el privilegio USAGE no se otorga a ningún usuario, grupo o rol cuando se crea una carga de trabajo. Para habilitar el uso de una carga de trabajo, otorgue el privilegio USAGE de esa carga de trabajo a un usuario, grupo o rol mediante la sentencia GRANT USAGE ON WORKLOAD.
- *Alternativas de sintaxis*: Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse COLLECT UNIT OF WORK DATA PACKAGE LIST en lugar de COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST.

Ejemplos

- *Ejemplo 1*: La carga de trabajo PAYROLL se ha colocado en la actualidad de modo que la carga de trabajo INVENTORY se tome en consideración en primer lugar cuando DB2 seleccione una carga de trabajo en tiempo de ejecución. Modifique el orden de evaluación para que PAYROLL se tome en consideración en primer lugar.

```
ALTER WORKLOAD PAYROLL
  POSITION BEFORE INVENTORY
```

- *Ejemplo 2*: Modifique el orden de evaluación de modo que la carga de trabajo BENCHMARK se evalúe por medio de DB2 antes de cualquier carga de trabajo del catálogo.

```
ALTER WORKLOAD BENCHMARK
  POSITION AT 1
```

ALTER WORKLOAD

- *Ejemplo 3:* Se ha creado la carga de trabajo REPORTS con APPLNAME establecido en app11, app12 y app13 y SYSTEM_USER establecido en BOB y MARY. Modifique la carga de trabajo para añadir una aplicación nueva, app14 a la lista de nombres de aplicación y elimine app12, ya que no debería correlacionarse con REPORTS.

```
ALTER WORKLOAD REPORTS
  ADD APPLNAME ('app14')
  DROP APPLNAME ('app12')
```

- *Ejemplo 4:* suponiendo que existe un supervisor de sucesos llamado LOCK y que está activo, crear registros de sucesos de bloqueo con el historial de sentencias para los sucesos de tiempo de espera de bloqueo que tengan lugar dentro de la carga de trabajo APP.

```
ALTER WORKLOAD APP
  COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

- *Ejemplo 5:* suponiendo que existe un supervisor de sucesos de bloqueo denominado LOCK y que está activo, crear registros de sucesos de bloqueo sólo para sucesos de tiempo de espera de punto muerto y bloqueo que tengan lugar en la carga de trabajo PAYROLL en todas las particiones.

```
ALTER WORKLOAD PAYROLL
  COLLECT DEADLOCK DATA
  COLLECT LOCK TIMEOUT DATA WITHOUT HISTORY
```

- *Ejemplo 6:* suponiendo que existe un supervisor de sucesos de bloqueo denominado LOCK y que está activo, crear registros de sucesos de bloqueo con un historial de sentencias para sucesos de punto muerto que tengan lugar dentro de la carga de trabajo INVOICE.

```
ALTER WORKLOAD
INVOICE
  COLLECT DEADLOCK DATA WITH HISTORY AND VALUES
```

- *Ejemplo 7:* suponiendo que existe un supervisor de sucesos de bloqueo denominado LOCK y que está activo, crear registros de sucesos de bloqueo con un historial de sentencias y valores para bloqueos adquiridos después de esperar más de 150 milisegundos dentro de la carga de trabajo INVOICE.

```
ALTER WORKLOAD
INVOICE
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 150000
  MICROSECONDS WITH HISTORY AND VALUES
```

- *Ejemplo 8:* modificar la carga de trabajo REPORTS para recopilar datos de la unidad de trabajo y enviarlos al supervisor de sucesos de la unidad de trabajo:

```
ALTER WORKLOAD REPORTS
  COLLECT UNIT OF WORK DATA BASE
```

ALTER WRAPPER

La sentencia ALTER WRAPPER se utiliza para actualizar las propiedades de un derivador.

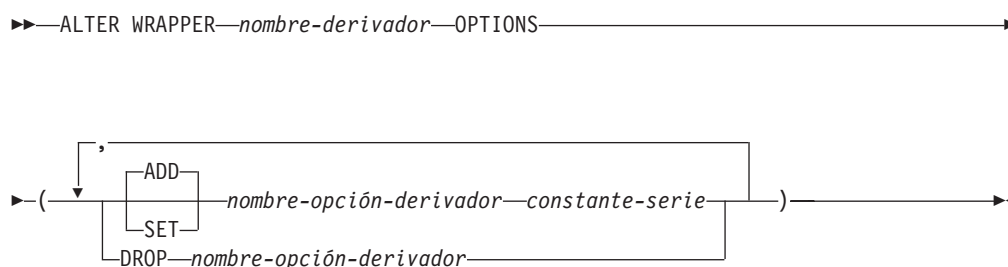
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

Sintaxis



Descripción

nombre-derivador

Especifica el nombre del derivador.

OPTIONS

Indica que se deben habilitar, restaurar o descartar las opciones de derivador.

ADD

Habilita una opción de servidor.

SET

Cambia el valor de una opción de derivador.

nombre-opción-derivador

Nombra una opción de derivador que se debe habilitar o restaurar. Actualmente el único nombre de opción de derivador soportado es DB2_FENCED.

constante-serie

Especifica el valor para *nombre-opción-derivador* como una constante de serie de caracteres. Los valores válidos son 'Y' o 'N'. El valor por omisión para los derivadores relacionales es 'N', y el valor por omisión para los derivadores no relacionales es 'Y'.

DROP *nombre-opción-derivador*

Descarta una opción de derivador.

ALTER WRAPPER

Notas

- La ejecución de la sentencia ALTER WRAPPER no incluye la comprobación de validez de las opciones específicas de derivador.
- Una sentencia ALTER WRAPPER de una unidad de trabajo (UOW) determinada no puede procesarse (SQLSTATE 55007) si la UOW ya incluye uno de los siguientes elementos:
 - Una sentencia SELECT que hace referencia a un apodo que pertenece al derivador.
 - Un cursor abierto en un apodo que pertenece al derivador.
 - Una sentencia INSERT, DELETE o UPDATE emitida para un apodo que pertenece al derivador.

Ejemplo

Activar la opción DB2_FENCED para el derivador NET8.

```
ALTER WRAPPER NET8 OPTIONS (SET DB2_FENCED 'Y')
```

ALTER XSROBJECT

Esta sentencia se utiliza para habilitar o inhabilitar el soporte de la descomposición para un esquema XML específico. Los esquemas XML anotados se pueden utilizar para descomponer documentos XML en tablas relacionales, si se ha habilitado la descomposición para dichos esquemas XML.

Invocación

La sentencia ALTER XSROBJECT se puede incorporar a un programa de aplicación o emitir a través del uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Se necesita una de las autorizaciones siguientes:

- DBADM
- ALTERIN para el esquema de SQL
- Propiedad del objeto XSR que se va a modificar

Sintaxis

```

▶▶ ALTER XSROBJECT nombre-objetoXSR {
    ENABLE DECOMPOSITION
  | DISABLE DECOMPOSITION
}

```

Descripción

nombre-objetoXSR

Identifica el objeto XSR que se va a modificar. El *nombre-objetoXSR* (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este identificador, se devolverá un error (SQLSTATE 42704).

ENABLE DECOMPOSITION o DISABLE DECOMPOSITION

Habilita o inhabilita el uso del objeto XSR para la descomposición. El objeto XSR identificado debe ser un esquema XML (SQLSTATE 42809). Para habilitar la descomposición, es preciso anotar el esquema XML con normas de descomposición (SQLSTATE 225DE) y los objetos a los que dichas normas hacen referencia deben existir en el servidor actual (SQLSTATE 42704).

Notas

- Cuando se inhabilita la descomposición de un objeto XSR, se eliminan todas las entradas de catálogo relacionadas.
- El soporte de descomposición de un objeto XSR se inhabilitará si se descarta o se modifica algún objeto del que dependa el objeto XRS (como por ejemplo tablas) para convertirlo en incompatible con el objeto XSR.
- En un entorno de base de datos particionada, se puede emitir esta sentencia conectándose a cualquier partición.

ASSOCIATE LOCATORS

La sentencia ASSOCIATE LOCATORS obtiene el valor localizador de cada conjunto de resultados devuelto por un procedimiento.

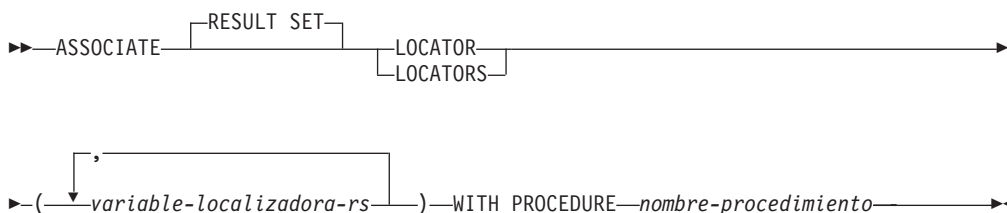
Invocación

Esta sentencia sólo puede incorporarse en un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

variable-localizadora-cr

Especifica una variable localizadora de conjuntos de resultados que se ha declarado en una sentencia de SQL compuesto (procedimiento).

WITH PROCEDURE

Identifica el procedimiento que devuelve localizadores de conjuntos de resultados de acuerdo con el nombre de procedimiento especificado.

nombre-procedimiento

Un nombre de procedimiento es un nombre calificado o no calificado.

Un nombre de procedimiento totalmente calificado consta de dos partes. La primera parte es un identificador que contiene el nombre de esquema del procedimiento. La última parte es un identificador que contiene el nombre del procedimiento. 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. En el párrafo siguiente se describen las normas para especificar un nombre de procedimiento.

Cuando se ejecuta la sentencia ASSOCIATE LOCATORS, el nombre o especificación del procedimiento debe identificar un procedimiento que el

petionario 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.

Notas

- Si el número de variables localizadoras de conjuntos de resultados que aparecen en la sentencia ASSOCIATE LOCATORS es menor que el número de localizadores devueltos por el procedimiento, todas las variables de la sentencia se asignan a un valor, y se emite un aviso.
- Si el número de variables localizadoras de conjuntos de resultados que aparecen en la sentencia ASSOCIATE LOCATORS es mayor que el número de localizadores devueltos por el procedimiento, se asigna el valor 0 a las variables sobrantes.
- Si un mismo llamador invoca un procedimiento más de una vez, sólo son accesibles los conjuntos de resultados más recientes.
- Los valores de localizadora de conjunto de resultados están disponibles para un procedimiento que se llame utilizando una sentencia EXECUTE que ejecuta la sentencia CALL preparada con anterioridad por medio de la sentencia PREPARE. Sin embargo, los valores de localizadora de conjunto de resultados no están disponibles para un procedimiento que se llame utilizando una sentencia EXECUTE IMMEDIATE.
- Los nombres de procedimientos de módulo a los que se hace referencia en una sentencia ASSOCIATE LOCATORS solo pueden ser referencias de nombre calificado de 1 ó 2 partes. No se permiten las referencias de nombre de 3 partes (SQLSTATE 42601). Las sentencias CALL que hacen referencia a un procedimiento-módulo al que se hacía referencia en una sentencia ASSOCIATE LOCATORS deben especificar el procedimiento-módulo con el mismo nombre calificado de 1 ó 2 partes utilizado en la sentencia ASSOCIATE LOCATORS.

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 de resultados LOC1 y LOC2 para obtener los valores de los dos conjuntos de resultados devueltos por el procedimiento P1. Se supone que el procedimiento 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 para el procedimiento para asegurar que se utilice ese procedimiento P1 del esquema MYSCHEMA.

```
CALL MYSCHEMA.P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
WITH PROCEDURE MYSCHEMA.P1;
```

AUDIT

La sentencia AUDIT determina la política de auditoría que se va a utilizar para una base de datos concreta u objeto de base de datos en el servidor actual. Cada vez que se está utilizando el objeto, éste se auditará con arreglo a dicha política.

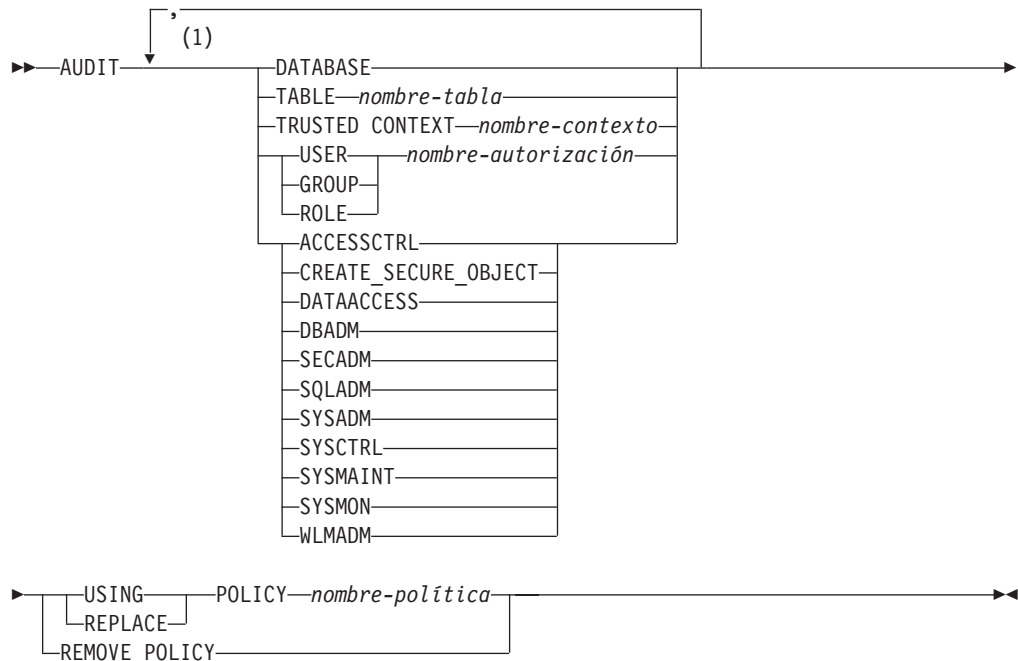
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Notas:

- 1 Cada cláusula (con el mismo nombre de objeto, si es de aplicación) puede especificarse como máximo una vez (SQLSTATE 42713).

Descripción

ACCESSCTRL, CREATE_SECURE_OBJECT, DATAACCESS, DBADM, SECADM, SQLADM, SYSADM, SYSCTRL, SYSMANT, SYSMON o WLMADM

Especifica que una política de auditoría va a asociarse o eliminarse de la autorización especificada. Todos los sucesos auditable que se inician por medio de un usuario que tiene la autorización especificada, incluso en el caso

de que dicha autorización no se necesite para el suceso, se auditarán con arreglo a la política de auditoría asociada.

DATABASE

Especifica que una política de auditoría va a asociarse o eliminarse de la base de datos del servidor actual. Se comprueban todos los sucesos auditables que se produzcan en la base de datos con arreglo a la política de auditoría asociada.

TABLE *nombre-tabla*

Especifica que una política de auditoría va a asociarse o eliminarse de *nombre-tabla*. El *nombre-tabla* debe identificar una tabla, tabla de consulta materializada (MQT) o apodo que exista en el servidor actual (SQLSTATE 42704). No puede ser una vista, una tabla de catálogo, una tabla temporal creada, una tabla temporal declarada (SQLSTATE 42995) ni una tabla con tipo (SQLSTATE 42997). Cuando se acceda a la tabla sólo se accederá a los sucesos de auditoría de la categoría EXECUTE con o sin datos, incluso si la política indica que deberían auditarse otras categorías.

TRUSTED CONTEXT *nombre-contexto*

Especifica que una política de auditoría va a asociarse o eliminarse de *nombre-contexto*. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). Todos los sucesos auditables que se produzcan en la conexión fiable definida por medio del contexto fiable *nombre-contexto* se auditarán con arreglo a la política de auditoría asociada.

USER *nombre-autorización*

Especifica que una política de auditoría va a asociarse o eliminarse del usuario que tiene ID de autorización *nombre-autorización*. Todos los sucesos auditables que se inician por medio del *nombre-autorización* se auditarán con arreglo a la política de auditoría asociada.

GROUP *nombre-autorización*

Especifica que una política de auditoría va a asociarse o eliminarse del grupo que tiene ID de autorización *nombre-autorización*. Todos los sucesos auditables que se inician por medio de usuarios que son miembros del *nombre-autorización* se auditarán con arreglo a la política de auditoría asociada. Si la pertenencia de un usuario a un grupo no puede determinarse, la política no se aplicará a dicho usuario.

ROLE *nombre-autorización*

Especifica que una política de auditoría va a asociarse o eliminarse del rol que tiene ID de autorización *nombre-autorización*. El *nombre-autorización* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). Todos los sucesos auditables que se inician por medio de usuarios que son miembros del *nombre-autorización* se auditarán con arreglo a la política de auditoría asociada. La pertenencia al rol indirecta por medio de otros roles o grupos es válida.

USING, REMOVE o REPLACE

Especifica si debe utilizarse, eliminarse o sustituirse la política de auditoría para el objeto especificado.

USING

Especifica que la política de auditoría va a utilizarse para el objeto especificado. No debe haberse definido una política de auditoría existente para el objeto (SQLSTATE 5U041). Si ya existe una política de auditoría, ésta debe eliminarse o sustituirse.

REMOVE

Especifica que la política de auditoría va a eliminarse para el objeto

especificado. La utilización del objeto ya no se auditará con arreglo a la política de auditoría. La asociación se suprimirá del catálogo cuando se elimina la política de auditoría del objeto.

REPLACE

Especifica que la política de auditoría va a sustituir una política de auditoría existente para el objeto especificado. Esta acción combina las opciones REMOVE y USING en un paso para asegurar que no hay ningún período de tiempo en el que una política de auditoría no se aplique al objeto especificado. Si una política no se estaba utilizando para el objeto especificado, REPLACE es equivalente a USING.

POLICY *nombre-política*

Especifica la política de auditoría que se va a utilizar para determinar los valores de auditoría. El *nombre-política* debe identificar una política de auditoría existente en el servidor actual (SQLSTATE 42704).

Normas

- Una sentencia de SQL exclusiva de AUDIT debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de AUDIT son:
 - AUDIT
 - CREATE AUDIT POLICY, ALTER AUDIT POLICY o DROP (AUDIT POLICY)
 - DROP (ROLE o TRUSTED CONTEXT si está asociada a una política de auditoría)
- Una sentencia de SQL exclusiva de AUDIT no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.
- Un objeto que puede estar asociado con no más de una política (SQLSTATE 5U042).

Notas

- Los cambios se graban en el catálogo, pero no surten efecto hasta después de que se ejecute una sentencia COMMIT.
- Los cambios no surten efecto hasta la siguiente unidad de trabajo que haga referencia al objeto al que se aplica la política de auditoría. Por ejemplo, si se utiliza la política de auditoría para la base de datos, ninguna unidad de trabajo actual comenzará la auditoría con arreglo a la política hasta después de que se complete una sentencia COMMIT o ROLLBACK.
- Las vistas que acceden a una tabla asociada con una política de auditoría se comprueban con arreglo a la política de la tabla subyacente.
- La política de auditoría que se aplica a una tabla no se aplica a una tabla de consulta materializada (MQT) basada en dicha tabla. Resulta recomendable que si se asocia una política de auditoría con una tabla, también se asocie dicha política con cualquier MQT basada en dicha tabla. Es posible que el compilador utilice automáticamente una MQT, aún en el caso de que una sentencia de SQL haga referencia a la tabla base; sin embargo, estará vigente la política de auditoría que se utilice para la tabla base.
- Cuando se realice una operación de usuario de conmutador en un contexto fiable, se vuelven a evaluar todas las políticas de auditoría con arreglo al usuario nuevo y no se utiliza ninguna política del usuario antiguo para la sesión actual. Esta acción se aplica específicamente a las políticas de auditoría asociadas directamente con el usuario, las pertenencias al rol o al grupo del usuario y las autorizaciones del usuario. Por ejemplo, si se auditó la sesión actual debido a

que el usuario anterior era un miembro de un rol auditado y el usuario conmutado no es un miembro de dicho rol, la política ya no se aplicará a la sesión.

- Cuando se ejecuta una sentencia SET SESSION USER, las políticas de auditoría asociadas al usuario original (y las autorizaciones y pertenencias al grupo y rol del usuario) se combinan con las políticas asociadas al usuario especificado en la sentencia SET SESSION USER. Las políticas de auditoría asociadas al usuario original siguen vigentes, al igual que las políticas para el usuario especificado en la sentencia SET SESSION USER. Si se emiten diversas sentencias SET SESSION USER en una sesión, sólo se toman en consideración las políticas de auditoría asociadas al usuario original y al usuario actual.
- Si se descarta el objeto al que está asociada una política de auditoría, la asociación con la política de auditoría se elimina del catálogo y ya no existirá. Si se vuelve a crear dicho objeto con posterioridad, el objeto no se auditará con arreglo a la política asociada con el mismo cuando se descartó el objeto.

Ejemplos

- *Ejemplo 1:* Utilice la política de auditoría DBAUDPRF para determinar los valores de auditoría para la base de datos en el servidor actual.

```
AUDIT DATABASE USING POLICY DBAUDPRF
```

- *Ejemplo 2:* Elimine la política de auditoría de la tabla EMPLOYEE.

```
AUDIT TABLE EMPLOYEE REMOVE POLICY
```

- *Ejemplo 3:* Utilice la política de auditoría POWERUSERS para determinar los valores de auditoría para las autorizaciones SYSADM, DBADM y SECADM, así como el grupo DBAS.

```
AUDIT SYSADM, DBADM, SECADM, GROUP DBAS USING POLICY POWERUSERS
```

- *Ejemplo 4:* Sustituya la política de auditoría por el rol TELLER por la política nueva TELLERPRF.

```
AUDIT ROLE TELLER REPLACE POLICY TELLERPRF
```

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 incorporarse en un programa de aplicación. No es una sentencia ejecutable. No se debe especificar 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 normas 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`.

Normas

- Las sentencias `BEGIN DECLARE SECTION` y `END DECLARE SECTION` deben especificarse por pares y no pueden anidarse.
- No pueden incorporarse sentencias de SQL dentro de la sección de declaración.
- Las variables a las que se hace referencia en las sentencias de SQL se deben declarar 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. En general, las variables de 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 declaradas fuera de una sección de declaración no deben tener el mismo nombre que las variables declaradas dentro de una 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 del 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.

```
EXEC SQL BEGIN DECLARE SECTION;
       short hv_smint;
       struct {
         short hv_vchar24_len;
         char  hv_vchar24_value[24];
       } hv_vchar24;
```

BEGIN DECLARE SECTION

```
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 bytes, debe utilizarse una variable del lenguaje principal CLOB.

- *Ejemplo 4:* Defina las variables de 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

La sentencia CALL llama a un procedimiento o a un procedimiento externo.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. Cuando se invoca utilizando el procesador de línea de mandatos, existen algunas normas adicionales para la especificación de los argumentos del procedimiento. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

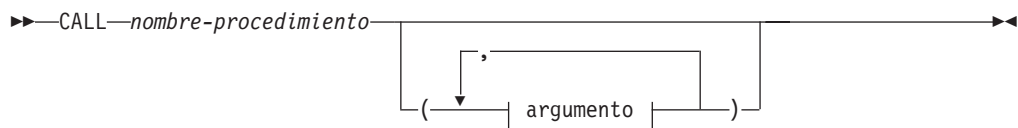
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

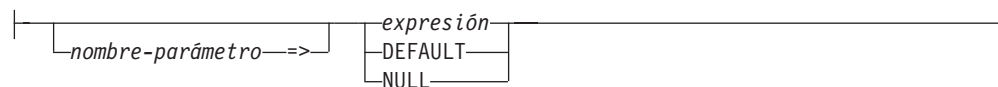
- Privilegio EXECUTE para el procedimiento
- Autorización DATAACCESS

Si existe un procedimiento coincidente para el que el ID de autorización de la sentencia no tiene autorización de ejecución, se devolverá un error (SQLSTATE 42501).

Sintaxis



Argumento:



Descripción

nombre-procedimiento

Especifica el procedimiento que va a llamarse. Debe ser un procedimiento que esté descrito en el catálogo o que esté declarado en el ámbito de la sentencia de SQL compuesto (compilado) que incluye la sentencia CALL. El procedimiento específico que debe invocarse se elige mediante la utilización de la resolución de procedimiento. (Para obtener más detalles, consulte el apartado "Notas" de esta sentencia.)

argumento

nombre-parámetro

Nombre del parámetro al que se asigna el argumento. Cuando se asigna un argumento a un parámetro por nombre, todos los argumentos que le siguen también deben asignarse por nombre (SQLSTATE 4274K).

Un argumento con nombre sólo debe especificarse una vez (implícita o explícitamente) (SQLSTATE 4274K).

Los argumentos con nombre no reciben soporte en la llamada a un procedimiento no catalogado (SQLSTATE 4274K).

expresión o **DEFAULT** o **NULL**

Cada especificación de *expresión*, la palabra clave **DEFAULT** o la palabra clave **NULL** es un argumento de **CALL**. El argumento número *n* sin nombre de la sentencia **CALL** corresponde al parámetro número *n* definido en la sentencia **CREATE PROCEDURE** del procedimiento.

Los argumentos con nombre corresponden al mismo parámetro con nombre, independientemente del orden en que se especificaron.

Si se especifica la palabra clave **DEFAULT**, se utilizará el valor por omisión tal como está definido en la sentencia **CREATE PROCEDURE**, si existe; de lo contrario, se utilizará el valor nulo como valor por omisión.

Si se especifica la palabra clave **NULL**, el valor nulo se transfiere como valor del parámetro.

Cada argumento de **CALL** debe ser compatible con el correspondiente parámetro de la definición del procedimiento, como se indica a continuación:

- Parámetro **IN**
 - El argumento debe poder asignarse al parámetro.
 - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de almacenamiento.
- Parámetro **OUT**
 - El argumento debe ser una única variable o marcador de parámetro (SQLSTATE 42886).
 - El argumento debe poder asignarse al parámetro.
 - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de recuperación.
- Parámetro **INOUT**
 - El argumento debe ser una única variable o marcador de parámetro (SQLSTATE 42886).
 - El argumento debe poder asignarse al parámetro.
 - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de almacenamiento y, a su vez, las normas de asignación de recuperación.

Notas

- **Asignaciones de parámetros:** cuando se ejecuta la sentencia **CALL**, el valor de cada uno de sus argumentos se asigna (mediante la asignación de almacenamiento) al parámetro correspondiente del procedimiento. Los valores de los parámetros que se hayan definido con un valor por omisión pueden omitirse de la lista de argumentos cuando se invoque el procedimiento.

Cuando se ejecuta la sentencia **CALL**, se transfiere el control al procedimiento según los convenios de llamada del lenguaje principal. Cuando la ejecución del procedimiento se ha completado, el valor de cada parámetro del procedimiento se asigna (utilizando la asignación de almacenamiento) al argumento correspondiente de la sentencia **CALL** definido como **OUT** o **INOUT**. Si el procedimiento devuelve un error, los argumentos **OUT** no se definen y los argumentos **INOUT** no cambian. Para obtener detalles sobre las normas de asignación, consulte “Asignaciones y comparaciones”.

Cuando la sentencia CALL se encuentra en un procedimiento de SQL y llama a otro procedimiento de SQL, la asignación de parámetros XML se realiza por referencia. Cuando se pasa un argumento XML por referencia, los árboles de nodos de entrada, si existe alguno, se utilizan directamente del argumento XML, manteniendo todas las propiedades, incluyendo el orden de documentos, las identidades de nodo originales y todas las propiedades padre.

- **Signaturas de procedimiento:** un procedimiento se identifica por su esquema, un nombre de procedimiento y el número de parámetros. Esto se denomina *signatura de procedimiento*, que debe ser exclusiva dentro de la base de datos. Puede haber más de un procedimiento con el mismo nombre en un esquema, siempre que el número de parámetros sea distinto para cada procedimiento.
- **Vía de acceso de SQL:** puede invocarse un procedimiento haciendo referencia a un nombre calificado (nombre de esquema y de procedimiento) seguido por una lista opcional de argumentos, entre paréntesis. Un procedimiento también puede invocarse sin el nombre de esquema, lo que da como resultado una elección de posibles procedimientos de distintos esquemas que tienen el mismo número de parámetros. En este caso, la vía de acceso de SQL se utiliza como ayuda para la resolución del procedimiento. La vía de acceso de SQL es una lista de esquemas en la que se realizan búsquedas para identificar un procedimiento que tiene el mismo nombre y número de parámetros. Para las sentencias CALL estáticas, la vía de acceso de SQL se especifica utilizando la opción de vinculación FUNCPATH. Para las sentencias CALL dinámicas, la vía de acceso de SQL es el valor del registro especial CURRENT PATH.
- **Resolución del procedimiento:** después de la invocación de un procedimiento, el gestor de bases de datos debe decidir cuál de los posibles procedimientos con el mismo nombre es el que debe ejecutar.

La resolución de procedimiento de ámbito local se usa cuando se invoca un procedimiento desde una sentencia de SQL compuesto (compilado) y se da uno de los criterios siguientes:

- Un procedimiento con el mismo nombre que el invocado se declara en la misma sentencia de SQL compuesto (compilado)
- Un procedimiento con el mismo nombre que el invocado se declara en una sentencia de SQL compuesto (compilado), dentro de la cual está anidada la sentencia de SQL compuesto (compilado) que invocó el procedimiento

La resolución de procedimientos de ámbito local significa que sólo se consideran los procedimientos declarados dentro del ámbito de la sentencia de SQL compuesto (compilado) que ha invocado el procedimiento durante la resolución de procedimientos, independientemente de la posible existencia de procedimientos incorporados coincidentes, procedimientos de esquema o procedimientos de módulo. La *resolución de procedimientos de ámbito global* se utiliza en todos los demás casos y considera los candidatos de los esquemas y módulos según el contexto de la invocación y la calificación del nombre de procedimiento.

- Dejar que *A* sea el número de argumentos en la invocación de un procedimiento.
- Dejar que *P* sea el número de parámetros en la signatura de un procedimiento.
- Dejar que *N* sea el número de parámetros sin valor por omisión.

Los procedimientos candidatos para la resolución de una invocación de procedimiento se seleccionan en función de los criterios siguientes:

- Cada procedimiento candidato tiene un nombre coincidente y un número de parámetros aplicable. Un número aplicable de parámetros cumple la condición $N = A = P$.

- Cada procedimiento candidato tiene los parámetros necesarios para que para cada argumento con nombre incluido en la sentencia CALL exista un parámetro con un nombre coincidente que no corresponda ya a un argumento de posición (o sin nombre).
- Cada parámetro de un procedimiento candidato que no tenga un argumento correspondiente en la sentencia CALL, especificado por posición o por nombre, se define con un valor por omisión.
- Cada procedimiento candidato de un conjunto de uno o varios esquemas tiene el privilegio EXECUTE asociado con el ID de autorización de la sentencia que invoca la función.

Asimismo, el conjunto de procedimientos candidatos depende del entorno en el que se invoca el procedimiento y de la forma de calificarse el nombre de procedimiento.

- Si el nombre del procedimiento no está calificado, la resolución del procedimiento se realiza siguiendo estos pasos:
 1. Si se invoca un procedimiento desde una sentencia de SQL compuesto (compilado) y existe un procedimiento declarado con el mismo nombre en el ámbito anidado, busque el conjunto de sentencias de SQL compuesto (compilado) en el que la sentencia CALL está anidada para los procedimientos candidatos. Si no se encuentra ningún procedimiento candidato, se devuelve un error (SQLSTATE 42884). Si se encuentra un procedimiento candidato, la resolución ha finalizado. Si hay varios procedimientos candidatos, determine el que tenga el menor número de parámetros y elimine los que tengan un mayor número de parámetros.
 2. Si se invoca un procedimiento desde un objeto de módulo, busque procedimientos candidatos dentro del módulo. Si se encuentran uno o más procedimientos candidatos en el módulo de contexto, estos procedimientos candidatos se incluyen con cualquier procedimiento candidato de los esquemas en la vía de acceso de SQL (véase el siguiente elemento).
 3. Buscar todos los procedimientos de esquema con un esquema en la vía de acceso de SQL para los procedimientos candidatos. Si se encuentran uno o más procedimientos candidatos en los esquemas de la vía de acceso de SQL, estos procedimientos candidatos se incluyen con cualquier procedimiento candidato del módulo de contexto (véase el elemento anterior). Si permanece un solo procedimiento candidato, la resolución ha finalizado. Si hay varios procedimientos candidatos, elija el procedimiento del módulo de contexto si continúa siendo candidato; de lo contrario, elija el procedimiento cuyo esquema aparezca antes en la vía de acceso de SQL. Si todavía hay varios procedimientos candidatos, determine el que tenga el menor número de parámetros y elimine los que tengan un mayor número de parámetros.

Si no quedan procedimientos candidatos después del paso 3, se devuelve un error (SQLSTATE 42884).

- Si el nombre del procedimiento está calificado, la resolución del procedimiento se realiza siguiendo estos pasos:
 1. Si se invoca el procedimiento desde una sentencia de SQL compuesto (compilado) y existe un procedimiento declarado con el mismo nombre donde el calificador coincide con la etiqueta de la sentencia de SQL compuesto (compilado) del conjunto de sentencias de SQL compuesto (compilado) en el que la sentencia CALL está anidada, busque esa sentencia de SQL compuesto (compilado) con la etiqueta coincidente para los procedimientos candidatos. Si no se encuentra ningún procedimiento candidato, se devuelve un error (SQLSTATE 42884). Si se encuentra un

procedimiento candidato, la resolución ha finalizado. Si hay varios procedimientos candidatos, determine el que tenga el menor número de parámetros y elimine los que tengan un mayor número de parámetros.

2. Si el procedimiento se invoca desde un módulo y el calificador coincide con el nombre del módulo desde el cual se invoca el procedimiento, busque procedimientos candidatos en el módulo. Si el calificador es un solo identificador, el nombre de esquema del módulo se pasa por alto cuando coincide con el nombre del módulo. Si el calificador es un identificador de dos partes, se compara con el nombre de módulo calificado por esquema al determinar una coincidencia. Si hay un solo procedimiento candidato que coincida, la resolución ha finalizado. Si hay varios procedimientos candidatos, elija el que tenga el menor número de parámetros. Si el calificador no coincide o no hay ningún procedimiento candidato, continúe con el paso siguiente.
3. Considerar el calificador como nombre de esquema y buscar procedimientos candidatos en ese esquema. Si hay un solo procedimiento candidato que coincida, la resolución ha finalizado. Si hay varios procedimientos candidatos, elija el que tenga el menor número de parámetros y la resolución habrá finalizado. Si el esquema no existe o no hay procedimientos candidatos autorizados, y el calificador ha coincidido con el nombre del módulo del primer paso, se devolverá un error. De lo contrario, continúe con el paso siguiente.
4. Considerar el calificador como nombre de módulo, sin considerar el privilegio EXECUTE para los módulos:
 - Si el nombre de módulo está calificado con un nombre de esquema, busque los procedimientos publicados de este módulo para encontrar procedimientos candidatos.
 - Si el nombre del módulo no está calificado con un nombre de esquema, el esquema del módulo es el primer esquema de la vía de acceso SQL que tiene un nombre de módulo coincidente. Si se encuentra, busque los procedimientos publicados en este módulo para encontrar procedimientos candidatos.
 - Si el módulo no se encuentra utilizando la vía de acceso de SQL, consulte un alias público de módulo que coincida con el nombre del calificador del procedimiento. Si se encuentra, busque los procedimientos publicados en este módulo para encontrar procedimientos candidatos.

Si no se encuentra un módulo coincidente o no hay procedimientos candidatos en el módulo coincidente, se devuelve un error de procedimiento no encontrado (SQLSTATE 42884). Si hay varios procedimientos candidatos, elija el que tenga el menor número de parámetros. La resolución ha finalizado si el ID de autorización de la sentencia CALL tiene el privilegio EXECUTE para el módulo del procedimiento candidato restante; de lo contrario, se devuelve un error de autorización (SQLSTATE 42501).

- **Recuperación de DB2_RETURN_STATUS de un procedimiento SQL:** si un procedimiento SQL emite satisfactoriamente una sentencia RETURN con un valor de estado, este valor se devuelve en el primer campo SQLERRD de SQLCA. Si la sentencia CALL se emite en un procedimiento de SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de DB2_RETURN_STATUS. El valor es -1 cuando SQLSTATE indica un error. El valor es 0 si no se ha devuelto ningún error y si la sentencia RETURN no se ha especificado en el procedimiento.

- **Devolución de conjuntos de resultados de procedimientos:** si el programa de llamada se ha grabado con CLI, JDBC o SQLJ o el que realiza la llamada es un procedimiento SQL, los conjuntos de resultados pueden devolverse directamente al proceso que realiza la llamada. El procedimiento indica que va a devolverse un conjunto de resultados declarando un cursor en ese conjunto de resultados, abriendo un cursor en el conjunto de resultados y dejando el cursor abierto al salir del procedimiento.

Al final de un procedimiento:

- Para cada cursor que se haya dejado abierto, se devuelve un conjunto de resultados al llamante o (para cursores WITH RETURN TO CLIENT) directamente al cliente.
- Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto de resultados de un cursor tiene 500 filas, y el procedimiento ha leído 150 de dichas filas una vez que finaliza el procedimiento, las filas de la 151 a la 500 se devolverán al llamante o a la aplicación (según proceda).

Si el procedimiento se ha invocado desde la CLI o desde JDBC y se ha dejado abierto más de un cursor, los conjuntos de resultados sólo pueden procesarse en el orden en que se han abierto los cursores.

- **Mejora del rendimiento:** los valores de todos los argumentos se pasan de la aplicación al procedimiento. Para mejorar el rendimiento de esta operación, las variables del lenguaje principal que corresponden a los parámetros OUT y que tienen longitudes de más de simplemente algunos bytes deben establecerse en el valor null antes de ejecutarse la sentencia CALL.
- **Anidamiento de las sentencias CALL:** los procedimientos pueden llamarse desde rutinas y desde programas de aplicación. Cuando se llama a un procedimiento desde una rutina, se considera que la llamada está anidada.

Si un procedimiento devuelve algún conjunto de resultados de la consulta, los conjuntos de resultados se devuelven como se indica a continuación:

- Los conjuntos de resultados RETURN TO CALLER sólo puede verlos el programa que se encuentra en el nivel anterior de anidación.
- Los conjuntos de resultados de RETURN TO CLIENT sólo están visibles si el procedimiento se ha invocado desde un conjunto de procedimientos anidados. Si la función o el método tienen lugar en cualquier punto de la cadena de la llamada, el conjunto de resultados no está visible. Si el conjunto de resultados es visible, sólo lo es para la aplicación cliente que ha realizado la llamada de procedimiento inicial.

Consideremos el ejemplo siguiente:

```
Programa cliente:
EXEC SQL CALL PROCA;
```

```
PROCA:
EXEC SQL CALL PROCB;
```

```
PROCB:
EXEC SQL DECLARE B1 CURSOR WITH RETURN TO CLIENT ...;
EXEC SQL DECLARE B2 CURSOR WITH RETURN TO CALLER ...;
EXEC SQL DECLARE B3 CURSOR FOR SELECT UDFA FROM T1;
```

```
UDFA:
EXEC SQL CALL PROCC;
```

```
PROCC:
EXEC SQL DECLARE C1 CURSOR WITH RETURN TO CLIENT ...;
EXEC SQL DECLARE C2 CURSOR WITH RETURN TO CALLER ...;
```

En el procedimiento PROCB:

CALL

- El cursor B1 está visible en la aplicación cliente, pero no está visible en el procedimiento PROCA.
- El cursor B2 está visible en PROCA, pero no está visible para el cliente.

En el procedimiento PROCC:

- El cursor C1 no está visible para UDFA ni para la aplicación cliente. (Puesto que UDFA aparece en la cadena de llamada entre el cliente y PROCC, el conjunto de resultados no se devuelve al cliente.)
- El cursor C2 está visible en UDFA, pero no está visible para ninguno de los procedimientos superiores.
- **Anidamiento de procedimientos en activadores, sentencias compuestas, funciones o métodos:** cuando se llama a un procedimiento en un activador, en una sentencia compuesta, en una función o en un método:
 - El procedimiento no debe emitir una sentencia COMMIT ni ROLLBACK.
 - No se puede acceder a los conjuntos de resultados que devuelve el procedimiento.
 - Si el procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del mismo podrá acceder a la tabla que la sentencia que ha invocado el procedimiento está modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del mismo podrá modificar la tabla que la sentencia que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

Cuando se llama a un procedimiento desde una función o un método:

- El procedimiento tiene las mismas restricciones de acceso a las tablas que la función o el método que realiza la invocación.
- Los puntos de salvaguarda definidos antes de invocar la función o el método no estarán visibles para el procedimiento, y los puntos de salvaguarda definidos dentro del procedimiento no estarán visibles fuera de la función o del método.
- Desde el cliente no es posible acceder a los conjuntos de resultados RETURN TO CLIENT que devuelve el procedimiento.
- **Compilación de sentencias CALL de DB2 para i y DB2 para z/OS:** La compilación de sentencias CALL de DB2 para i y DB2 para z/OS se comporta implícitamente como si CALL_RESOLUTION DEFERRED se hubiera especificado. Cuando las sentencias CALL se compilan con CALL_RESOLUTION DEFERRED, todos los argumentos deben proporcionarse mediante variables de lenguaje principal, y no se permiten las expresiones.
- **Alternativas de sintaxis:** existe un formato anterior de la sentencia CALL que puede incluirse en una aplicación mediante la precompilación de esta con la opción CALL_RESOLUTION DEFERRED. Esta opción no está disponible para procedimientos SQL ni para procedimientos federados.

Ejemplos

- *Ejemplo 1:* Un procedimiento 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 utilizando el fragmento de código siguiente:

```

...
CallableStatement stpCall ;

String sql = "CALL PARTS_ON_HAND (?, ?, ?)";

stpCall = con.prepareStatement( sql ) ; /* con es la conexión */

stpCall.setInt(1, hvPartnum);
stpCall.setBigDecimal(2, hvCost);
stpCall.setInt(3, hvQuantity);

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

hvCost = stpCall.getBigDecimal(2);
hvQuantity = stpCall.getInt(3);
...

```

Este fragmento de código de aplicación invocará el método Java onhand de la clase parts, porque el nombre de procedimiento especificado en la sentencia CALL se encuentra en la base de datos y tiene el nombre externo parts!onhand.

- *Ejemplo 2:* Existen seis procedimientos FOO, en cuatro esquemas distintos, registrados de la forma siguiente (observe que no aparecen todas las palabras clave necesarias):

```

CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...

```

La referencia de procedimiento es la siguiente (donde I1 e I2 son valores INTEGER):

```
CALL FOO(I1, I2)
```

Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
"JULIUS", "AUGUSTUS", "CAESAR"
```

De acuerdo con el algoritmo...

El procedimiento que tiene el nombre específico FOO_6 se elimina como candidato, ya que el esquema "NERO" no se ha incluido en la vía de acceso de SQL. FOO_1, FOO_3 y FOO_4 se eliminan como candidatos, pues tienen un número incorrecto de parámetros. Se considera que los restantes candidatos son correctos, tal como determina la vía de acceso de SQL. Observe que los tipos de los argumentos y parámetros se pasan por alto. Los parámetros de FOO_5 coinciden exactamente con los argumentos de CALL, pero se ha elegido FOO_2 porque "AUGUSTUS" aparece antes que "CAESAR" en la vía de acceso de SQL.

- *Ejemplo 3:* Supongamos que existe el procedimiento siguiente.

```

z CREATE PROCEDURE update_order(
    IN IN_POID BIGINT,
    IN IN_CUSTID BIGINT DEFAULT GLOBAL_CUST_ID,
    IN NEW_STATUS VARCHAR(10) DEFAULT NULL,
    IN NEW_ORDERDATE DATE DEFAULT NULL,
    IN NEW_COMMENTS VARCHAR(1000)DEFAULT NULL)...

```

Supongamos también que la variable global GLOBAL_CUST_ID está establecida en el valor 1002. Llame al procedimiento para cambiar el estado del pedido 5000 del cliente 1002 a 'Shipped' (enviado). Deje el resto de los datos del pedido tal como están, dejando que el resto de los argumentos tomen como valor por omisión el valor nulo.

CALL

```
CALL update_order (5000, NEW_STATUS => 'Shipped')
```

El cliente con el ID 1001 ha llamado y ha indicado que han recibido su envío correspondiente a la orden de compra 5002 y que está conforme. Actualice su pedido.

```
CALL update_order (5002,  
  IN_CUSTID => 1001,  
  NEW_STATUS => 'Received',  
  NEW_COMMENTS => 'Customer satisfied with the order.')
```

- *Ejemplo 4:* en el siguiente ejemplo se muestra la resolución del procedimiento, con dos procedimientos denominados *p1*:

```
CREATE PROCEDURE p1(i1 INT) ...  
CREATE PROCEDURE p1(i1 INT DEFAULT 0, i2 INT DEFAULT 0) ...  
CALL p1(i2=>1)
```

A partir de DB2 Versión 9.7 Fixpack 1, durante el proceso de selección de candidatos se tienen en consideración los nombres de los argumentos. Por lo tanto, sólo se considerará como candidato la segunda versión de *p1*. Asimismo, podrá llamarse a este correctamente porque *i1*, en esta versión de *p1*, se ha definido con un valor por omisión; por lo tanto, es válido especificar sólo *i2* en la llamada a *p1*.

- *Ejemplo 5:* en el siguiente ejemplo se muestra otra resolución del procedimiento, con dos procedimientos denominados *p1*:

```
CREATE PROCEDURE p1(i1 INT, i2 INT DEFAULT 0) ...  
CREATE PROCEDURE p1(i1 INT DEFAULT 0, i2 INT DEFAULT 0, i3 INT DEFAULT 0) ...  
CALL p1(i2=>1)
```

A partir de DB2 Versión 9.7 Fixpack 1, uno de los criterios para un parámetro de procedimiento que no tiene un argumento correspondiente en la sentencia CALL (especificado por posición o por nombre) consiste en que el parámetro se defina con un valor por omisión. Por lo tanto, no se considera como candidata la primera versión de *p1*.

CASE

La sentencia CASE selecciona una vía de ejecución de acuerdo con varias condiciones. Esta sentencia no se debe confundir con la expresión CASE, que permite seleccionar una expresión basándose en la evaluación de una o varias condiciones.

Invocación

Esta sentencia se puede incluir en:

- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias de SQL compuesto pueden incorporarse en una definición de procedimiento, función o activador de SQL. La sentencia CASE no es una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización

No se requieren privilegios para invocar la sentencia CASE. Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL y expresiones incorporadas en la sentencia CASE.

Sintaxis

► CASE ┌ cláusula-when-sentencia-case-búsqueda ┌ END CASE ───────────────────►
 └ cláusula-when-sentencia-case-simple └

cláusula-when-sentencia-case-simple:

┌ expresión ─ WHEN ─ expresión ─ THEN ─ sentencia-procedimiento-SQL ─ ; ─►
 └ ───┘
 ┌ ───┘
 └ ELSE ─ sentencia-procedimiento-SQL ─ ; ─►

cláusula-when-sentencia-case-búsqueda:

┌ ───┘
 └ WHEN ─ condición-búsqueda ─ THEN ─ sentencia-procedimiento-SQL ─ ; ─►



Descripción

CASE

Empieza una *sentencia-case*.

cláusula-when-sentencia-case-simple

El valor de la *expresión* anterior a la primera palabra clave WHEN se comprueba para determinar 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. Consulte *sentencia-procedimiento-SQL* en la sentencia de “SQL compuesto (compilado)”.

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

En función del valor de la variable de SQL `v_workdept`, actualice la columna DEPTNAME de la tabla DEPARTMENT con el nombre adecuado.

- *Ejemplo 1*: 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
```

- *Ejemplo 2:* 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  
        SET deptname = 'DATA ACCESS 2';  
  ELSE UPDATE department  
        SET deptname = 'DATA ACCESS 3';  
END CASE
```

CLOSE

La sentencia CLOSE cierra un cursor. Si se ha creado una tabla de resultados cuando se ha abierto el cursor, se ha destruido esa tabla.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que no se puede preparar dinámicamente. Cuando se invoca utilizando el procesador de línea de mandatos, no se pueden especificar algunas opciones. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

Autorización

Si se hace referencia a una variable global, los privilegios del ID de autorización de la sentencia deben incluir unas de las autorizaciones siguientes:

- el privilegio READ sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE en el módulo de la variable global que está definida en un módulo

Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

Sintaxis

```

→ CLOSE nombre-cursor | nombre-variable-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.

nombre-variable-cursor

Identifica el cursor que se va a cerrar. *nombre-variable-cursor* debe identificar una variable de cursor. Cuando se ejecuta la sentencia CLOSE, el cursor subyacente del *nombre-variable-cursor* debe estar en estado abierto (SQLSTATE 24501). Una sentencia CLOSE que utiliza un *nombre-variable-cursor* sólo se puede utilizar en una sentencia de SQL compuesto (compilado).

WITH RELEASE

Se intenta liberar todos los bloqueos que se han mantenido para el cursor. Tenga en cuenta que no se liberan necesariamente todos los bloqueos; se pueden conservar bloqueos 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.
- Un cursor subyacente de una variable de cursor está implícitamente cerrado cuando se convierte en un cursor huérfano. Un cursor subyacente se convierte en huérfano cuando deja de ser un cursor subyacente de cualquier variable de

cursor. Por ejemplo, esto puede ocurrir si todas las variables de cursor de un cursor subyacente están en el mismo ámbito y todas salen del ámbito al mismo tiempo.

- La cláusula WITH RELEASE no tiene ningún efecto al cerrar los cursores que se han definido en las funciones o en los métodos. La cláusula tampoco tiene ningún efecto al cerrar los cursores que se han definido en los procedimientos llamados desde las funciones o los métodos.
- 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'.

Si un cursor que originalmente era RR o RS se vuelve a abrir tras haberse cerrado utilizando la cláusula WITH RELEASE, adquirirá nuevos bloqueos.

- Se aplican normas especiales a los cursores de un procedimiento que no se hayan cerrado antes de volver al programa que efectúa la llamada.
- Mientras un cursor está abierto (es decir, todavía no se ha cerrado), los cambios realizados en los valores de secuencia como resultado de las sentencias que implican a ese cursor (por ejemplo FETCH o UPDATE utilizando el cursor que incluye una expresión NEXT VALUE para una secuencia) no darán como resultado una actualización en PREVIOUS VALUE para esas secuencias desde el punto de vista del cursor. Los valores PREVIOUS VALUE de esas secuencias afectadas se actualizarán cuando el cursor se cierre explícitamente con la sentencia CLOSE. En un entorno de base de datos particionada, si un cursor se cierra implícitamente mediante una operación de confirmar o retrotraer, puede que no se actualice PREVIOUS VALUE con el valor generado más recientemente para la secuencia.

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 debe 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

La sentencia COMMENT añade o sustituye comentarios en las descripciones del catálogo de diversos objetos.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

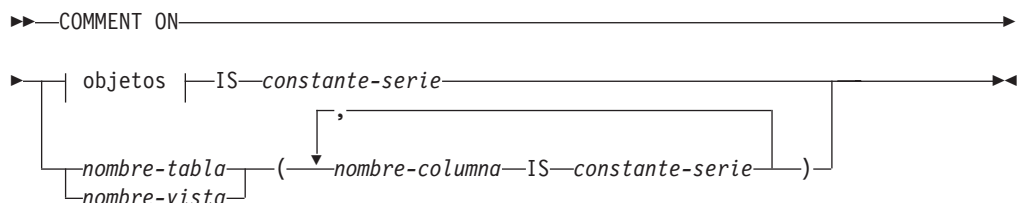
Autorización

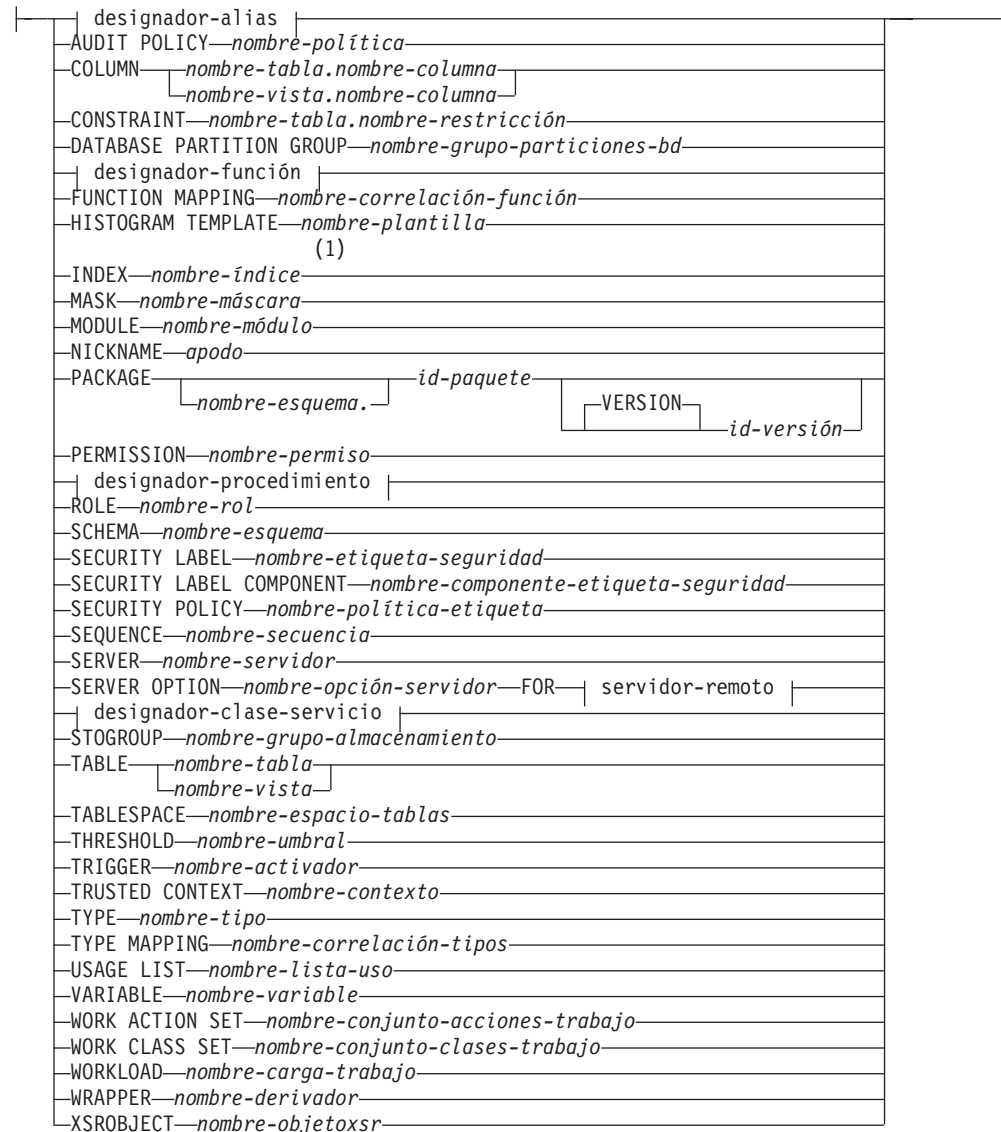
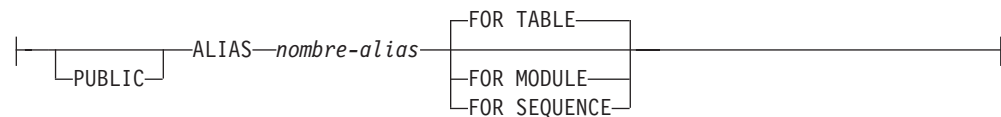
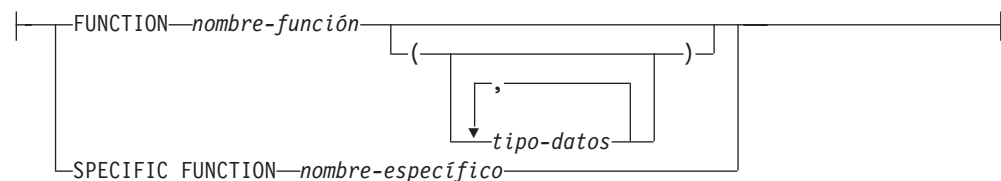
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Propietario del objeto (tabla subyacente para la columna o restricción), tal como está registrado en la columna OWNER de la vista de catálogo del objeto
- 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 o vista)
- Privilegio ALTER para el objeto (sólo aplicable a los objetos de tabla)
- Autorización CREATE_SECURE_OBJECT (sólo aplicable a funciones seguras o activadores seguros)
- WITH ADMIN OPTION (aplicable sólo a roles)
- Autorización WLMADM (aplicable sólo a los objetos del gestor de carga de trabajo)
- Autorización SECADM (sólo aplicable a política de auditoría, máscara de columna, rol, permiso de fila, función segura, activador seguro, etiqueta de seguridad, componente de etiqueta de seguridad, política de seguridad u objetos de contexto fiable; también aplicable a tablas para las que se ha activado el control de acceso de nivel de fila o el control de acceso de nivel de columna)
- Autorización DBADM (aplicable a todos los objetos, salvo a política de auditoría, rol, etiqueta de seguridad, componente de etiqueta de seguridad, política de seguridad u objetos de contexto fiable)

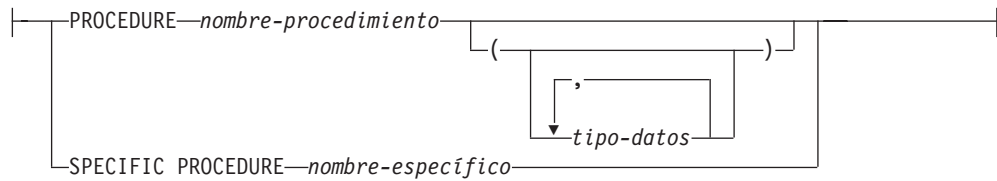
Tenga en cuenta que para el espacio de tablas, el grupo de almacenamiento o el grupo de particiones de base de datos y las agrupaciones de almacenamientos intermedios, el ID de autorización debe tener la autorización SYSCTRL o SYSADM.

Sintaxis

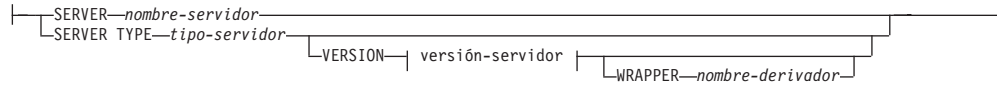


objetos:**designador-alias:****designador-función:**

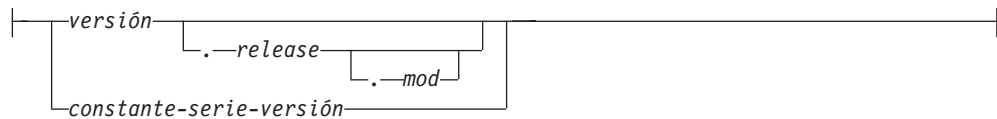
designador-procedimiento:



servidor-remoto:



versión-servidor:



designador-clase-servicio:



Notas:

- 1 Nombre-índice puede ser el nombre de un índice o una especificación de índice.

Descripción

designador-alias

ALIAS nombre-alias

Indica un comentario que se añadirá o sustituirá para un alias. El nombre-alias debe identificar un alias que exista en el servidor actual (SQLSTATE 42704).

Para TABLE, FOR MODULE o FOR SEQUENCE

Especifica el tipo de objeto del alias.

FOR TABLE

El alias es para una tabla, vista o apodo. El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES para la fila que describe el alias.

FOR MODULE

El alias es para un módulo. El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.MODULES para la fila que describe el alias.

FOR SEQUENCE

El alias es para una secuencia. El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SEQUENCES para la fila que describe el alias.

Si se especifica PUBLIC, el *nombre-alias* debe identificar un alias público que existe en el servidor actual (SQLSTATE 42704).

AUDIT POLICY *nombre-política*

Indica un comentario que se añadirá o sustituirá para una política de auditoría. El *nombre-política* debe identificar una política de auditoría que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.AUDITPOLICIES para la fila que describe la política de auditoría.

COLUMN *nombre-tabla.nombre-columna* o *nombre-vista.nombre-columna*

Indica que se añadirá o se sustituirá un comentario para una columna. La combinación *nombre-tabla.nombre-columna* o *nombre-vista.nombre-columna* debe identificar una combinación de tabla y columna que exista en el servidor actual (SQLSTATE 42704), pero no debe identificar una tabla temporal global (SQLSTATE 42995). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.COLUMNS para la fila que describe la columna.

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 existir en el servidor actual (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.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*

Indica que se añadirá o se sustituirá un comentario para un grupo de particiones de base de datos. El *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos diferenciado que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.DBPARTITIONGROUPS de la fila que describe al grupo de particiones de base de datos.

designador-función

Indica que se añadirá o se sustituirá un comentario para una función. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

No es posible comentar una función que está en el esquema SYSIBM, SYSIBMADM, SYSFUN o SYSPROC (SQLSTATE 42832).

El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.ROUTINES de 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 exista en el servidor actual (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.

HISTOGRAM TEMPLATE *nombre-plantilla*

Indica un comentario que se añadirá o sustituirá para una plantilla de histograma. El *nombre-plantilla* debe identificar una plantilla de histograma que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.HISTOGRAMTEMPLATES para la fila que describe la plantilla de histograma.

COMMENT

INDEX *nombre-índice*

Indica que se añadirá o sustituirá un comentario para un índice o especificación de índice. El *nombre-índice* debe identificar un índice diferenciado o una especificación de índice que exista en el servidor actual (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.

MASK *nombre-máscara*

Identifica la máscara de columna a la que se aplica el comentario. *nombre-máscara* debe identificar una máscara de columna que exista en el servidor actual (SQLSTATE 42704). El comentario se encuentra en la columna REMARKS de la tabla de catálogo SYSCAT.CONTROLS para la fila que describe la máscara.

MODULE *nombre-módulo*

Indica que se añadirá un comentario o se sustituirá para un módulo. El *nombre-módulo* debe identificar un módulo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.MODULES para la fila que describe el módulo.

NICKNAME *apodo*

Indica que se añadirá o sustituirá un comentario para un apodo. El *apodo* debe ser un apodo que exista en el servidor actual (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.

PACKAGE *nombre-esquema.id-paquete*

Indica que se añadirá o se sustituirá un comentario para un paquete. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. El nombre de esquema y el ID de paquete, junto con el ID de versión especificado de forma implícita o explícita, deben identificar un paquete que exista en el servidor actual (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.

VERSION *id-versión*

Identifica en qué versión del paquete va a realizarse el comentario. Si no se especifica un valor, la versión tomará por omisión una serie de caracteres vacía. Si existen varios paquetes con el mismo nombre de paquete pero con distintas versiones, sólo podrá comentarse una versión del paquete en una invocación de la sentencia COMMENT. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

Si la sentencia se invoca desde el indicador de mandatos del sistema operativo, preceda cada delimitador de dobles comillas con una barra inclinada invertida para asegurar que el sistema operativo no divide los delimitadores.

PERMISSION *nombre-permiso*

Identifica el permiso de fila al que se aplica el comentario. *nombre-permiso* debe identificar un permiso de fila que existe en el servidor actual (SQLSTATE 42704, SQLCODE -204). El comentario se encuentra en la columna REMARKS de la tabla de catálogo SYSCAT.CONTROLS para la fila que describe el permiso.

designador-procedimiento

Indica que se añadirá o sustituirá un comentario para un procedimiento. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

No es posible comentar un procedimiento que está en el esquema SYSIBM, SYSIBMADM, SYSFUN o SYSPROC (SQLSTATE 42832).

El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.ROUTINES de la fila que describe al procedimiento.

ROLE nombre-rol

Indica que se añadirá o sustituirá un comentario para el rol. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.ROLES para la fila que describe el rol.

SCHEMA nombre-esquema

Indica que se añadirá o sustituirá un comentario para un esquema. El *nombre-esquema* debe identificar un esquema que exista en el servidor actual (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.

SECURITY LABEL nombre-etiqueta-seguridad

Indica que se añadirá o sustituirá un comentario para la etiqueta de seguridad denominada *nombre-etiqueta-seguridad*. Se debe calificar el nombre con una política de seguridad y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYLABELS para la fila que describe la etiqueta de seguridad.

SECURITY LABEL COMPONENT nombre-componente-etiqueta

Indica que se añadirá o sustituirá un comentario para el componente de etiqueta de seguridad denominado *nombre-componente-etiqueta*. El *nombre-componente-etiqueta* debe identificar un componente de etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYLABELCOMPONENTS para la fila que describe el componente de etiqueta de seguridad.

SECURITY POLICY nombre-política-seguridad

Indica que se añadirá o sustituirá un comentario para la política de seguridad denominada *nombre-política-seguridad*. El *nombre-política-seguridad* debe identificar una política de seguridad que exista en el servidor actual (SQLSTATE 42704). El componente sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYPOLICIES para la fila que describe la política de seguridad.

SEQUENCE nombre-secuencia

Indica que se añadirá o sustituirá un comentario para una secuencia. El *nombre-secuencia* debe identificar una secuencia que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.SEQUENCES para la fila que describe la secuencia.

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 exista en el servidor

actual (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 debe ser una que exista en el servidor actual (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 exista en el servidor actual.

TYPE *tipo-servidor*

Especifica el tipo de fuente de datos (por ejemplo, DB2 para z/OS u Oracle) al que se aplica *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 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 estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-derivador*

Identifica el derivador que se utiliza para acceder a la fuente de datos referenciada por *nombre-servidor*.

designador-clase-servicio

SERVICE CLASS *nombre-clase-servicio*

Indica que se añadirá o sustituirá un comentario para una clase de servicio. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). Para añadir o sustituir un comentario para una subclase de servicio, el *nombre-superclase-servicio* debe especificarse utilizando la cláusula UNDER. El comentario sustituye el valor para la

columna REMARKS de la vista de catálogo SYSCAT.SERVICECLASSES para la fila que describe la clase de servicio.

UNDER *nombre-superclase-servicio*

Especifica la superclase de servicio de la subclase de servicio al añadir o sustituir un comentario para la subclase de servicio. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704).

STOGROUP *nombre-grupo-almacenamiento*

Indica que se añadirá o se sustituirá un comentario para un grupo de almacenamiento. El *nombre-grupo-almacenamiento* debe identificar un grupo de almacenamiento diferencia que existe en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.STOGROUPS de la fila que describe el grupo de almacenamiento.

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 alias o apodo) que exista en el servidor actual (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 exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLESPACES para la fila que describe el espacio de tablas.

THRESHOLD *nombre-umbral*

Indica que se añadirá o suprimirá un comentario para un umbral. El *nombre-umbral* debe identificar un umbral que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.THRESHOLDS para la fila que describe el umbral.

TRIGGER *nombre-activador*

Indica que se añadirá o se sustituirá un comentario para un activador. El *nombre-activador* debe identificar un activador diferenciado que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TRIGGERS para la fila que describe el activador.

TRUSTED CONTEXT *nombre-contexto*

Indica que se añadirá o sustituirá un comentario para un contexto fiable. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.CONTEXTS para la fila que describe el contexto fiable.

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 exista en el servidor actual (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 dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto 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 exista en el servidor actual (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.

USAGE LIST *nombre-lista-uso*

Indica que se añadirá o se sustituirá un comentario para una lista de uso. El *nombre-lista-uso* debe identificar una lista de uso que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.USAGELISTS para la fila que describe la lista de uso.

VARIABLE *nombre-variable*

Indica que se añadirá o sustituirá un comentario para una variable global. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.VARIABLES para la fila que describe la variable.

WORK ACTION SET *nombre-conjunto-acciones-trabajo*

Indica que se añadirá o sustituirá un comentario para un conjunto de acciones de trabajo. El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la fila REMARKS de la vista de catálogo SYSCAT.WORKACTIONSETS para la fila que describe el conjunto de acciones de trabajo.

WORK CLASS SET *nombre-conjunto-clases-trabajo*

Indica que se añadirá o sustituirá un comentario para un conjunto de clases de trabajo. El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WORKCLASSSETS para la fila que describe el conjunto de clases de trabajo.

WORKLOAD *nombre-carga-trabajo*

Indica que se añadirá o sustituirá un comentario para una carga de trabajo. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WORKLOADS para la fila que describe la carga de trabajo.

WRAPPER *nombre-derivador*

Indica que se añadirá o sustituirá un comentario para un derivador. El *nombre-derivador* debe identificar un derivador que exista en el servidor actual (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 derivador.

XROBJECT *nombre-objetoXSR*

Indica un comentario que se añadirá o sustituirá para un objeto XSR. El

nombre-objeto debe identificar un objeto XSR que exista en el servidor actual (SQLSTATE 42704). El componente sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.XSROBJECTS para que describe el objeto XSR.

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 ser calificados, cada nombre debe identificar una columna de la tabla o vista especificada y la tabla o vista debe existir en el servidor actual. 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).

Notas

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
 - DISTINCT TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
 - DATA TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
 - SYNONYM puede especificarse en lugar de ALIAS

Ejemplos

- *Ejemplo 1:* Añada un comentario para la tabla EMPLOYEE.


```
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 la información de 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 debe comentar.


```
COMMENT ON FUNCTION CENTRE (INT,FLOAT)
  IS 'func CENTRE de Frank, utiliza método Chebychev'
```

COMMENT

- *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:* Comente la función ATOMIC_WEIGHT en el esquema CHEM, donde se sabe que sólo hay una función con ese 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 debe comentar.
COMMENT ON PROCEDURE SEARCH (CHAR,INT)
IS 'algoritmo de sustitución y búsqueda masiva de Frank'
- *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**
'Algoritmo de destrucción y búsqueda masiva de Patrick'
- *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 'Cálculos al modelar 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 derivador cuyo nombre por omisión es NET8.
COMMENT ON WRAPPER NET8
IS 'El derivador de las fuentes de datos asociadas al software cliente Net8 de Oracle.'
- *Ejemplo 13:* Cree un comentario sobre el esquema XML HR.EMPLOYEE.
COMMENT ON XSROBJECT HR.EMPLOYEE
IS 'Éste es el esquema XML base para los datos de empleado.'
- *Ejemplo 14:* Cree un comentario para el contexto fiable APPSERVER.
COMMENT ON TRUSTED CONTEXT APPSERVER
IS 'WebSphere Server'
- *Ejemplo 15:* Crear un comentario para una máscara de columna M1.
COMMENT ON MASK M1 **IS** 'Máscara de la columna EMP.SALARY'

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 se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. 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 realizados por las siguientes sentencias ejecutadas durante la unidad de trabajo: ALTER, COMMENT, CREATE, DROP, GRANT, LOCK TABLE, REVOKE, SET INTEGRITY, SET Variable y las sentencias de cambio de datos (INSERT, DELETE, MERGE, UPDATE), incluidas las anidadas en una consulta.

Sin embargo, las sentencias siguientes no están bajo el control de la transacción y los cambios que éstas realizan son independientes de la sentencia COMMIT:

- SET CONNECTION
- SET PASSTHRU

Nota: Aunque la sentencia SET PASSTHRU no está bajo el control de la transacción, la sesión PASSTHRU que ha iniciado la sentencia está bajo el control de la transacción.

- SET SERVER OPTION
- Asignaciones a registros especiales actualizables

Todos los bloqueos que ha adquirido la unidad de trabajo con posterioridad a su inicio se liberan, excepto los bloqueos que son necesarios para abrir los cursores que se han declarado como WITH HOLD. Todos los cursores abiertos no definidos como WITH HOLD se cierran. Los cursores abiertos que se han definido como WITH HOLD siguen abiertos y el cursor se coloca delante de la siguiente fila lógica de la tabla de resultados. (FETCH debe realizarse antes de emitirse una sentencia UPDATE o DELETE con posición.) 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.

Las sentencias de SQL dinámico preparadas en un paquete vinculado con la opción KEEP DYNAMIC YES se conservan en el contexto SQL después de una sentencia COMMIT. Éste es el comportamiento por omisión. La sentencia se puede volver a

COMMIT

preparar de forma implícita, como resultado de operaciones DDL que se retrotraen dentro de la unidad de trabajo. Las sentencias de SQL dinámico inactivas preparadas en un paquete vinculado con KEEP DYNAMIC NO se eliminan del contexto SQL después de una operación COMMIT. La sentencia debe volver a prepararse antes de que se pueda ejecutar en una transacción nueva.

Se liberan todos los puntos de salvaguarda definidos dentro de la transacción.

Las siguientes sentencias tienen un comportamiento diferente al de otras sentencias de lenguaje de definición de datos (DDL) y de lenguaje de control de datos (DCL). Los cambios realizados por estas sentencias no surten efecto hasta que se confirma la sentencia, incluso para la conexión actual que emite la sentencia. Una aplicación sólo puede emitir una de estas sentencias cada vez y sólo se permite una de estas sentencias dentro de una unidad de trabajo. Cada sentencia debe ir seguida de una sentencia COMMIT o ROLLBACK antes de que pueda emitirse otra de estas sentencias.

- CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
- CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
- CREATE WORK ACTION, ALTER WORK ACTION o DROP (WORK ACTION)
- CREATE WORK CLASS, ALTER WORK CLASS o DROP (WORK CLASS)
- CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
- GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)

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.
- Para obtener información acerca del efecto de COMMIT en las sentencias de SQL dinámico colocadas en la antememoria, consulte la sección "EXECUTE".
- Para obtener información acerca de los posibles efectos de COMMIT en las tablas temporales creadas, consulte la sección "CREATE GLOBAL TEMPORARY TABLE".
- Para obtener información acerca de los posibles efectos de COMMIT en las tablas temporales declaradas, consulte la sección "DECLARE GLOBAL TEMPORARY TABLE".
- Las sentencias de SQL dinámico siguientes pueden estar activas durante la operación COMMIT:
 - Cursor WITH HOLD abierto
 - Sentencia COMMIT
 - Sentencias CALL en las que se ha ejecutado la sentencia COMMIT

Ejemplo

Confirme las modificaciones en la base de datos efectuadas desde el último punto de confirmación.

```
COMMIT WORK
```

SQL compuesto

Una sentencia de SQL compuesto es una secuencia de sentencias de SQL individuales especificada entre las palabras clave BEGIN y END.

Existen tres tipos de sentencias de SQL compuesto:

- En línea: Una sentencia de SQL compuesto (en línea) es una sentencia de SQL compuesto que se coloca en línea durante el tiempo de ejecución en otra sentencia de SQL. Las sentencias de SQL compuesto (en línea) tienen la propiedad de ejecutarse automáticamente; si la ejecución de cualquiera de las sentencias origina un error, se retrotrae la sentencia completa.
- Incorporada: Combina una o más sentencias de SQL distintas (*subsentencias*) en un bloque ejecutable.
- Compilada: Una secuencia de sentencias de SQL que se ejecuta con un ámbito local para variables, condiciones, cursores y manejadores.

SQL compuesto (en línea)

Una sentencia de SQL compuesto (en línea) es una sentencia de SQL compuesto que se coloca en línea durante el tiempo de ejecución en otra sentencia de SQL. Las sentencias de SQL compuesto (en línea) tienen la propiedad de ejecutarse automáticamente; si la ejecución de cualquiera de las sentencias origina un error, se retrotrae la sentencia completa.

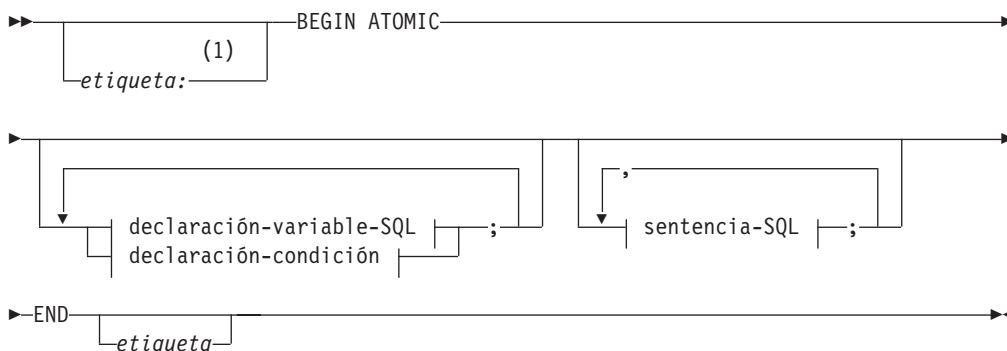
Invocación

Esta sentencia puede incorporarse a un activador, una función de SQL o un método de SQL o bien emitirse mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

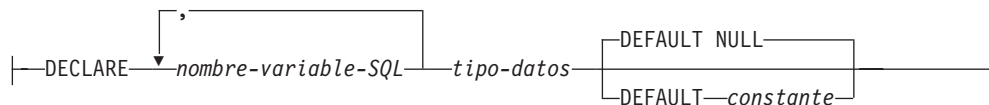
Autorización

Los privilegios que mantiene el ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL especificadas en la sentencia compuesta.

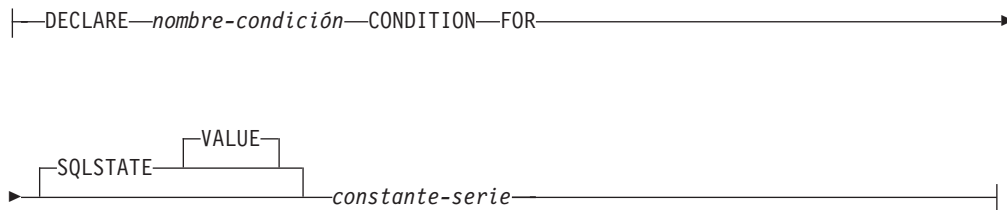
Sintaxis

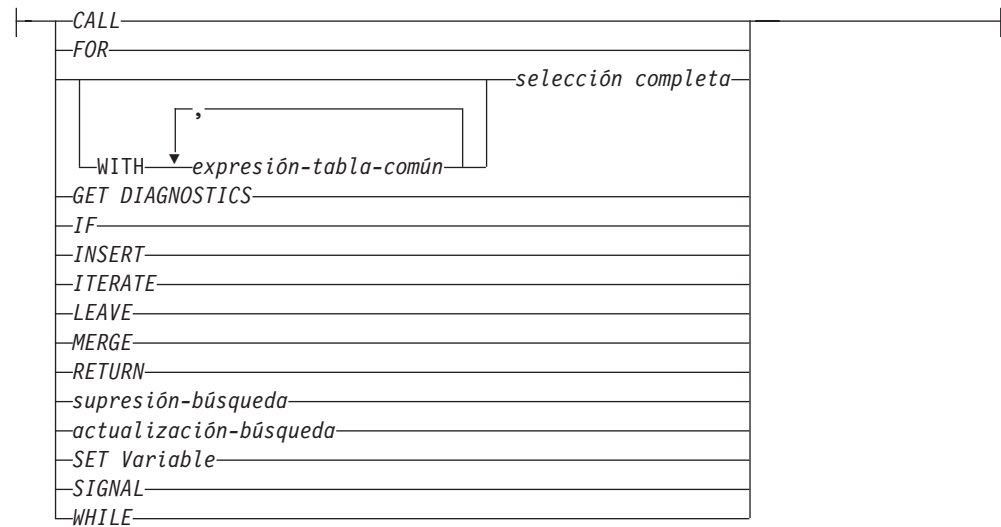


declaración-variable-SQL:



declaración-condición:



sentencia-SQL:**Notas:**

- 1 Sólo se puede especificar una etiqueta cuando la sentencia está en una definición de función, método o activador.

Descripción*etiqueta*

Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, se puede utilizar para calificar variables de SQL declaradas en la sentencia de SQL compuesto (en línea) y también se puede especificar en una sentencia LEAVE. Si se especifica una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC

ATOMIC indica que, si se produce un error en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta y no se procesarán las restantes sentencias de SQL de la sentencia.

Si se especifica la palabra clave ATOMIC en una función de SQL de un módulo o un procedimiento de SQL, la sentencia compuesta se procesa como si se tratara de una sentencia de SQL compuesto (compilado).

sentencia-SQL

Especifica que se ejecute una sentencia de SQL en la sentencia de SQL compuesto (en línea).

declaración-variable-SQL

Declara una variable que es local respecto a la sentencia de SQL compuesto (en línea).

nombre-variable-SQL

Define el nombre de una variable local. Las bases de datos DB2 convierten todos los nombres de variables de SQL a mayúsculas. El nombre no puede ser igual a:

- Otra variable de SQL de la sentencia compuesta
- Un nombre de parámetro

SQL compuesto (en línea)

Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable de SQL y una referencia a columna, el identificador se interpreta como una columna.

tipo-datos

Especifica el tipo de datos de la variable. No se da soporte al tipo de datos XML en sentencias de SQL compuesto (en línea) utilizadas en un activador, en un método o como sentencia independiente (SQLSTATE 429BB). Se da soporte al tipo de datos XML cuando la sentencia de SQL compuesto (en línea) se utiliza en el cuerpo de una función de SQL.

DEFAULT

Define el valor por omisión de la variable de SQL. La variable se inicializa cuando se ejecuta la sentencia de SQL compuesto (en línea). El valor por omisión debe ser una asignación compatible con el tipo de datos de la variable. Si no se especifica un valor por omisión, el valor por omisión para la variable de SQL se inicializa con el valor nulo.

NULL

Especifica NULL como valor por omisión para la variable de SQL.

constante

Especifica una constante como valor por omisión para la variable de SQL.

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 de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una condición dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42737).

FOR SQLSTATE *constante-serie*

Especifica el SQLSTATE asociado con la condición. Debe especificarse la *constante-serie* con cinco caracteres entre comillas simples, y la clase SQLSTATE no debe ser '00'.

Notas

- Las sentencias de SQL compuesto (en línea) se compilan como una única sentencia. Esta sentencia es eficaz en scripts breves que supongan una lógica mínima de flujo de control, pero un flujo importante de datos. En el caso de construcciones más grandes con requisitos de manejo de condiciones o flujo de control anidado, es preferible utilizar la sentencia de SQL compuesto (compilada) o un procedimiento SQL.
- Un procedimiento llamado dentro de una sentencia compuesta no debe emitir una sentencia COMMIT ni ROLLBACK (SQLSTATE 42985).
- **Restricciones de acceso a las tablas:** si un procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del procedimiento puede acceder a una tabla que la sentencia compuesta que ha invocado el procedimiento está modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del procedimiento puede modificar una tabla que la sentencia compuesta que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

- *Asignaciones de XML*: la asignación a parámetros y variables de tipo de datos XML se realiza por referencia en cuerpos de funciones de SQL.
Cuando los valores XML se pasan por referencia, los árboles de nodos de entrada se utilizan directamente. Este uso directo mantiene todas las propiedades, incluyendo el orden de documentos, las identidades de nodo originales y todas las propiedades padre.
- *Nivel de aislamiento*: si una *sentencia-select*, *selección completa* o *subselección* especifica una *cláusula-isolation*, se omite la cláusula y se devuelve un aviso.

Ejemplo

Este ejemplo ilustra cómo se puede utilizar SQL PL en línea 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 las filas que no pueden limpiarse (excepciones) y la tabla SOURCE contiene los datos sin procesar que han de limpiarse.

Para clasificar y modificar los datos se utiliza una función de SQL simple denominada DISCRETIZE. Ésta devuelve el valor nulo para todos los datos incorrectos. La sentencia de SQL compuesto (en línea) limpia entonces los datos. Recorre todas las filas de la tabla SOURCE en un bucle-FOR y determina si la fila actual se inserta en la tabla TARGET o la tabla EXCEPT, en función 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 de SQL compuesto (en línea) 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 de SQL compuesto (en línea) 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 lo tanto, la lógica en su conjunto se ejecuta en el entorno de ejecución de la base de datos DB2. No se mueven datos fuera del motor central de DB2, como sucedería en el caso de un procedimiento.

El primer paso es crear las tablas necesarias:

```
CREATE TABLE TARGET
(PK INTEGER NOT NULL
PRIMARY KEY, C1 INTEGER)
```

Esto crea una tabla denominada TARGET para contener los datos que se han limpiado.

```
CREATE TABLE EXCEPT
(PK INTEGER NOT NULL
PRIMARY KEY, C1 INTEGER)
```

Esto crea una tabla denominada EXCEPT para contener las excepciones.

SQL compuesto (en línea)

```
CREATE TABLE SOURCE
(PK INTEGER NOT NULL
PRIMARY KEY, C1 INTEGER)
```

Esto crea una tabla denominada SOURCE para contener los datos que van a limpiarse.

A continuación, se crea una función denominada DISCRETIZE para limpiar los datos descartando todos los valores que no estén dentro del rango [0..1000] y alineando éstos en pasos de 10.

```
CREATE FUNCTION DISCRETIZE(RAW INTEGER) RETURNS INTEGER
RETURN CASE
WHEN RAW < 0 THEN CAST(NULL AS INTEGER)
WHEN RAW > 1000 THEN NULL
ELSE ((RAW / 10) * 10) + 5
END
```

A continuación, se insertan los valores:

```
INSERT INTO SOURCE (PK, C1)
VALUES (1, -5),
(2, NULL),
(3, 1200),
(4, 23),
(5, 10),
(6, 876)
```

Invoque la función:

```
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
```

Y compruebe los resultados:

```
SELECT * FROM EXCEPT ORDER BY 1
PK      C1
-----
1       -5
2       -
3       1200
3 registro(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)

Combina una o más sentencias de SQL distintas (*subsentencias*) en un bloque ejecutable.

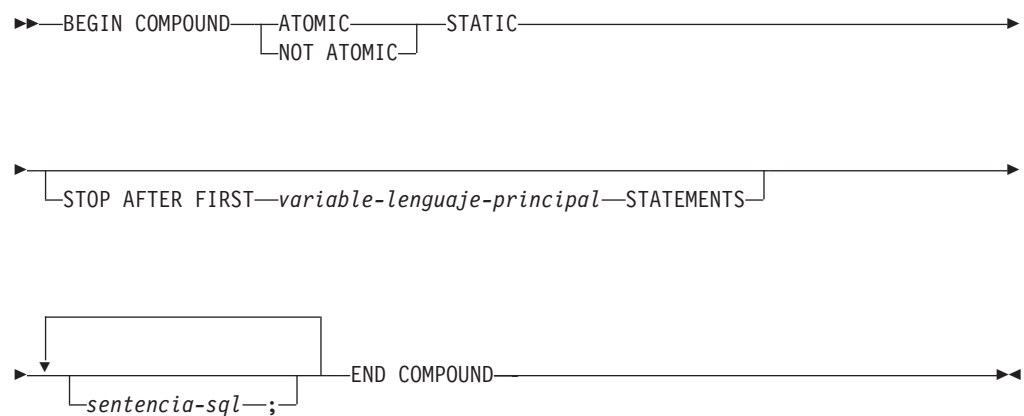
Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Toda la construcción de la sentencia de SQL compuesto (incorporado) es una sentencia ejecutable que no puede prepararse de forma dinámica. La sentencia no se soporta en REXX.

Autorización

No se requieren privilegios para invocar un SQL compuesto (incorporado). Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta.

Sintaxis



Descripción

ATOMIC

Especifica que, si falla alguna de las subsentencias de la sentencia de SQL compuesto (incorporado), se deshacen todos los cambios efectuados en la base de datos por cualquiera de las subsentencias, incluidos los cambios realizados por subsentencias satisfactorias.

NOT ATOMIC

Especifica que, sin tener en cuenta si falla alguna subsentencia, la sentencia de SQL compuesto (incorporado) no desharrá ningún cambio efectuado en la base de datos por las otras subsentencias.

STATIC

Especifica que las variables de entrada para todas las subsentencias conservan su valor original. Por ejemplo, si

```
SELECT ... INTO :abc ...
```

va seguido de:

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

SQL compuesto (incorporado)

la sentencia UPDATE utilizará el valor que :abc tenía al principio de la ejecución de la sentencia de SQL compuesto (incorporado), 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 compuesto (incorporado) 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-lenguaje-principal

Un entero pequeño que especifica el número de subsentencias que se deben ejecutar.

STATEMENTS

Completa la cláusula STOP AFTER FIRST *variable-lenguaje-principal*.

sentencia-sql-

Todas las sentencias ejecutables, excepto las siguientes, pueden estar contenidas en una sentencia de SQL compuesto (incorporado) estática incorporada:

CALL	FETCH
CLOSE	OPEN
CONNECT	PREPARE
SQL compuesto	RELEASE (conexión)
DESCRIBE	ROLLBACK
DISCONNECT	SET CONNECTION
EXECUTE IMMEDIATE	variable SET

Nota: INSERT, UPDATE y DELETE no reciben soporte en el SQL compuesto para su utilización con apodos.

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-lenguaje-principal* 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 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 SQL compuesto.
- No está permitido ningún código de lenguaje principal en una sentencia de SQL compuesto (incorporado); es decir, no está permitido ningún código de lenguaje principal entre las subsentencias que componen la sentencia de SQL compuesto (incorporado).
- DB2 Connect sólo aceptará sentencias de SQL compuesto (incorporado) NOT ATOMIC.
- Las sentencias de SQL compuesto (incorporado) no se pueden anidar.

- No se permite una sentencia COMMIT preparada en una sentencia de SQL compuesto (incorporado) ATOMIC.

Notas

- Se devuelve una SQLCA para toda la sentencia de SQL compuesto (incorporado). 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 ha devuelto el aviso 'no se han encontrado datos' (SQLSTATE 02000), se da prioridad a ese aviso respecto a cualquier otro aviso con el fin de que pueda realizarse una acción en la excepción WHENEVER NOT FOUND. (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 ha activado el aviso 'no se han encontrado datos'. Si existe más de un aviso indicando que no se han encontrado datos dentro de la sentencia de SQL compuesto (incorporado), los campos de la última subsentencia serán los campos que se devuelven).
 - Los indicadores SQLWARN son una acumulación de los indicadores establecidos para todas las subsentencias.
- Si se han producido uno o varios errores durante la ejecución del SQL compuesto NOT ATOMIC y ninguno de éstos es grave, SQLERRMC contendrá información acerca de estos errores, hasta un máximo de siete errores. El primer símbolo de SQLERRMC indicará el número total de errores que se han producido. Los símbolos restantes contendrán cada uno la posición ordinal y el SQLSTATE de la subsentencia anómala dentro de la sentencia de SQL compuesto (incorporado). 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. (Si el número excede de 999, el recuento volverá a empezar desde cero.) Este campo se alinea a 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. (Si el número excede de 999, el recuento volverá a empezar desde cero.) Por ejemplo, si la primera sentencia no se ha ejecutado satisfactoriamente, este campo contendría el número uno alineado a 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 compuesto (incorporado), el cuarto campo SQLERRD se establecería en 2, lo que indicaría que dos subsentencias se habrían procesado de manera satisfactoria antes de encontrar el error.

SQL compuesto (incorporado)

- 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 referencial para todas las subsentencias que han activado dicha actividad de restricción.

Ejemplos

- *Ejemplo 1:* En un programa C, emitir una sentencia de SQL compuesto (incorporado) 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
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;
```

SQL compuesto (compilado)

Una secuencia de sentencias de SQL que se ejecuta con un ámbito local para variables, condiciones, cursores y manejadores.

Invocación

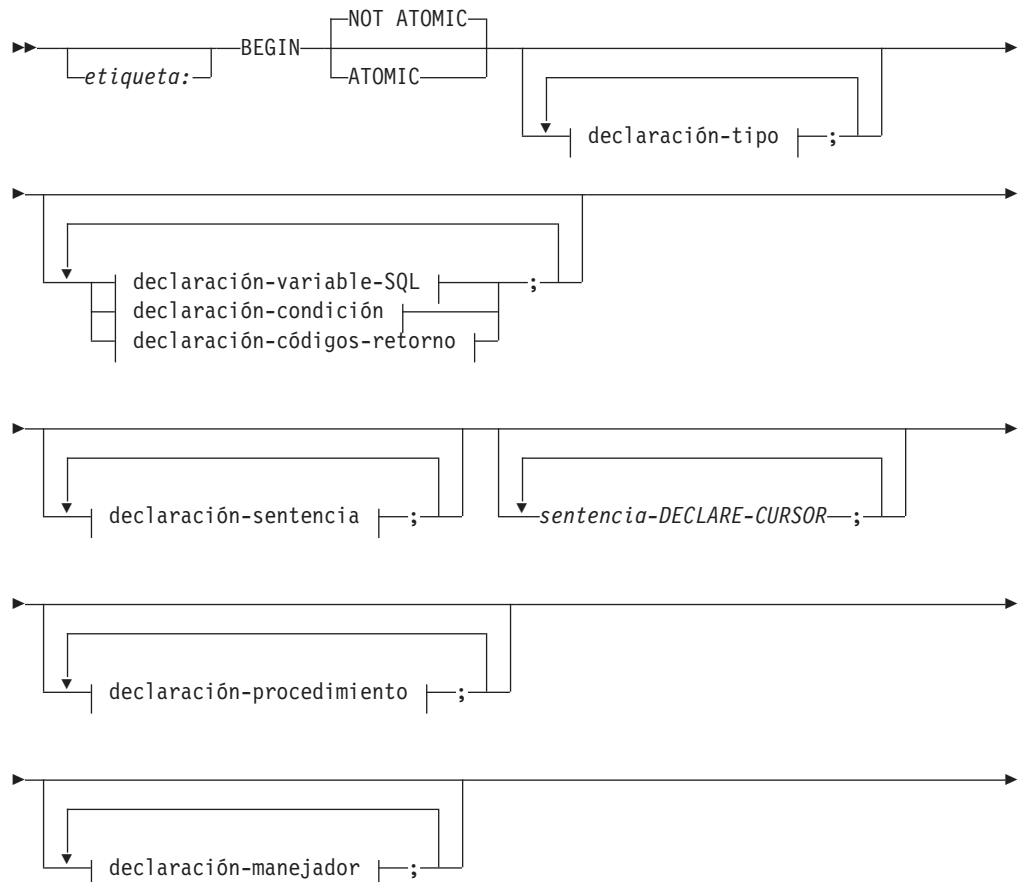
Esta sentencia puede incorporarse a un activador, una función de SQL o un procedimiento de SQL, o puede emitirse mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

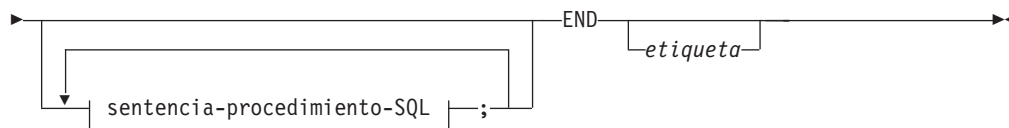
Para una *declaración-variable-SQL* que especifica un *constructor-valor-cursor* que utiliza una *sentencia-select*, los privilegios del ID de autorización de la sentencia deben incluir los privilegios necesarios para ejecutar la *sentencia-select*. Consulte la sección Autorización en "Consultas de SQL".

Los privilegios que mantiene el ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL especificadas en la sentencia compuesta.

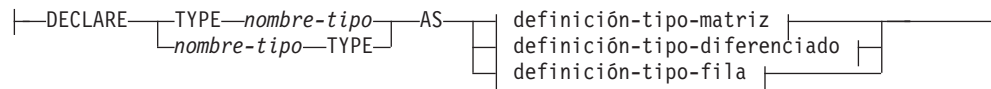
Sintaxis



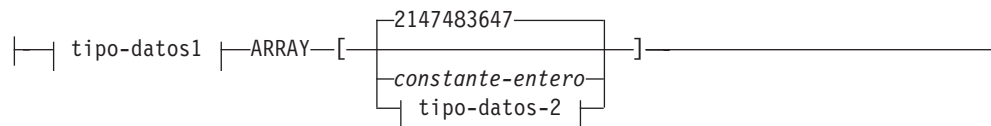
SQL compuesto (compilado)



declaración-tipo:



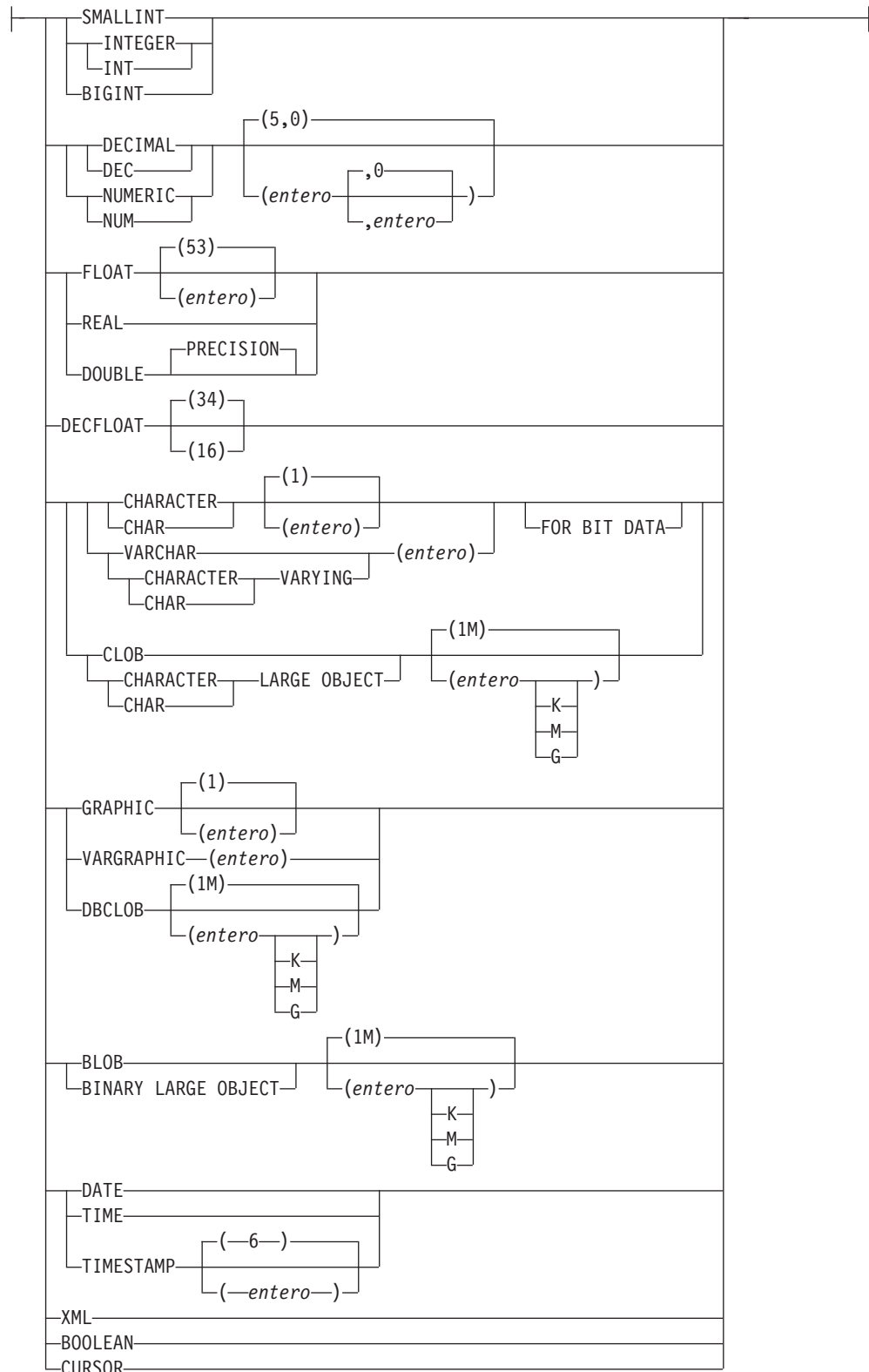
definición-tipo-matriz:



tipo-datos1:

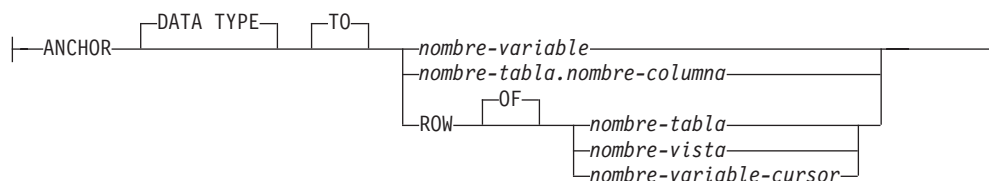


tipo-incorporado:

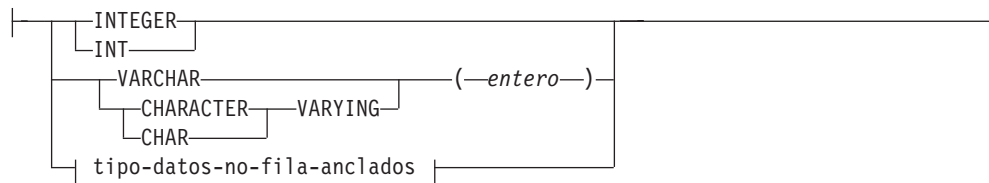


tipo-datos-anclados:

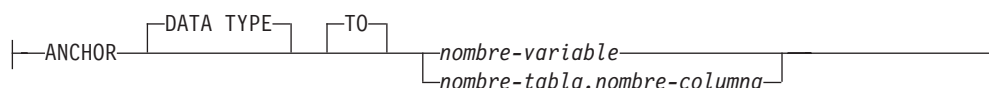
SQL compuesto (compilado)



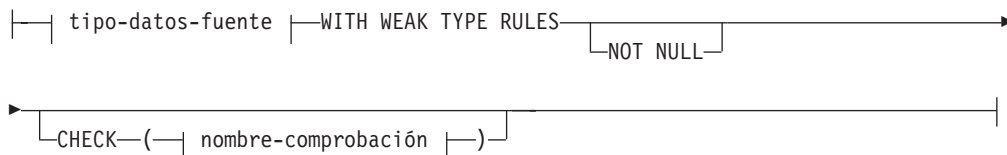
tipo-datos-2:



tipo-datos-no-fila-anclados:



definición-tipo-diferenciado:



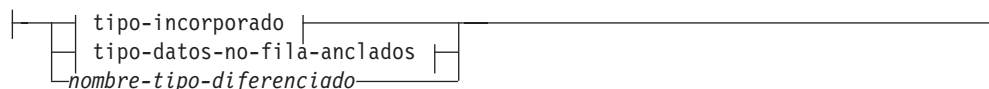
tipo-datos-fuente:



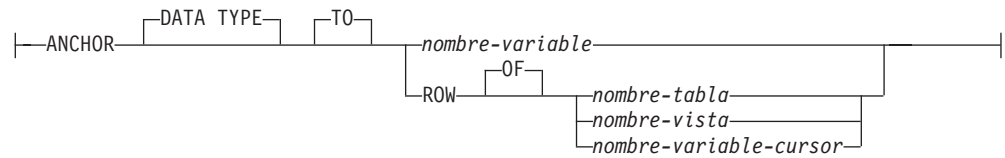
definición-tipo-fila:



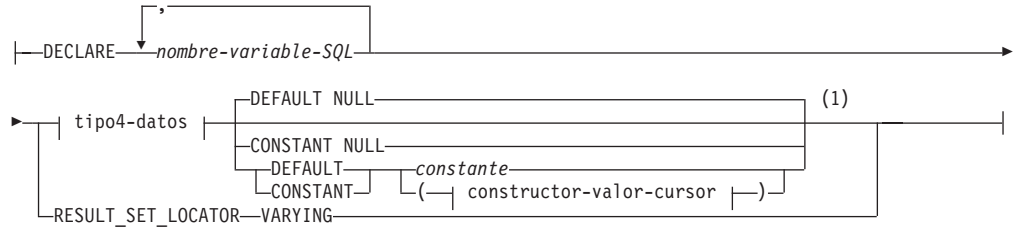
tipo3-datos:



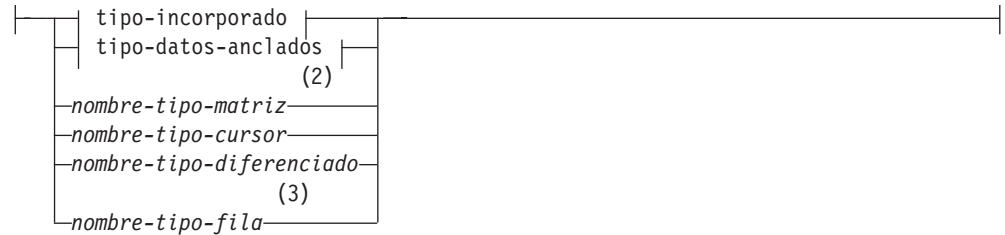
tipo-datos-fila-anclados:



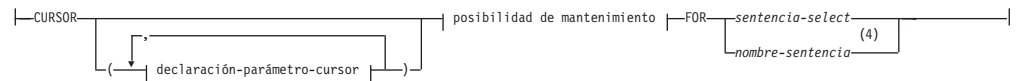
declaración-variable-SQL:



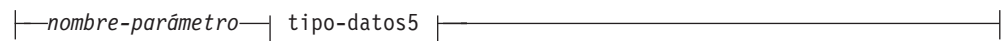
tipo4-datos:



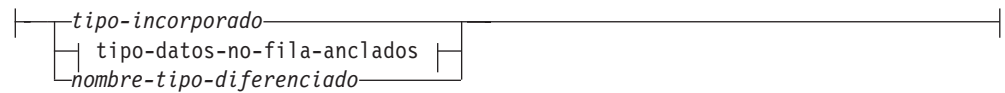
constructor-valor-cursor:



declaración-parámetro-cursor:



tipo-datos5:

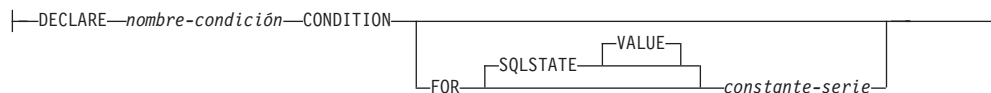


posibilidad de mantenimiento:



SQL compuesto (compilado)

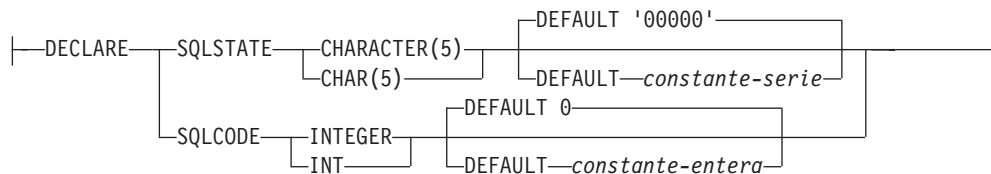
declaración-condición:



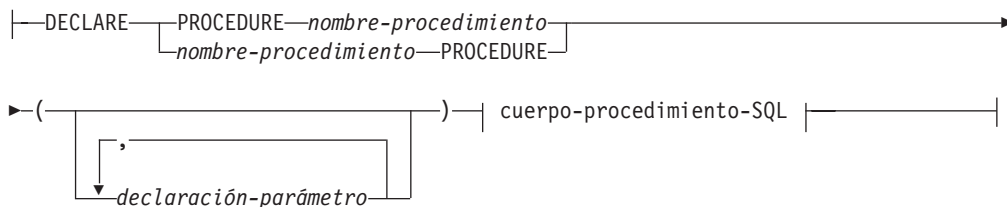
declaración-sentencia:



declaración-códigos-retorno:



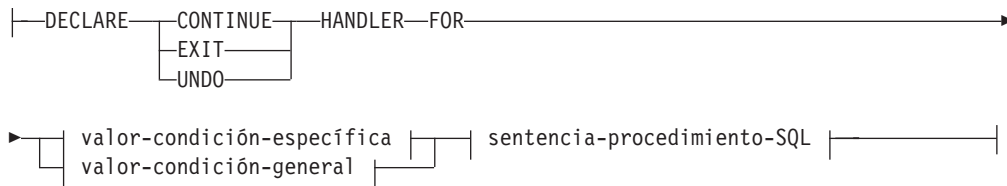
declaración-procedimiento:



cuerpo-procedimiento-SQL:

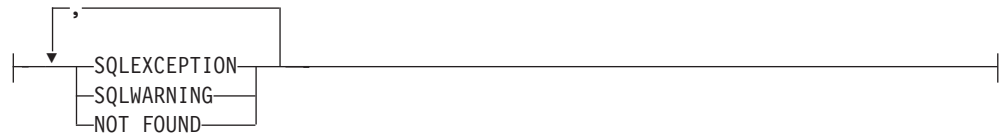
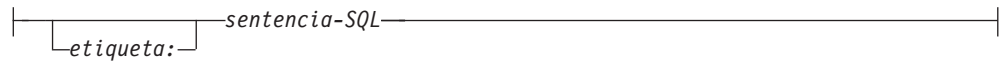


declaración-manejador-condiciones:



valor-condición-específica:



valor-condición-general:**sentencia-procedimiento-SQL:****Notas:**

- 1 Si *tipo-datos4* especifica un tipo incorporado **CURSOR** o un *nombre-tipo-cursor*, solamente pueden especificarse **NULL** o *constructor-valor-cursor*. Sólo puede especificarse de forma explícita **DEFAULT NULL** para *nombre-tipo-matriz* o *nombre-tipo-fila*.
- 2 Sólo puede especificarse de forma explícita **DEFAULT NULL** para *nombre-tipo-matriz*.
- 3 Sólo puede especificarse de forma explícita **DEFAULT NULL** para *nombre-tipo-fila*.
- 4 El *nombre-sentencia* no se puede especificar si se ha especificado la *declaración-parámetro-cursor*.

Descripción*etiqueta*

Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, se puede utilizar para calificar variables de SQL declaradas en la sentencia compuesta y también se puede especificar en una sentencia **LEAVE**. Si se especifica una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC o NOT ATOMIC

ATOMIC indica que si se produce una condición de excepción no manejada en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta.

NOT ATOMIC indica que una condición de excepción no manejada dentro de la sentencia compuesta no da lugar a la retrotracción de la sentencia compuesta.

Si se especifica la palabra clave **ATOMIC** en una sentencia compuesta preparada dinámicamente o una función de SQL que no está en un módulo, la sentencia compuesta se procesa como si se tratara de una sentencia de SQL compuesto (en línea).

Una sentencia compuesta que se utiliza en el cuerpo de la función de una función de tabla de módulo solo puede definirse como **NOT ATOMIC**.

declaración-tipo

Declara un tipo de datos definido por el usuario que es local respecto a la sentencia compuesta.

nombre-tipo

Especifica el nombre de un tipo de datos definido por el usuario local.

SQL compuesto (compilado)

El nombre no puede ser igual que cualquier otro tipo declarado dentro de la sentencia compuesta actual (SQLSTATE 42734). El *nombre-tipo* sin calificar tiene las mismas restricciones que las descritas en cualquier sentencia CREATE TYPE (SQLSTATE 42939).

definición-tipo-matriz

Especifica los atributos de un tipo de datos de matriz que se deben asociar con el *nombre-tipo*. Consulte "CREATE TYPE (matriz)" para obtener una descripción de los elementos de sintaxis. El *nombre-tipo-fila* puede hacer referencia a un tipo de fila declarado que se haya declarado previamente y que esté en el ámbito de la sentencia de SQL compuesto (compilado) actual. El *nombre-variable* especificado está en una cláusula de tipo-datos-anclados puede hacer referencia a una variable local en el ámbito de la sentencia de SQL compuesto (compilado) actual.

definición-tipo-diferenciado

Especifica las restricciones de tipo de fuente y de tipo de datos opcionales de un tipo diferenciado con tipo no firme que ha de asociarse al *nombre-tipo*. Consulte el apartado "CREATE TYPE (diferenciado)" para obtener una descripción completa de los elementos de la sintaxis. El *nombre-variable* especificado en una cláusula de tipo-datos-anclados puede hacer referencia a una variable local en el ámbito de la sentencia de SQL compuesto (compilado) actual. El tipo de datos del *nombre-variable* o *nombre-columna* de anclaje debe ser un tipo de datos incorporado.

definición-tipo-fila

Especifica los campos de un tipo de datos de fila que se deben asociar con el *nombre-tipo*. Consulte "CREATE TYPE (fila)" para obtener una descripción completa de los elementos de sintaxis. El *nombre-variable* especificado está en una cláusula de tipo-datos-no-fila anclados puede hacer referencia a una variable local en el ámbito de la sentencia de SQL compuesto (compilado) actual.

declaración-variable-SQL

Declara una variable que es local respecto a la sentencia compuesta.

nombre-variable-SQL

Define el nombre de una variable local. Todos los nombres de variables de SQL se convierten en mayúsculas. El nombre no puede ser el mismo que otra variable de SQL existente en la misma sentencia compuesta y no puede ser igual que un nombre de parámetro. El nombre de una variable de SQL no debe ser el mismo que el de un nombre de columna. Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable de SQL y una referencia a columna, el identificador se interpreta como una columna. Si la sentencia compuesta en la que se ha declarado la variable se ha etiquetado, las referencias a la variable pueden calificarse con la etiqueta. Por ejemplo, a la variable V declarada en una sentencia compuesta que se ha etiquetado con C puede hacerse referencia como C.V.

tipo4-datos

Especifica el tipo de datos de la variable. No se puede especificar tipos estructurados ni tipos de referencia (SQLSTATE 429BB).

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado excepto BOOLEAN y

CURSOR, que no pueden especificarse para una tabla, consulte el apartado "CREATE TABLE". No puede especificarse el tipo de datos XML en sentencias de SQL compuesto (compilado) utilizadas en un activador, en una función o como sentencia independiente (SQLSTATE 429BB). El tipo de datos XML puede especificarse cuando la sentencia de SQL compuesto (compilado) se utiliza en el cuerpo de un procedimiento de SQL.

BOOLEAN

Para un booleano.

CURSOR

Para un cursor.

tipo-datos-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos de la variable de SQL. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican a la especificación del tipo de datos directamente o, en el caso de una fila, a la creación de un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable de SQL, un parámetro de SQL o una variable global. El tipo de datos de la variable a la que se hace referencia se utiliza como tipo de datos para el *nombre-variable-SQL*.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el *nombre-variable-SQL*.

ROW OF nombre-tabla o nombre-vista

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. El tipo de datos del *nombre-variable-SQL* es un tipo de fila sin nombre.

ROW OF nombre-variable-cursor

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los elementos siguientes (SQLSTATE 428HS):

- Una variable de SQL o una variable global con un tipo de datos de cursor con tipo firme.
- Una variable de SQL o una variable declarada localmente con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula CONSTANT especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable de cursor no es de tipo firme que utiliza un tipo de fila con nombre, el tipo de datos de *nombre-variable-SQL* es un tipo de fila sin nombre.

SQL compuesto (compilado)

nombre-tipo-matriz

Especifica el nombre de un tipo de matriz definido por el usuario. El tipo de datos de matriz puede ser un tipo de datos declarado localmente, un tipo de datos de esquema o un tipo de datos de módulo.

nombre-tipo-cursor

Especifica el nombre de un tipo de cursor. El tipo de datos de cursor puede ser un tipo de datos de esquema o un tipo de datos de módulo.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. El tipo de datos diferenciado puede ser un tipo de datos de esquema o un tipo de datos de módulo. La longitud, la precisión y la escala de la variable declarada son, respectivamente, la longitud, la precisión y la escala del tipo fuente del tipo diferenciado.

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. El tipo de datos de fila puede ser un tipo de datos declarado localmente, un tipo de datos de esquema o un tipo de datos de módulo. Los campos de la variable son los campos del tipo de fila.

DEFAULT o CONSTANT

Especifica un valor para la variable de SQL cuando se hace referencia a la sentencia de SQL compuesto (compilado). Si no se especifica ninguna, el valor por omisión para la variable de SQL es el valor nulo. Sólo puede especificarse de forma explícita DEFAULT NULL si está especificado *nombre-tipo-matriz* o *nombre-tipo-fila*.

DEFAULT

Define el valor por omisión de la variable de SQL. La variable se inicializa cuando se hace referencia a la sentencia de SQL compuesto (compilado). El valor por omisión debe ser una asignación compatible con el tipo de datos de la variable.

CONSTANT

Especifica que la variable de SQL tiene un valor fijo que no se puede cambiar. Una variable de SQL que se defina mediante CONSTANT no puede utilizarse como destino de una operación de asignación. El valor fijo debe ser compatible con la asignación al tipo de datos de la variable.

NULL

Especifica NULL como valor por omisión para la variable de SQL.

constante

Especifica una constante como valor por omisión para la variable de SQL. Si *tipo-datos4* especifica un tipo incorporado CURSOR o un *nombre-tipo-cursor*, no puede especificarse la *constante* (SQLSTATE 42601).

constructor-valor-cursor

Un *constructor-valor-cursor* especifica la *sentencia-select* asociada con la variable de SQL. La asignación de un *constructor-valor-cursor* a una variable de cursor define el cursor subyacente de esa variable de cursor.

(declaración-parámetro-cursor, ...)

Especifica los parámetros de entrada del cursor, incluido el nombre y el tipo de datos de cada parámetro. Sólo se pueden especificar

parámetros de entrada con nombre si la *sentencia-select* también está especificada en el *constructor-valor-cursor* (SQLSTATE 428HU).

nombre-parámetro

Asigna un nombre al parámetro de cursor que se debe utilizar como variable de SQL dentro de la *sentencia-select*. El nombre no puede ser igual que ningún otro nombre de parámetro del cursor. Los nombres deben elegirse también evitando nombres de columna que se puedan utilizar en la *sentencia-select*, ya que los nombres de columna se resuelven antes que los nombres de parámetro.

tipo-datos5

Especifica el tipo de datos del parámetro de cursor utilizado dentro de una *sentencia-select*. No se pueden especificar tipos estructurados ni tipos de referencia (SQLSTATE 429BB).

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Los tipos incorporados BOOLEAN y CURSOR no pueden especificarse (SQLSTATE 429BB).

tipo-datos-no-fila-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos del parámetro de cursor. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable de SQL local, un parámetro SQL o una variable global. El tipo de datos de la variable a la que se hace referencia se utiliza como tipo de datos para el parámetro de cursor.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el parámetro de cursor.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

posibilidad de mantenimiento

Especifica si se impedirá que el cursor se cierre como consecuencia de una operación de confirmación. Consulte el apartado "DECLARE CURSOR" para obtener más información. El valor por omisión es WITHOUT HOLD.

WITHOUT HOLD

No impide que el cursor se cierre como consecuencia de una operación de confirmación.

SQL compuesto (compilado)

WITH HOLD

Mantiene recursos en varias unidades de trabajo. Impide que el cursor se cierre como consecuencia de una operación de confirmación.

sentencia-select

Especifica la sentencia SELECT del cursor. Consulte el apartado "sentencia-select" para obtener más información. Si se incluye una *declaración-parámetro-cursor* en el *constructor-valor-cursor*, la *sentencia-select* no debe incluir ninguna variable de SQL local ni parámetros de SQL de rutina (SQLSTATE 42704).

nombre-sentencia

Especifica la *sentencia-select* preparada del cursor. Consulte el apartado "PREPARE" para obtener una explicación de las sentencias preparadas. La variable del cursor de destino no debe tener un tipo de datos que sea un tipo de cursor definido por el usuario con tipo firme (SQLSTATE 428HU). No se deben especificar parámetros de entrada con nombre en el *constructor-valor-cursor* si se especifica un *nombre-sentencia* (SQLSTATE 428HU).

RESULT_SET_LOCATOR VARYING

Especifica el tipo de datos de una variable localizadora de conjuntos de resultados.

declaración-condición

Declara el nombre de una condición con un valor SQLSTATE opcional asociado.

nombre-condición

Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una condición dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42737).

CONDITION FOR SQLSTATE VALUE*constante-serie*

Especifica el SQLSTATE que está asociado a la condición. La constante de serie debe especificarse como cinco caracteres entre comillas simples, y la clase de SQLSTATE (los dos primeros caracteres) no deben ser '00'. Si esta cláusula no se especifica, la condición no tendrá un valor SQLSTATE asociado.

declaración-sentencia

Declara una lista de uno o más nombres que son locales en la sentencia compuesta. Cada nombre en *nombre-sentencia* no debe ser el mismo que cualquier otro nombre de sentencia declarado 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 declarar en la sentencia compuesta más externa del cuerpo del procedimiento de SQL cuando existen sentencias de SQL compuesto (compilado) anidadas, por ejemplo, en un cuerpo de procedimiento de SQL. Estas variables sólo se pueden declarar una vez para cada procedimiento de SQL.

sentencia-declare-cursor

Declara un cursor incorporado en el cuerpo del procedimiento. Las variables de los tipos de datos de cursor definidos por el usuario se declaran con las sentencias *declaración-variable-SQL*.

Todos los cursores declarados han de tener un nombre exclusivo dentro de la sentencia compuesta en la que están declarados, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al cursor dentro de la sentencia compuesta en la que está declarado, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 34000).

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 de resultados a la aplicación cliente, el cursor se debe declarar utilizando la cláusula WITH RETURN. El ejemplo siguiente devuelve un conjunto de resultados 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 conjuntos de resultados debe grabar la aplicación cliente utilizando una de las interfaces de programación de aplicaciones siguientes: DB2 Call Level Interface (DB2 Call Level Interface), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC) o SQL incorporado para Java (SQLJ).

Para obtener más información acerca de la declaración de un cursor, consulte el apartado "DECLARE CURSOR".

declaración-procedimiento

Declara un procedimiento que es local respecto a la sentencia compuesta. La definición de un procedimiento local no incluye la especificación de ninguna de las opciones posibles en una sentencia "CREATE PROCEDURE (SQL)". Las opciones tienen los mismos valores por omisión que tendrían para una sentencia "CREATE PROCEDURE (SQL)" con la excepción de MODIFIES SQL DATA. El nivel de acceso a datos para el procedimiento queda determinado de forma automática en el nivel mínimo necesario para procesar el cuerpo del procedimiento de SQL.

nombre-procedimiento

Define los nombres de un procedimiento local. El nombre debe especificarse sin ninguna calificación (SQLSTATE 42601). La signature de procedimiento, que consta del *nombre-procedimiento* y el número de parámetros declarados, debe ser exclusiva dentro de la sentencia compuesta actual. Las sentencias compuestas externas en las que está anidada la sentencia compuesta actual no pueden contener un procedimiento con el mismo nombre.

declaración-parámetro

Especifica los parámetros del procedimiento local. Consulte "CREATE PROCEDURE (SQL)" para obtener una descripción de los elementos de sintaxis. El tipo de datos de parámetro puede ser un tipo de datos declarado localmente en el ámbito de la sentencia compuesta actual.

SQL compuesto (compilado)

cuerpo-procedimiento-SQL

Especifica la sentencia de SQL que forma el cuerpo del procedimiento de SQL. Los nombres a los que se hace referencia en el *cuerpo-procedimiento-SQL* pueden hacer referencia a objetos declarados (como variables declaradas, tipos de datos y procedimientos) que se han declarado anteriormente y que están en el ámbito de la sentencia compuesta en la que se declara el procedimiento local.

declaración-manejador

Especifica un *manejador* y un conjunto de una o varias *sentencias-procedimiento-SQL* que debe ejecutarse cuando se produzca una condición de excepción o de terminación en la sentencia compuesta. *sentencia-procedimiento-SQL* es una sentencia que se ejecuta cuando el manejador de condiciones recibe el control.

Se dice que un manejador está activo mientras dura la ejecución del conjunto de *sentencias-procedimiento-SQL* que sigue al conjunto de *declaraciones-manejador* dentro de la sentencia compuesta en la que se ha declarado el manejador, incluidas las sentencias compuestas anidadas.

Existen tres tipos de manejadores de condiciones:

CONTINUE

Tras la invocación satisfactoria del manejador 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 manejador de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el manejador de condiciones.

UNDO

Antes de invocar el manejador de condiciones, se retrotraen los cambios de SQL que se hicieron en la sentencia compuesta. Tras la invocación satisfactoria del manejador de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el manejador de condiciones. Si se especifica UNDO, la sentencia compuesta en la que se declara el manejador de condiciones debe ser ATOMIC.

Las condiciones que dan lugar a la activación del manejador de condiciones se definen en la declaración-manejador-condiciones, tal como se indica a continuación:

valor-condición-específica

Especifica que el manejador es un *manejador de condiciones específicas*.

SQLSTATE VALUE*constante-serie*

Especifica un SQLSTATE para el cual se invoca el manejador de condiciones. Los dos primeros caracteres del valor SQLSTATE no deben ser '00'.

nombre-condición

Especifica una condición para la cual se invoca el manejador de condiciones. El nombre de la condición debe estar definido previamente en una declaración de condición o debe identificar una condición que existe en el servidor actual.

valor-condición-general

Especifica que el manejador es un *manejador de condiciones generales*.

SQL EXCEPTION

Especifica que el manejador se invoca cuando se produce una condición de excepción. Un valor SQLSTATE representa una condición de excepción cuyos dos primeros caracteres no son '00', '01' o '02'.

SQL WARNING

Especifica que el manejador se invoca cuando se produce una condición de aviso. Un valor SQLSTATE representa una condición de aviso cuyos dos primeros caracteres son '01'.

NOT FOUND

Especifica que el manejador de condiciones se invoca cuando se produce una condición de "no encontrado" (NOT FOUND). Un valor SQLSTATE representa una condición NOT FOUND cuyos dos primeros caracteres son '02'.

sentencia-procedimiento-SQL

Especifica la sentencia de procedimiento de SQL.

etiqueta

Especifica una etiqueta para la sentencia de procedimiento de SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento de SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control de SQL admiten la especificación de una lista de sentencias de procedimiento de SQL.

sentencia-SQL

Todas las sentencias de SQL ejecutables, excepto:

- ALTER
- CONNECT
- CREATE
- DESCRIBE
- DISCONNECT
- DROP
- FLUSH EVENT MONITOR
- GRANT
- REFRESH TABLE
- RELEASE (sólo conexión)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY
- SET PASSTHRU
- SET SERVER OPTION
- TRANSFER OWNERSHIP

Las sentencias ejecutables siguientes no están soportadas en sentencias independientes de SQL compuesto (compilado), pero sí están soportadas

SQL compuesto (compilado)

en sentencias de SQL compuesto (compilado) utilizadas en una función de SQL, un procedimiento de SQL o un activador:

- CREATE de un índice, tabla o vista
- DROP de un índice, tabla o vista
- GRANT
- ROLLBACK

La sentencia ROLLBACK tampoco está soportada en cualquier sentencia anidada invocada dentro de la sentencia independiente de SQL compuesto (compilado).

Las sentencias siguientes, que son sentencias no ejecutables, están soportadas en sentencias de SQL compuesto (compilado):

- ALLOCATE CURSOR
- ASSOCIATE LOCATORS

Normas

- Las sentencias compuestas definidas como ATOMIC no se pueden anidar.
- Las normas siguientes son aplicables a la declaración de un manejador de condiciones:
 - Una declaración de manejador de condiciones no puede contener el mismo *nombre-condición* o valor SQLSTATE más de una vez y no puede contener un valor SQLSTATE y un *nombre-condición* que representen el mismo valor SQLSTATE.
 - Cuando se declaran dos o más manejadores de condiciones en una sentencia compuesta:
 - Dos declaraciones de manejador no pueden especificar la misma categoría de condición general (SQLEXCEPTION, SQLWARNING, NOT FOUND).
 - Dos declaraciones de manejador de condiciones no pueden especificar la misma condición específica, ya sea como un valor SQLSTATE o como un *nombre-condición* que represente el mismo valor.
 - Un manejador de condiciones se activa cuando es el manejador más apropiado para una condición de excepción o terminación. El manejador más adecuado se determina basándose en las consideraciones siguientes:
 - El ámbito de una declaración de manejador *H* es la lista de la *sentencia-procedimiento-SQL* que sigue a las declaraciones de manejador contenidas dentro de la sentencia compuesta en la que aparece *H*. Esto significa que el ámbito de *H* no incluye las sentencias contenidas en el cuerpo del manejador de condiciones *H*, lo que implica que un manejador de condiciones no puede manejar las condiciones que se producen dentro de su propio cuerpo. De forma similar, en el caso de dos manejadores cualesquiera, *H1* y *H2*, declarados en la misma sentencia compuesta, *H1* no manejará las condiciones que se produzcan en el cuerpo de *H2* y *H2* no manejará las condiciones que se produzcan en el cuerpo de *H1*.
 - Un manejador de condiciones para un *valor-condición-específica* o un *valor-condición-general* *C* declarado en un ámbito interno tiene prioridad respecto a cualquier otro manejador de condiciones para *C* declarado en un ámbito de inclusión.
 - Cuando, en el mismo ámbito, se declaran un manejador de condiciones específico para una condición *C* y un manejador de condiciones general que también manejaría *C*, el manejador de condiciones específico tiene prioridad respecto al manejador de condiciones general.

- Cuando se declaran un manejador de una condición de módulo que no tiene un valor SQLSTATE asociado y un manejador de SQLSTATE 45000 en el mismo ámbito, el primero tiene prioridad sobre el segundo.

Si se produce una condición de excepción para la que no existe un manejador adecuado, el procedimiento de SQL que contiene la sentencia anómala finaliza con una condición de excepción no manejada. Si se produce una condición de finalización para la que no existe un manejador de condiciones adecuado, la ejecución continuará con la siguiente sentencia de SQL.

- La referencia a variables o a parámetros del tipo de datos XML en los procedimientos de SQL después de que se haya producido una operación de confirmación o de retroacción, sin que primero se hayan asignado valores nuevos a estas variables, no tiene soporte (SQLSTATE 560CE).
- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.
- Si se utilizan marcadores de parámetros con nombre en una sentencia de SQL compuesto (compilado) que se prepara o ejecuta dinámicamente, el nombre de cada marcador de parámetro debe ser exclusivo (SQLSTATE 42997).

Notas

- **Asignaciones de XML:** la asignación a parámetros y variables de tipo de datos XML se realiza por referencia.

Los parámetros del tipo de datos XML de una sentencia CALL se pasan a un procedimiento de SQL por referencia. Cuando los valores XML se pasan por referencia, los árboles de nodos de entrada se utilizan directamente desde el argumento XML. Este uso directo mantiene todas las propiedades, incluyendo el orden de documentos, las identidades de nodo originales y todas las propiedades padre.

Ejemplos

- *Ejemplo 1:* crear un procedimiento con una sentencia de SQL compuesto (compilado) que lleva a cabo las acciones siguientes:
 1. Declara variables de 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 manejador 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 de 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;
```

SQL compuesto (compilado)

```
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
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
```

- *Ejemplo 2:* en el ejemplo siguiente se muestra el flujo de ejecución en un caso hipotético donde se ha activado un manejador UNDO desde otra condición como resultado de RESIGNAL:

```
CREATE PROCEDURE A()
LANGUAGE SQL
CS1: BEGIN ATOMIC
  DECLARE C CONDITION FOR SQLSTATE '12345';
  DECLARE D CONDITION FOR SQLSTATE '23456';

  DECLARE UNDO HANDLER FOR C
  H1: BEGIN
    -- Realizar retrotracción tras error, realizar limpieza final y salir
    -- de procedimiento A.

    -- ...

    -- Cuando este manejador finaliza, la ejecución continúa después
    -- de la sentencia compuesta CS1; el procedimiento A terminará.
  END;

  -- Realizar aquí algún trabajo ...
CS2: BEGIN
  DECLARE CONTINUE HANDLER FOR D
  H2: BEGIN
    -- Realizar recuperación local y reenviar la condición de
    -- error al manejador externo para su proceso
    -- adicional.

    -- ...

    RESIGNAL C; -- activará el manejador UNDO H1; la ejecución
    -- NO VOLVERÁ aquí. Los cursores locales
    -- declarados en H2 y CS2 se cerrarán.
  END;

  -- Realizar aquí algún trabajo adicional ...

  -- Simular la generación de la condición D por parte de alguna
  -- sentencia de SQL en la sentencia compuesta CS2:
  SIGNAL D; -- activará H2
END;
END
```

CONNECT (tipo 1)

La sentencia CONNECT (Tipo 1) conecta un proceso de aplicación con el servidor de aplicaciones identificado según las normas 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 la aplicación puede conectarse explícitamente a otro servidor de aplicaciones ejecutando una sentencia CONNECT. Una conexión dura hasta que se ejecuta una sentencia CONNECT RESET o una sentencia DISCONNECT o hasta que otra sentencia CONNECT cambia el servidor de aplicaciones.

Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente. Cuando se invoca utilizando el procesador de línea de mandatos, se pueden especificar opciones adicionales. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

Autorización

El proceso de CONNECT pasa por dos niveles de control de acceso. Ambos niveles deben cumplirse para que la conexión sea correcta.

El primer nivel de control de acceso es la autenticación, en la que el ID de usuario asociado a la conexión debe autenticarse correctamente en función del método de autenticación configurado para el servidor. Cuando se produce la autenticación correcta, se calcula un ID de autorización de DB2 a partir del ID de usuario de conexión, en función del conector de autenticación en vigor en el servidor. A continuación, este ID de autorización de DB2 debe pasar el segundo nivel de control de acceso de la conexión, es decir, la autorización. Para ello, este ID de autorización debe tener como mínimo una de las autorizaciones siguientes:

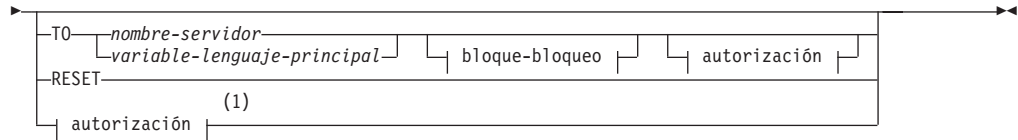
- Autorización CONNECT
- Autorización SECADM
- Autorización DBADM
- Autorización SYSADM
- Autorización SYSCTRL
- Autorización SYSMANT
- Autorización SYSMON

Nota: En el caso de una base de datos dividida en particiones, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones de la base de datos.

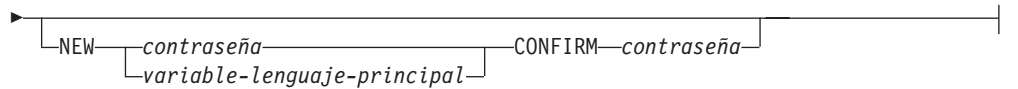
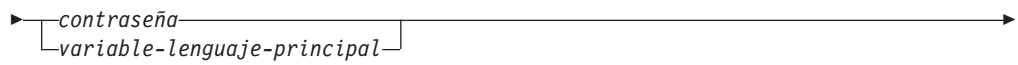
Sintaxis

►►—CONNECT—►

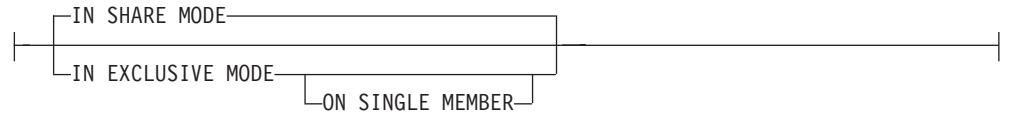
CONNECT (tipo 1)



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 alias 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 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 territorio y la página de códigos del servidor de aplicaciones se encuentran en el campo SQLERRMC (ya que están con una sentencia CONNECT correcta).

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 o de región y la página de códigos del servidor de aplicaciones se ponen en el campo SQLERRMC, como una sentencia CONNECT 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-lenguaje-principal*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lenguaje-principal* que contenga el nombre-servidor.

Si se especifica una *variable-lenguaje-principal*, 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 de indicador. El *nombre-servidor* contenido dentro de la *variable-lenguaje-principal* debe alinearse a la izquierda y no debe delimitarse mediante comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

Cuando se ejecuta la sentencia CONNECT, el proceso de aplicación debe estar en un estado conectable.

Conexión satisfactoria

Si la sentencia CONNECT es satisfactoria:

- 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 alias) 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 z/OS
 - ARI para DB2 Server para VSE & VM
 - QSQ para DB2 para i
 - SQL para DB2 para Linux, UNIX y Windows
 - *vv* es un identificador de versión de dos dígitos como, por ejemplo, '08'
 - *rr* es un identificador de release de dos dígitos como, por ejemplo, '01'
 - *m* es un identificador de nivel de modificación de un único carácter como, por ejemplo, '0'.

Por ejemplo, la versión 9.5 de DB2 para Linux, UNIX y Windows se identifica como 'SQL09050'.

- El campo SQLERRMC de la SQLCA se establece en los valores siguientes (separados por X'FF')
1. El código de país o región del servidor de aplicaciones (o espacios en blanco si se utiliza DB2 Connect)

CONNECT (tipo 1)

2. La página de códigos del servidor de aplicaciones (o CCSID si se utiliza DB2 Connect)
3. El ID de autorización (sólo los primeros 8 bytes como máximo)
4. El alias de base de datos
5. El tipo de plataforma del servidor de aplicaciones. Los valores reconocidos actualmente son:

Símbolo

Servidor

QAS DB2 para i

QDB2 DB2 para z/OS

QDB2/6000

DB2 Database para AIX

QDB2/HPUX

DB2 Database para HP-UX

QDB2/LINUX

DB2 Database para Linux

QDB2/NT

DB2 Database para Windows

QDB2/SUN

DB2 Database para el sistema operativo Solaris

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 **agent_id** que devuelve el supervisor de bases de datos.
7. El índice de agente. Identifica el índice del agente y se utiliza para el servicio.
8. Si la instancia de servidor funciona en un entorno DB2 pureScale, tal como indican los valores de SQLWARN0 y SQLWARN4 establecidos en 'W' y 'S' respectivamente, este valor representa el número de miembro. Si, tal como indica el símbolo 10, la instancia de servidor funciona en un entorno particionado, este símbolo representa el número de nodo. Si la instancia de servidor funciona en un entorno no particionado y fuera de un entorno DB2 pureScale, este valor no es aplicable y siempre es 0.
9. La página de códigos de la aplicación cliente.
10. Si este valor es cero, la instancia de servidor funciona en un entorno no particionado y fuera de un entorno DB2 pureScale. De lo contrario, este valor no cero representa el número de miembros de una instancia de DB2 pureScale, si SQLWARN0 y SQLWARN4 se establecen en 'W' y 'S' respectivamente. Si este valor no es cero, pero no se han establecido los valores de SQLWARN0 ni SQLWARN4, representa el número de nodos en un entorno particionado. Este símbolo está presente sólo 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 datos CHAR) al convertirlos a la página de códigos de la base de datos a partir 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.
- 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. 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.
- 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 que no utiliza el último nivel, tiene confirmación de una fase y es 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 de TP).
 - 3 confirmación de dos fases.
- El campo SQLERRD(5) de la SQLCA devuelve el tipo de autenticación para la conexión. El valor es uno de los siguientes:
 - 0 Autenticado en el servidor.
 - 1 Autenticado en el cliente.
 - 2 Autenticado mediante DB2 Connect.
 - 4 Autenticado en el servidor con cifrado.
 - 5 Autenticado utilizando DB2 Connect con cifrado.
 - 7 Autenticado utilizando un mecanismo de seguridad Kerberos externo.
 - 9 Autenticado utilizando un mecanismo de seguridad de plugin de API GSS externo.
 - 11 Autenticado en el servidor, que acepta datos cifrados.
 - 255 Autenticación no especificada.
- El campo SQLERRD(6) de SQLCA devuelve el número de la partición de base de datos con la que se ha establecido la conexión si se está en un entorno de base de datos particionada. De lo contrario, se devuelve un valor de 0.

CONNECT (tipo 1)

- 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.

Conexión no satisfactoria

Si la sentencia CONNECT 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. Los tres primeros caracteres del nombre del módulo identifican el producto.
- Si la sentencia CONNECT da un resultado anómalo porque el proceso de aplicación no está en estado conectable, no se cambiará el estado de conexión del proceso de aplicación.
- Si la sentencia CONNECT no se ejecuta satisfactoriamente porque el *nombre-servidor* no se encuentra en el directorio local, se emitirá un mensaje de error (SQLSTATE 08001) y el estado de conexión del proceso de la aplicación no cambiará:
 - 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 no se ejecuta satisfactoriamente por cualquier otro motivo, el proceso de la aplicación pasará a estado de desconexión.

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 MODE

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. DB2 Connect no da soporte a esta opción.

ON SINGLE MEMBER

Especifica que el miembro de base de datos coordinador está conectado en modalidad exclusiva y todos los demás miembros están conectados en modalidad de compartimiento. Si la base de datos no está en un entorno particionado ni en un entorno DB2 pureScale, esta opción puede especificarse aunque no tiene ningún efecto.

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-lenguaje-principal*

Identifica el ID de usuario que intenta conectarse con el servidor de aplicaciones. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres, que no incluye una variable de indicador. El ID de usuario contenido dentro de la *variable-lenguaje-principal* debe alinearse a la izquierda y no debe delimitarse mediante comillas.

USING *contraseña/variable-lenguaje-principal*

Identifica la contraseña del ID de usuario que intenta conectarse con el servidor de aplicaciones. La *contraseña* o la *variable-lenguaje-principal* pueden tener una longitud de hasta 14 bytes. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 14 y no debe incluir una variable de indicador.

NEW *contraseña/variable-lenguaje-principal* **CONFIRM** *contraseña*

Identifica la nueva contraseña que debe asignarse al ID de usuario que se identifica por medio de la opción USER. La *contraseña* o la *variable-lenguaje-principal* pueden tener una longitud de hasta 14 bytes. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 14 y no debe incluir una variable de indicador. El sistema en que se cambiará la contraseña depende de cómo esté configurada la autenticación de usuario. Se pueden asignar nuevas contraseñas utilizando esta cláusula en los servidores siguientes en los releases indicados (y posteriores): DB2 Universal Database Versión 8 en sistemas operativos AIX y Windows, DB2 Versión 9.1 Fixpack 3 o versiones posteriores en sistemas operativos Linux, DB2 para z/OS Versión 7, DB2 para i5/OS V6R1. A fin de dar soporte al cambio de contraseñas para los productos de base de datos DB2 en Linux, la instancia de DB2 debe estar configurada para utilizar los conectores de seguridad IBMOSchgpwdclient e IBMOSchgpwdserver.

Notas

- Es conveniente que la primera sentencia de SQL que ejecute un proceso de la aplicación sea la sentencia CONNECT.
- Si se emite una sentencia CONNECT al servidor de aplicaciones actual con otro ID de usuario y contraseña, la conversación se desasignará y se reasignará. 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 al servidor de aplicaciones actual con el mismo ID de usuario y contraseña, la conversación no se desasignará ni se reasignará. En este caso, los cursores no se cierran.
- Para utilizar un entorno de base de datos particionada con varias particiones, el usuario o la aplicación debe conectarse a una de las particiones de base de datos indicadas en el archivo db2nodes.cfg. Debe intentar garantizar que todos los usuarios no utilicen la misma partición de base de datos como partición coordinadora.
- El *nombre-autorización* SYSTEM no se puede especificar explícitamente en la sentencia CONNECT. No obstante, en sistemas operativos Windows, las aplicaciones locales que se ejecutan con la cuenta de sistema local pueden conectarse implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Cuando se establece conexión con Windows Server de forma explícita, el *nombre-autorización* o *variable-lenguaje-principal* de usuario puede especificarse utilizando el nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft Windows.
- Puede que no pueda accederse a la base de datos si ésta no se ha activado explícitamente, si una aplicación de cliente realiza reconexiones frecuentes o si el intervalo de tiempo entre emisiones de los mandatos **DEACTIVATE DATABASE** y **ACTIVATE DATABASE** se ha establecido en un valor muy breve. Active la base de datos mediante la emisión del mandato **ACTIVATE DATABASE** y, a continuación, intente conectarse con la base de datos.

CONNECT (tipo 1)

- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Se pueden especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* en un programa C, conectar el servidor de aplicaciones TOROLAB, utilizando el alias de base de datos TOROLAB, el ID de usuario FERMAT y la contraseña THEOREM.

```
EXEC SQL CONNECT TO TOROLAB USER FERMAT USING THEOREM;
```

- *Ejemplo 2:* en un programa C, conectar con un servidor de aplicaciones cuyo alias de base de datos esté almacenado en la variable del lenguaje principal APP_SERVER (varchar(8)). Después de haber establecido una conexión satisfactoria, copie el identificador del producto de 3 caracteres del servidor de aplicaciones en la variable PRODUCT (char(3)).

```
EXEC SQL CONNECT TO :APP_SERVER;  
if (strncmp(SQLSTATE, '00000', 5))  
    strncpy(PRODUCT, sqlca.sqlerrp, 3);
```

CONNECT (tipo 2)

La sentencia CONNECT (Tipo 2) conecta un proceso de aplicación con el servidor de aplicaciones identificado y establece las normas para una unidad de trabajo distribuida y dirigida por aplicación. Este servidor es entonces el servidor actual para el proceso.

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

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente. Cuando se invoca utilizando el procesador de línea de mandatos, se pueden especificar opciones adicionales. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

Autorización

El proceso de CONNECT pasa por dos niveles de control de acceso. Ambos niveles deben cumplirse para que la conexión sea correcta.

El primer nivel de control de acceso es la autenticación, en la que el ID de usuario asociado a la conexión debe autenticarse correctamente en función del método de autenticación configurado para el servidor. Cuando se produce la autenticación correcta, se calcula un ID de autorización de DB2 a partir del ID de usuario de conexión, en función del conector de autenticación en vigor en el servidor. A continuación, este ID de autorización de DB2 debe pasar el segundo nivel de control de acceso de la conexión, es decir, la autorización. Para ello, este ID de autorización debe tener como mínimo una de las autorizaciones siguientes:

- Autorización CONNECT
- Autorización SECADM
- Autorización DBADM
- Autorización SYSADM
- Autorización SYSCTRL
- Autorización SYSMAINT
- Autorización SYSMON

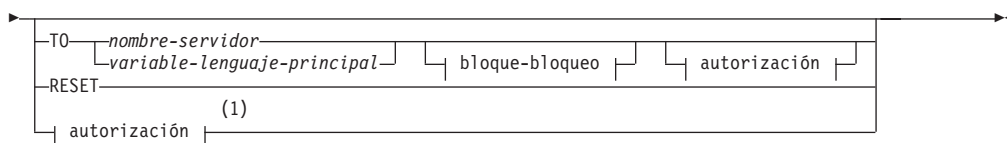
Nota: En el caso de una base de datos dividida en particiones, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones de la base de datos.

Sintaxis

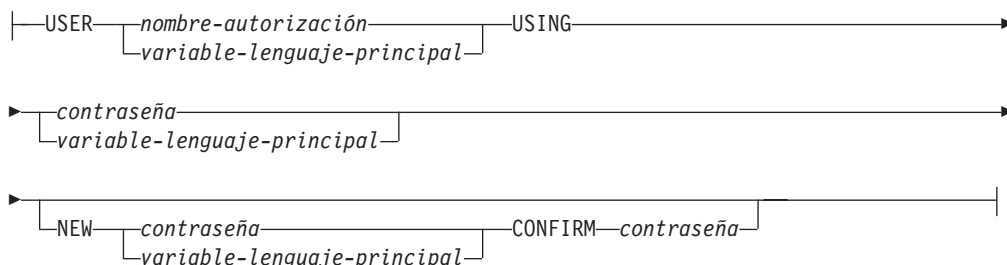
La selección entre el Tipo 1 y el Tipo 2 se determina según las opciones de precompilador. Si desea obtener una visión general de estas opciones, consulte "Conexión a bases de datos relacionales distribuidas".

►►—CONNECT—►►

CONNECT (tipo 2)



autorización:



bloque-bloqueo:



Notas:

1 Este formato sólo es válido si se ha habilitado la conexión implícita.

Descripción

T0 *nombre-servidor/variable-lenguaje-principal*

Las normas 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.

Para obtener información acerca de la especificación de SQLRULES, consulte el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida".

Conexión satisfactoria

Si la sentencia CONNECT 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 forma que con CONNECT (Tipo 1).

Conexión no satisfactoria

Si la sentencia CONNECT 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 alias 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 Tipo 2, pero está permitido utilizar 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.

Normas

- Como se describe en “Opciones que rigen la semántica de la unidad de trabajo distribuida”, un conjunto de opciones de conexión controla la semántica de la gestión de conexiones. 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 primero se haya ejecutado un mandato SET CLIENT o una API, las opciones de conexión que se utilizan al procesar previamente el archivo fuente que contiene la primera sentencia de SQL que se ejecuta en tiempo de ejecución se convierten en las opciones de conexión que están en vigor.

Si posteriormente se ejecuta una sentencia CONNECT desde un archivo fuente previamente procesado con opciones de conexión distintas sin que intervenga en la ejecución ningún mandato SET CLIENT o la API, se devuelve 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.

En el Ejemplo 1 del apartado “Ejemplos” de esta sentencia se muestran estas normas.

- Aunque la sentencia CONNECT se puede utilizar para establecer o conmutar conexiones, CONNECT con la cláusula USER/USING sólo se aceptará cuando no haya una conexión actual o inactiva al servidor indicado. La conexión debe liberarse antes de emitir una conexión con el mismo servidor con la cláusula

CONNECT (tipo 2)

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.

Comparación de las sentencias CONNECT de tipo 1 y de tipo 2:

La semántica de la sentencia CONNECT la determina la opción de precompilador CONNECT o la API SET CLIENT (consulte el apartado “Opciones que rigen la semántica de la unidad de trabajo distribuida”). Puede especificarse CONNECT Tipo 1 o CONNECT Tipo 2, y las sentencias CONNECT en esos programas se conocen como sentencias CONNECT Tipo 1 y Tipo 2 respectivamente. Su semántica se describe en las tablas siguientes:

Utilización de **CONNECT**:

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 debe 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 de tipo 1 si se la opción de precompilador SQLRULES se establece 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.	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. Si CONNECT se ejecuta para un servidor de aplicaciones que está en una conexión inactiva, esta se convierte en la conexión actual. 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.
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.

Uso de **CONNECT...USER...USING**:

Tipo 1	Tipo 2
Conectar con la cláusula USER...USING desconecta la conexión actual y establece una nueva conexión con el nombre de autorización y la contraseña proporcionados.	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	Tipo 2
CONNECT RESET puede utilizarse para desconectar la conexión actual.	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 retrotrae implícitamente la unidad de trabajo actual.	CONNECT RESET retrotrae implícitamente la unidad de trabajo actual.
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:

CONNECT (tipo 2)

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.

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.
- El *nombre-autorización* SYSTEM no se puede especificar explícitamente en la sentencia CONNECT. No obstante, en sistemas operativos Windows, las aplicaciones locales que se ejecutan con la cuenta de sistema local pueden conectarse implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Cuando se establece conexión con Windows Server de forma explícita, el *nombre-autorización* o *variable-lenguaje-principal* de usuario puede especificarse utilizando el nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft Windows.
- **Terminación de una conexión:** Cuando se termina una conexión y una transacción aún no se ha confirmado o retrotraído, consulte el siguiente apartado, "Utilización de CONNECT, CONNECT RESET implícitas y Desconexión", para obtener más información acerca de lo que les sucede a estas transacciones. Para garantizar un comportamiento coherente, codifique una sentencia COMMIT o ROLLBACK en lugar de depender de la sentencia CONNECT.
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Se pueden especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* En este ejemplo se muestra la utilización de varios programas fuente (se muestran en los recuadros), algunos procesados previamente con opciones de conexión distintas (se muestran en la sentencia que precede al código) y uno de los cuales contiene una llamada a la SET CLIENT API.

```
PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)
```

```
...  
exec sql CONNECT TO OTTAWA;  
exec sql SELECT col1 INTO :hv1  
FROM tb11;  
...
```

```
PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)
```

```

...
exec sql CONNECT TO QUEBEC;
exec sql SELECT col1 INTO :hv1
FROM tbl2;
...

```

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 tbl3;
...

```

Nota:

1. 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 tbl4;
...

```

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 de las sentencias CONNECT (tipo 2), SET CONNECTION, RELEASE y DISCONNECT. S0, S1, S2 y S3 representan cuatro servidores.

Secuencia	Sentencia	Servidor actual	Conexiones inactivas	Pendiente de liberación
0	• Sin sentencia	• Ninguna	• Ninguna	• Ninguna
1	• SELECT * FROM TBLA	• S0 (valor por omisión)	• Ninguna	• Ninguna
2	• CONNECT TO S1 • SELECT * FROM TBLB	• S1 • S1	• S0 • S0	• Ninguna • Ninguna

CONNECT (tipo 2)

Secuencia	Sentencia	Servidor actual	Conexiones inactivas	Pendiente de liberación
3	• CONNECT TO S2	• S2	• S0, S1	• Ninguna
	• UPDATE TBLC SET ...	• S2	• S0, S1	• Ninguna
4	• CONNECT TO S3	• S3	• S0, S1, S2	• Ninguna
	• SELECT * FROM TBLD	• S3	• S0, S1, S2	• Ninguna
5	• SET CONNECTION S2	• S2	• S0, S1, S3	• Ninguna
6	• RELEASE S3	• S2	• S0, S1	• S3
7	• COMMIT	• S2	• S0, S1	• Ninguna
8	• SELECT * FROM TBLE	• S2	• S0, S1	• Ninguna
9	• DISCONNECT S1	• S2	• S0	• Ninguna
	• SELECT * FROM TBLF	• S2	• S0	• Ninguna

CREATE ALIAS

La sentencia CREATE ALIAS define un alias para un módulo, apodo, secuencia, tabla, vista u otro alias. Los alias también se conocen con el nombre de sinónimos.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

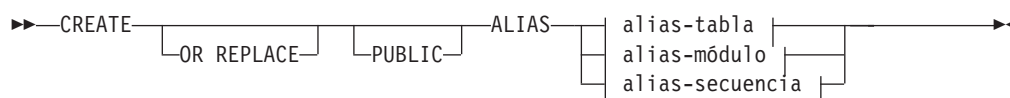
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del alias no existe
- El privilegio CREATEIN para el esquema, si el nombre del esquema del alias hace referencia a un esquema existente, o el privilegio CREATEIN en SYSPUBLIC, si se está creando un alias público
- Autorización DBADM

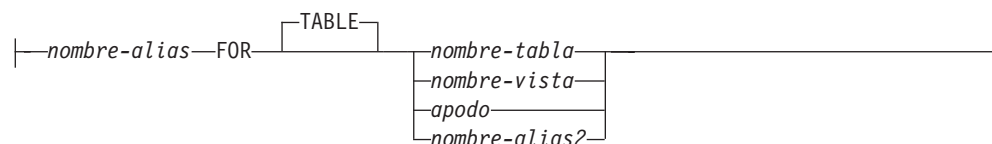
Los privilegios requeridos para utilizar el objeto al que se hace referencia mediante su alias son idénticos a los privilegios requeridos para utilizar el objeto directamente.

Para sustituir un alias existente, el ID de autorización de la sentencia debe ser el propietario del alias existente (SQLSTATE 42501).

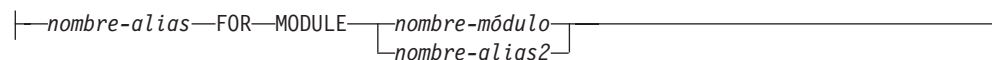
Sintaxis



alias-tabla:



alias-módulo:



alias-secuencia:

```
|—nombre-alias—FOR—SEQUENCE—|nombre-secuencia—|
|                               |nombre-alias2—|
```

Descripción**OR REPLACE**

Especifica que se debe sustituir la definición del alias si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que se sustituya la nueva definición en el catálogo. Esta opción se ignora si no existe una definición para el alias en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

PUBLIC

Especifica que el alias es un objeto del esquema del sistema SYSPUBLIC.

nombre-alias

Indica el nombre del alias. En el caso de los alias de tabla, el nombre no debe identificar un apodo, una tabla, una vista o un alias de tabla que ya exista en el servidor actual. En el caso de los alias de módulo, el nombre no debe identificar un módulo o un alias de módulo que ya exista en el servidor actual. En el caso de los alias de secuencia, el nombre no debe identificar una secuencia o un alias de secuencia que ya exista en el servidor actual.

Si se especifica un nombre de dos partes, el nombre del esquema no puede empezar por 'SYS' (SQLSTATE 42939) excepto si se especifica PUBLIC, en cuyo caso el nombre de esquema debe ser SYSPUBLIC (SQLSTATE 428EK).

FOR TABLE *nombre-tabla*, *nombre-vista*, *apodo* o *nombre-alias2*

Identifica la tabla, la vista, el apodo o el alias de tabla para el que se ha definido *nombre-alias*. En caso de proporcionar otro alias (*nombre-alias2*), este no debe ser el mismo que el nuevo *nombre-alias* que se está definiendo (en su forma completamente calificada). El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

FOR MODULE *nombre-módulo* o *nombre-alias2*

Identifica el módulo o el alias de módulo para el que se ha definido *nombre-alias*. En caso de proporcionar otro alias (*nombre-alias2*), este no debe ser el mismo que el nuevo *nombre-alias* que se está definiendo (en su forma completamente calificada).

FOR SEQUENCE *nombre-secuencia* o *nombre-alias2*

Identifica la secuencia o el alias de secuencia para el que se ha definido *nombre-alias*. En caso de proporcionar otro alias (*nombre-alias2*), este no debe ser el mismo que el nuevo *nombre-alias* que se está definiendo (en su forma completamente calificada). El *nombre-secuencia* no debe ser una secuencia generada por el sistema para una columna de identidad (SQLSTATE 428FB).

Notas

- La palabra clave PUBLIC se utiliza para crear un alias público (también conocido como sinónimo público). Si la palabra clave PUBLIC no se utiliza, el tipo de alias es un alias privado (también conocido como sinónimo privado).
- La definición del alias de tabla creado recientemente se almacena en SYSCAT.TABLES. La definición del alias de módulo creado recientemente se almacena en SYSCAT.MODULES. La definición del alias de secuencia creado recientemente se almacena en SYSCAT.SEQUENCES.

- Se puede definir un alias 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 alias; de lo contrario, aparece un mensaje de error (SQLSTATE 52004).
- Se puede definir un alias para hacer referencia a otro alias como parte de una cadena de alias, pero dicha cadena está sujeta a las mismas restricciones que un alias normal cuando se utiliza en una sentencia de SQL. La cadena de alias se resuelve de la misma manera que un solo alias. Si un alias utilizado en una sentencia de un paquete, una rutina de SQL, un activador, la expresión por omisión de una variable global o una definición de vista apunta a una cadena de alias, se registrará una dependencia para el paquete, la rutina de SQL, el activador, la variable global o la vista en cada alias de la cadena. Un alias no puede hacer referencia a sí mismo en una cadena de alias y un ciclo de este tipo se detecta en el momento de la definición de los alias (SQLSTATE 42916).
- **Resolución de un nombre de alias sin calificar:** cuando se resuelve un nombre sin calificar, se tienen en cuenta primero los alias privados y luego los públicos.
- **Vinculación conservadora para los alias públicos:** si se utiliza un alias público en una sentencia de un paquete, una rutina de SQL, un activador, la expresión por omisión de una variable global o una definición de vista, estos objetos seguirán utilizando el alias público independientemente de qué otro objeto con el mismo nombre se cree a continuación.
- La creación de un alias 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos.
 - SYNONYM puede especificarse en lugar de ALIAS

Ejemplos

- *Ejemplo 1:* HEDGES intenta crear un alias para una tabla T1 (ambos sin calificar).

```
CREATE ALIAS A1 FOR T1
```

Se crea el alias HEDGES.A1 para HEDGES.T1.

- *Ejemplo 2:* HEDGES intenta crear un alias para una tabla (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

Se crea el alias HEDGES.A1 para MCKNIGHT.T1.

- *Ejemplo 3:* HEDGES intenta crear un alias para una tabla (el alias en un esquema diferente; HEDGES no es DBADM; HEDGES no tiene el privilegio CREATEIN para el esquema MCKNIGHT).

```
CREATE ALIAS MCKNIGHT.A1 FOR MCKNIGHT.T1
```

Este ejemplo falla (SQLSTATE 42501).

- *Ejemplo 4:* HEDGES intenta crear un alias para una tabla no definida (ambos calificados; FUZZY.WUZZY no existe).

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

CREATE ALIAS

Esta sentencia resulta satisfactoria pero emite un mensaje de aviso (SQLSTATE 01522).

- *Ejemplo 5:* HEDGES intenta crear un alias para un alias (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 alias, 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 alias se crea sean cuales sean los privilegios de las tablas.

- *Ejemplo 6:* Designar A1 como alias 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 mantiene determinada información en una tabla que reside en un RDBMS de DB2. 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 alias para DEPTD108 y PERSONNEL como un alias para DEPTD577.

```
CREATE ALIAS FINANCE FOR DEPTD108  
CREATE ALIAS PERSONNEL FOR DEPTD577
```

- *Ejemplo 8:* Crear un alias público llamado TABS para la vista de catálogo SYSCAT.TABLES.

```
CREATE PUBLIC ALIAS TABS FOR SYSCAT.TABLES
```

CREATE AUDIT POLICY

La sentencia CREATE AUDIT POLICY define una política de auditoría en el servidor actual. La política determina las categorías que han de auditarse; después puede aplicarse a otros objetos de base de datos para determinar el modo en que van a auditarse dichos objetos.

Invocación

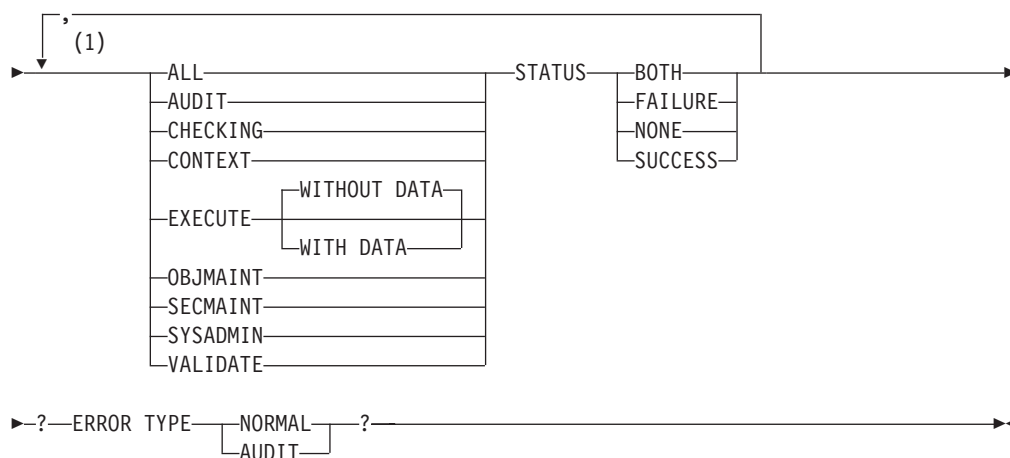
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

►► CREATE AUDIT POLICY *nombre-política* [?] CATEGORIES



Notas:

- 1 Cada categoría puede especificarse una vez como máximo (SQLSTATE 42614) y no puede especificarse ninguna otra categoría si se especifica ALL (SQLSTATE 42601).

Descripción

nombre-política

Asigna un nombre a la política de auditoría. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-política* no debe identificar una política de auditoría que ya esté descrita en el catálogo (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

CREATE AUDIT POLICY

CATEGORIES

Una lista de una o más categorías de auditoría para las que se especifica un estado. Si no se especifica ALL, el STATUS de cualquier categoría que no se especifique explícitamente se establece en NONE.

ALL

Establece todas las categorías en el mismo estado. La categoría EXECUTE es WITHOUT DATA.

AUDIT

Genera registros cuando se modifican los valores de auditoría o cuando se accede a las anotaciones cronológicas de auditoría.

CHECKING

Genera registros durante la comprobación de autorización de intentos de acceder o manipular funciones u objetos de base de datos.

CONTEXT

Genera registros para mostrar el contexto de operación cuando se realiza una operación de base de datos.

EXECUTE

Genera registros para mostrar la ejecución de sentencias de SQL.

WITHOUT DATA o WITH DATA

Especifica si se proporcionan valores de datos de entrada para las variables del lenguaje principal y deben registrarse marcadores de parámetro como parte de la categoría EXECUTE.

WITHOUT DATA

Se proporcionan valores de datos de entrada para las variables del lenguaje principal y no se registran marcadores de parámetro como parte de la categoría EXECUTE. WITHOUT DATA es el valor por omisión.

WITH DATA

Se proporcionan valores de datos de entrada para las variables del lenguaje principal y se registran marcadores de parámetro como parte de la categoría EXECUTE. No se registran todos los valores de entrada; específicamente, los parámetros de tipo estructurado, LOB, LONG y XML aparecen como valor nulo. Los campos de fecha, hora e indicación de fecha y hora se registran en formato ISO. Los valores de datos de entrada se convierten a la página de códigos de base de datos antes de registrarse. Si falla la conversión de página de códigos, no se devuelven errores y se registran los datos no convertidos.

OBJMAINT

Genera registros cuando se crean o descartan los objetos de datos.

SECMAINT

Genera registros cuando se otorgan o revocan privilegios de objeto, privilegios de base de datos o la autorización DBADM. También se generan registros cuando se modifican los parámetros de configuración de seguridad del gestor de base de datos **grupo_sysadm**, **grupo_sysctrl** o **grupo_sysmaint**.

SYSADMIN

Genera registros cuando se realizan operaciones que requieren autorización SYSADM, SYSMAINT o SYSCTRL.

VALIDATE

Genera registros cuando se autentican usuarios o cuando se recupera información de seguridad del sistema relacionada con un usuario.

STATUS

Especifica un estado para la categoría especificada.

BOTH

Se auditarán los sucesos satisfactorios y anómalos.

FAILURE

Sólo se auditarán los sucesos anómalos.

SUCCESS

Sólo se auditarán los sucesos satisfactorios.

NONE

No se auditarán los sucesos de esta categoría.

ERROR TYPE

Especifica si van a devolverse o ignorarse los errores de auditoría.

NORMAL

Los errores generados por la auditoría se ignorarán y sólo se devolverán a la aplicación los SQLCODE para los errores asociados con la operación que se está realizando.

AUDIT

Se devuelven a la aplicación todos los errores, incluyendo los errores que se producen dentro del propio recurso de auditoría.

Normas

- Una sentencia de SQL exclusiva de AUDIT debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de AUDIT son:
 - AUDIT
 - CREATE AUDIT POLICY, ALTER AUDIT POLICY o DROP (AUDIT POLICY)
 - DROP (ROLE o TRUSTED CONTEXT si está asociada a una política de auditoría)
- Una sentencia de SQL exclusiva de AUDIT no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Sólo se permite una sentencia de SQL exclusiva de AUDIT sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva de AUDIT sin confirmar, las siguientes sentencias de SQL exclusivas de AUDIT esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de AUDIT actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplo

Cree una política de auditoría para auditar éxitos y anomalías para las categorías AUDIT y OBJMAINT; sólo las anomalías para las categorías SECMAINT, CHECKING y VALIDATE y ningún suceso para las demás categorías.

CREATE AUDIT POLICY

```
CREATE AUDIT POLICY DBAUDPRF
  CATEGORIES AUDIT STATUS BOTH,
             SECMAINT STATUS FAILURE,
             OBJMAINT STATUS BOTH,
             CHECKING STATUS FAILURE,
             VALIDATE STATUS FAILURE
  ERROR TYPE NORMAL
```

CREATE BUFFERPOOL

La sentencia CREATE BUFFERPOOL define una agrupación de almacenamientos intermedios en el servidor actual. Las agrupaciones de almacenamientos intermedios se definen en miembros que pueden acceder a particiones de datos.

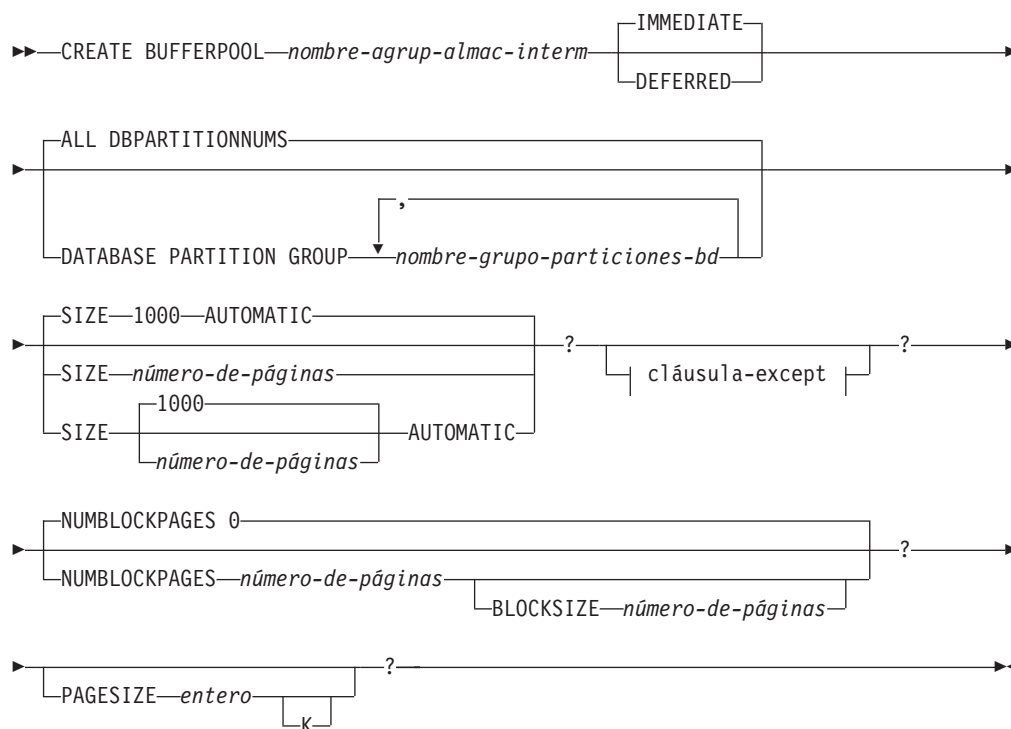
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

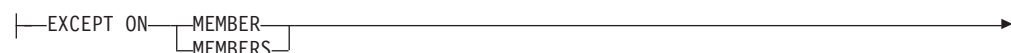
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis



cláusula-except:



CREATE BUFFERPOOL

```
▶ ( ( miembro-número1 [ TO miembro-número2 ] SIZE número-de-páginas ) )
```

Descripción

nombre-agrup-almac-interm

Indica el nombre de la agrupación de almacenamientos intermedios. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-agrup-almac-interm* no debe identificar ninguna agrupación de almacenamientos intermedios que ya exista en el catálogo (SQLSTATE 42710). El *nombre-agrup-almac-interm* no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

IMMEDIATE o DEFERRED

Indica si la agrupación de almacenamientos intermedios se creará de forma inmediata.

IMMEDIATE

La agrupación de almacenamientos intermedios se creará inmediatamente. Si no hay espacio reservado suficiente en la memoria compartida de la base de datos para asignar la nueva agrupación de almacenamientos intermedios (SQLSTATE 01657), la sentencia se ejecutará como DEFERRED.

DEFERRED

La agrupación de almacenamientos intermedios se creará cuando se desactive la base de datos (es necesario desconectar todas las aplicaciones de la base de datos). No es necesario espacio de memoria reservado; DB2 asignará la memoria necesaria del sistema.

ALL DBPARTITIONNUMS o DATABASE PARTITION GROUP

Identifica los miembros en los que se debe definir la agrupación de almacenamientos intermedios. El valor por omisión es ALL DBPARTITIONNUMS.

ALL DBPARTITIONNUMS

Esta agrupación de almacenamientos intermedios se creará en todos los miembros que puedan acceder a todas las particiones de datos de la base de datos.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd, ...*

Identifica el grupo o grupos de particiones de base de datos a los que se aplica la definición de agrupación de almacenamientos intermedios. La agrupación de almacenamientos intermedios se creará sólo en los miembros de los grupos de partición de base de datos especificados. Cada grupo de partición de base de datos debe existir en la base de datos (SQLSTATE 42704).

SIZE

Especifica el tamaño de la agrupación de almacenamientos intermedios. Este tamaño será el tamaño por omisión para todos los miembros en los que existe la agrupación de almacenamientos intermedios. El valor por omisión es 1.000 páginas.

número-de-páginas

Número de páginas de la nueva agrupación de almacenamientos intermedios. El número mínimo de páginas es 2 y el número máximo depende de la arquitectura (SQLSTATE 42615).

AUTOMATIC

Habilita el ajuste automático para la agrupación de almacenamientos intermedios. El gestor de bases de datos ajusta el tamaño de la agrupación de almacenamientos intermedios como respuesta a los requisitos de carga de trabajo. El número de páginas implícito o explícito especificado se utiliza como tamaño inicial de la agrupación de almacenamientos intermedios. En las activaciones posteriores de la base de datos, el tamaño de la agrupación de almacenamientos intermedios se basa en el último valor ajustado determinado por el gestor de memoria de autoajuste (STMM). Tenga en cuenta que STMM aplica un tamaño mínimo a las agrupaciones de almacenamientos intermedios automáticos. Para determinar el tamaño actual de las agrupaciones de almacenamientos intermedios que se han habilitado para el autoajuste, utilice el mandato **GET SNAPSHOT** y examine el tamaño actual de las agrupaciones de almacenamientos intermedios (el valor del elemento de supervisor **bp_cur_buffsz**).

NUMBLOCKPAGES *número-de-páginas*

Especifica el número de páginas que deben existir en el área basada en bloques. El número de páginas no debe superar el 98 por ciento del número de páginas para la agrupación de almacenamientos intermedios (SQLSTATE 54052). Si se especifica el valor 0, se inhabilita la E/S de bloque. El valor real de NUMBLOCKPAGES utilizado será un múltiplo de BLOCKSIZE.

No se da soporte a NUMBLOCKPAGES en un entorno DB2 pureScale (SQLSTATE 56038).

BLOCKSIZE *número-de-páginas*

Especifica el número de páginas de un bloque. El tamaño de bloque debe ser un valor comprendido entre el 2 y el 256 (SQLSTATE 54053). El valor por omisión es 32.

BLOCKSIZE no se soporta en un entorno DB2 pureScale (SQLSTATE 56038).

EXCEPT ON MEMBER o EXCEPT ON MEMBERS

Especifica el miembro o los miembros para los que el tamaño de la agrupación de almacenamientos intermedios será distinta del valor por omisión especificado para el grupo de particiones de base de datos al que el miembro tiene acceso. Si no se especifica esta cláusula, todos los miembros que puedan acceder a las particiones de datos del grupo de particiones de base de datos especificado tendrán el mismo tamaño que el especificado para esta agrupación de almacenamientos intermedios.

miembro-número1

Especifica un número de miembro para un miembro que tiene acceso a una partición de datos para la que se ha creado la agrupación de almacenamientos intermedios (SQLSTATE 42729).

TO *miembro-número2*

Especifica un rango de números de miembro. El valor de *miembro-número2* debe ser superior o igual al valor de *miembro-número1* (SQLSTATE 428A9). Cada miembro identificado por el rango de números de miembro debe tener acceso a la partición de datos para la que se ha creado la agrupación de almacenamientos intermedios (SQLSTATE 428A9).

SIZE *número-de-páginas*

El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas. El número mínimo de páginas es 2 y el número máximo depende de la arquitectura (SQLSTATE 42615).

CREATE BUFFERPOOL

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 o 32.768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio. Si el tamaño de página no es uno de estos valores, se devuelve un error (SQLSTATE 428DE).

El valor por omisión se proporciona mediante el parámetro de configuración de base de datos **tamaño-página**, que se establece cuando se crea la base de datos.

Notas

- Si la agrupación de almacenamientos intermedios se crea mediante la utilización de la opción DEFERRED, cualquier espacio de tablas que se cree en esta agrupación de almacenamientos intermedios utilizará una pequeña agrupación de almacenamientos intermedios del sistema con el mismo tamaño de página hasta la próxima vez que se active la base de datos. La base de datos deberá reiniciarse para que la agrupación de almacenamientos intermedios se active y para que entren en vigor las asignaciones de espacio de tablas para la nueva agrupación de almacenamientos intermedios. La opción por omisión es IMMEDIATE.
- 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 memoria para las agrupaciones de almacenamientos intermedios normales, intentará iniciarse con agrupaciones de almacenamientos intermedios del sistema pequeñas, una para cada tamaño de página (4 K, 8 K, 16 K y 32 K). Ante esta situación, se enviará un mensaje de aviso al usuario (SQLSTATE 01626) y las páginas de todos los espacios de tablas utilizarán las agrupaciones de almacenamientos intermedios del sistema.
- *Alternativas de sintaxis*: Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
 - Se puede especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se ha establecido en ON.
 - Se puede especificar DBPARTITIONNUMS o NODES en lugar de MEMBERS, salvo cuando se ha establecido la variable de registro DB2_ENFORCE_MEMBER_SYNTAX en ON.

CREATE DATABASE PARTITION GROUP

La sentencia CREATE DATABASE PARTITION GROUP define un grupo nuevo de particiones de base de datos, asigna particiones de base de datos en el grupo de particiones de base de datos y registra la definición del grupo de particiones de base de datos en el catálogo del sistema.

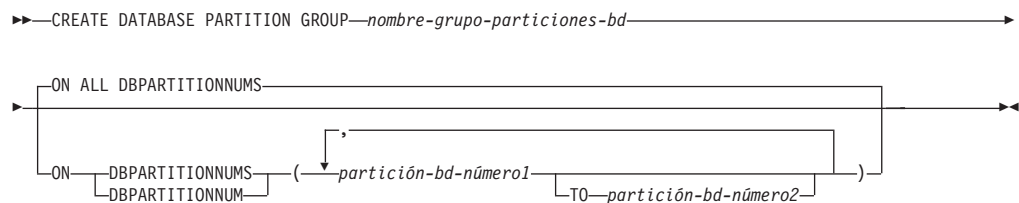
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis



Descripción

nombre-grupo-particiones-bd

Indica el nombre del grupo de particiones de base de datos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo-particiones-bd* no debe identificar un grupo de particiones de base de datos que ya exista en el catálogo (SQLSTATE 42710). El *nombre-grupo-particiones-bd* no debe empezar por los caracteres 'SYS' o 'IBM' (SQLSTATE 42939).

ON ALL DBPARTITIONNUMS

Especifica que el grupo de particiones de base de datos está definido en todas las particiones de base de datos definidas para la base de datos (archivo `db2nodes.cfg`) cuando se crea el grupo de particiones de base de datos.

Si se añade una partición de base de datos al sistema de la base de datos, debe emitirse la sentencia ALTER DATABASE PARTITION GROUP para que incluya esta nueva partición de base de datos en un grupo de particiones de base de datos (incluido IBMDEFAULTGROUP). Además, debe emitirse el mandato REDISTRIBUTE DATABASE PARTITION GROUP para mover datos a la partición de base de datos.

ON DBPARTITIONNUMS

Especifica las particiones de base de datos que se encuentran en el grupo de particiones de base de datos. DBPARTITIONNUM es sinónimo de DBPARTITIONNUMS.

CREATE DATABASE PARTITION GROUP

partición-bd-núm-1

Especifique un número de partición de base de datos. (Para mantener la compatibilidad con la versión anterior, puede especificarse un *nombre-nodo* en el formato NODEEnnnn.)

T0 *partición-bd-número2*

Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9). Todas las particiones de base de datos entre los números de partición de base de datos especificados e incluidos estos se encuentran en el grupo de particiones de base de datos.

Normas

- Cada partición de base de datos especificada por número debe estar definida en el archivo db2nodes.cfg (SQLSTATE 42729).
- Cada *número-partición-base-de-datos* de la lista de la cláusula ON DBPARTITIONNUMS deberá aparecer una vez como máximo (SQLSTATE 42728).
- Un *número-partición-base-de-datos* válido es un número comprendido entre 0 y 999, ambos inclusive (SQLSTATE 42729).
- La sentencia CREATE DATABASE PARTITION GROUP puede resultar anómala (SQLSTATE 55071) si una petición para añadir un servidor de particiones de base de datos está pendiente o en curso. Esta sentencia puede también resultar anómala (SQLSTATE 55077) si se añade en línea un servidor de particiones de base de datos nuevo a la instancia y no todas las aplicaciones saben de la existencia del servidor de particiones de base de datos nuevo.

Notas

- Esta sentencia crea una correlación de distribución para el grupo de particiones de base de datos. Para cada correlación de distribución, se genera un identificador de correlaciones de distribución (PMAP_ID). Esta información se graba en el catálogo y puede recuperarse desde SYSCAT.DBPARTITIONGROUPS y SYSCAT.PARTITIONMAPS. Cada entrada de la correlación de distribución especifica la partición de base de datos de destino en la que residen todas las filas generadas aleatoriamente. Para un grupo de particiones de base de datos de una sola partición, la correlación de distribución correspondiente sólo tiene una entrada. Para un grupo de particiones de base de datos de varias particiones, la correlación de distribución correspondiente tiene 32768 entradas, donde los números de partición de base de datos se asignan a las entradas de correlación de forma rotativa por omisión.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODE puede especificarse en lugar de DBPARTITIONNUM
 - NODES puede especificarse en lugar de DBPARTITIONNUMS
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP

Ejemplos

Los ejemplos siguientes se basan en una base de datos particionada con seis particiones de base de datos definidas como 0, 1, 2, 5, 7 y 8.

CREATE DATABASE PARTITION GROUP

- *Ejemplo 1:* supongamos que desea crear un grupo de particiones de base de datos denominado MAXGROUP en las seis particiones de base de datos. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- *Ejemplo 2:* supongamos que desea crear un grupo de particiones de base de datos denominada MEDGROUP en las particiones de base de datos 0, 1, 2, 5 y 8. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MEDGROUP  
ON DBPARTITIONNUMS ( 0 TO 2, 5, 8)
```

- *Ejemplo 3:* supongamos que desea crear un grupo de particiones de base de datos de una sola partición MINGROUP en la partición 7 de la base de datos. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MINGROUP  
ON DBPARTITIONNUM (7)
```

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.

Se pueden crear varios tipos distintos de supervisores de sucesos con esta sentencia. Siete de estos tipos se describen por separado (consulte Enlaces relacionados) y los tipos restantes se describen aquí. Los tipos de supervisores de sucesos que se describen por separado:

- **Actividades.** El supervisor de sucesos registrará sucesos de actividad que tienen lugar cuando se utiliza la base de datos. La definición del supervisor de sucesos estadísticos especifica también el lugar donde la base de datos debe registrar los sucesos.
- **Bloqueo.** El supervisor de sucesos registrará sucesos relacionados con bloqueos que tienen lugar cuando se utiliza la base de datos. Todos los registros se recopilan en la tabla de sucesos sin formato.
- **Antememoria de paquete.** El supervisor de sucesos registrará los sucesos relacionados con la sentencia de antememoria de paquete.
- **Estadísticos.** El supervisor de sucesos registrará los sucesos estadísticos que tienen lugar cuando se utiliza la base de datos. La definición del supervisor de sucesos estadísticos especifica también el lugar donde la base de datos debe registrar los sucesos.
- **Violaciones de umbral.** El supervisor de sucesos registrará sucesos de violaciones de umbral que tienen lugar cuando se utiliza la base de datos. La definición del supervisor de sucesos estadísticos especifica también el lugar donde la base de datos debe registrar los sucesos.
- **Unidad de trabajo.** El supervisor de sucesos registrará los sucesos cuando se complete una unidad de trabajo. Todos los registros se recopilan en la tabla de sucesos sin formato.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

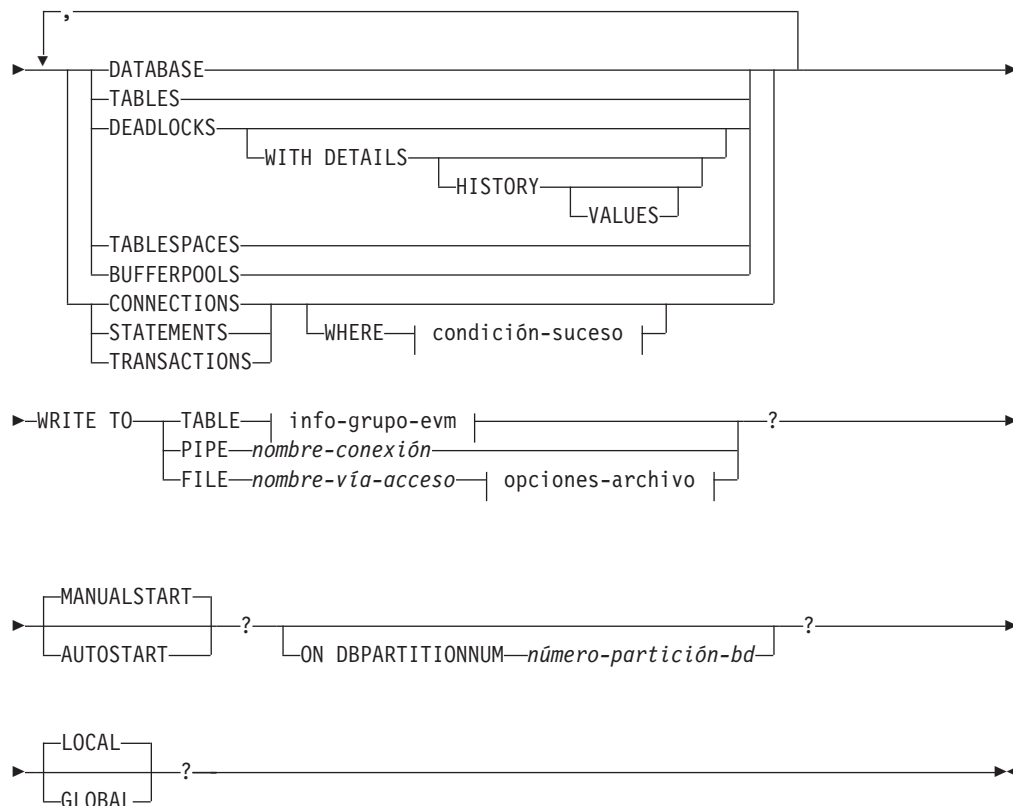
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM

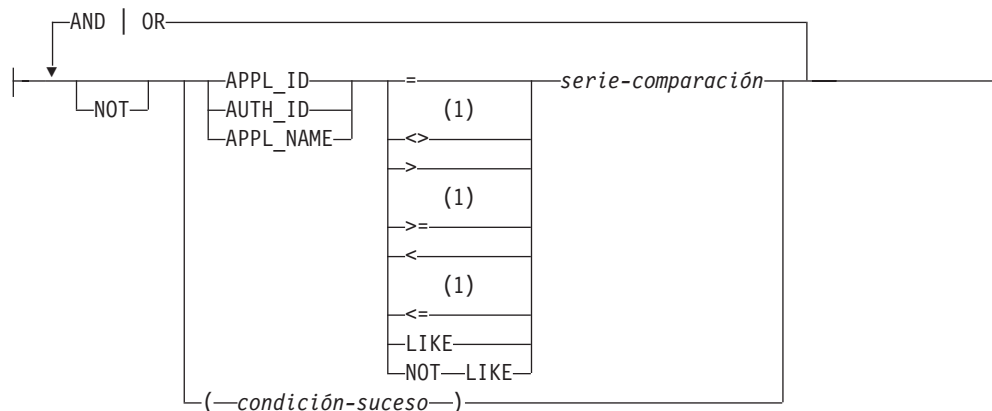
Sintaxis

►► CREATE EVENT MONITOR *nombre-supervisor-sucesos* FOR *FOR* ►►

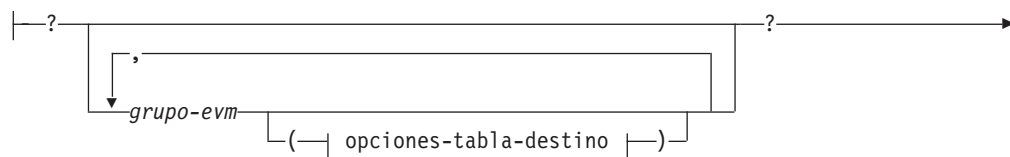
CREATE EVENT MONITOR



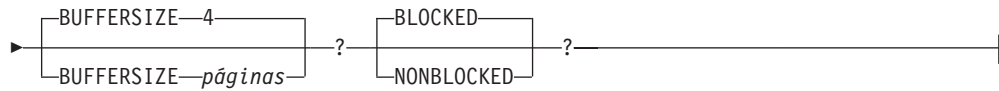
condición-suceso:



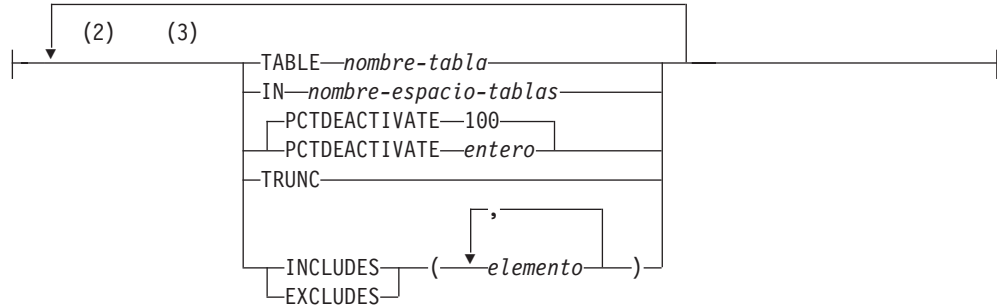
info-grupo-evm:



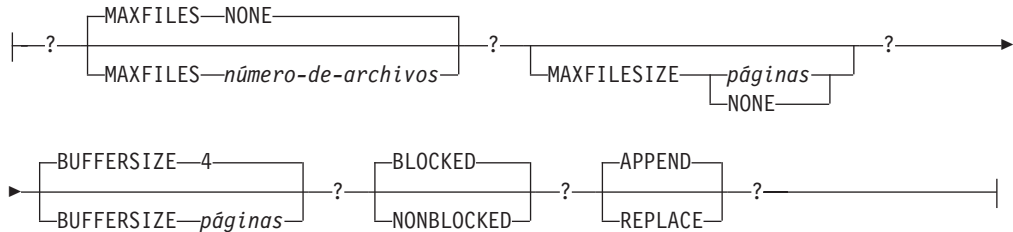
CREATE EVENT MONITOR



opciones-tabla-destino:



opciones-archivo:



Notas:

- 1 También se da soporte a otras formas de estos operadores.
- 2 Cada cláusula sólo puede especificarse una vez.
- 3 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe 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 base de datos. Para tablas particionadas, se registra un suceso de tabla para cada partición de datos de cada tabla activa. Una tabla activa es una tabla que ha cambiado desde la primera conexión con la base de datos.

DEADLOCKS

Nota: Esta opción está en desuso. No se recomienda su uso y podría eliminarse en un próximo release. Utilice la sentencia CREATE EVENT MONITOR FOR LOCKING para supervisar los sucesos relacionados con el bloqueo, como los tiempos de espera excedidos de bloqueo, las esperas de bloqueo y los puntos muertos.

Especifica que el supervisor de sucesos registre un suceso de punto muerto siempre que se produzca un punto muerto.

WITH DETAILS

Especifica que el supervisor de sucesos debe generar un suceso de conexión de punto muerto más detallado para cada aplicación que esté implicada en un punto muerto. Estos detalles adicionales incluyen:

- Información acerca de la sentencia que la aplicación estaba ejecutando cuando se ha producido el punto muerto, como, por ejemplo, el texto de la sentencia.
- Los bloqueos que mantenía la aplicación al producirse el punto muerto. En un entorno de base de datos particionada, sólo se incluyen los bloqueos mantenidos en la partición de base de datos en la que la aplicación estaba esperando su bloqueo cuando se ha producido el punto muerto. Para las tablas particionadas, ello incluye el identificador de la partición de datos.

HISTORY

Especifica que los datos del supervisor de sucesos también incluirán:

- El histórico de todas las sentencias de la unidad de trabajo actual en el nodo participante (incluidos los cursores WITH HOLD abiertos en unidades de trabajo anteriores). Las sentencias SELECT emitidas en el nivel de aislamiento UR (lectura no confirmada) no se incluyen en el histórico de sentencias.
- El entorno de compilación de sentencias para cada sentencia de SQL en formato binario (si está disponible).

VALUES

Especifica que los datos del supervisor de sucesos también incluirán:

- Los valores de datos utilizados como variables de entrada para cada sentencia de SQL. Estos valores no incluirán datos LOB, datos largos, datos de tipo estructurado ni datos XML.

En una única sentencia CREATE EVENT MONITOR, puede especificar sólo uno de estos elementos: DEADLOCKS, DEADLOCKS WITH DETAILS, DEADLOCKS WITH DETAILS HISTORY o DEADLOCKS WITH DETAILS HISTORY VALUES (SQLSTATE 42613).

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.

CREATE EVENT MONITOR

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

Nota: Esta opción está en desuso. No se recomienda su uso y podría eliminarse en un próximo release. Utilice la sentencia CREATE EVENT MONITOR FOR UNIT OF WORK para supervisar los sucesos de transacción.

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-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:

- Para cada conexión activa cuando se activa un supervisor de sucesos por primera vez
- Posteriormente para 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 ha especificado ninguna cláusula WHERE, se supervisarán todos los sucesos del tipo de suceso especificado.

La condición-suceso no debe exceder una longitud de 32.678 bytes en la página de códigos de base de datos (SQLSTATE 22001).

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 o 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

Es una serie de caracteres que debe compararse con el APPL_ID, AUTH_ID o APPL_NAME de cada aplicación que se conecta con 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

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-grupo-evm

Define la tabla de destino de un grupo de datos lógicos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula info-grupo-evm, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-evm

Identifica el grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tabla 16. Valores para grupo-evm basados en el tipo de supervisor de sucesos

Tipo de supervisor de sucesos	Valor grupo-evm
Base de datos	<ul style="list-style-type: none"> • DB • CONTROL¹ • DBMEMUSE
Tablas	<ul style="list-style-type: none"> • TABLE • CONTROL¹
Puntos muertos	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN • CONTROL¹
Puntos muertos con detalles	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • CONTROL¹

CREATE EVENT MONITOR

Tabla 16. Valores para grupo-evm basados en el tipo de supervisor de sucesos (continuación)

Tipo de supervisor de sucesos	Valor grupo-evm
Puntos muertos con detalles e histórico	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • STMTHIST • CONTROL¹
Puntos muertos con detalles, histórico y valores	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • STMTHIST • STMTVALS • CONTROL¹
Espacios de tablas	<ul style="list-style-type: none"> • TABLESPACE • CONTROL¹
Agrupaciones de almacenamientos intermedios	<ul style="list-style-type: none"> • BUFFERPOOL • CONTROL¹
Conexiones	<ul style="list-style-type: none"> • CONNHEADER • CONN • CONTROL¹ • CONNMEMUSE
Sentencias	<ul style="list-style-type: none"> • CONNHEADER • STMT • SUBSECTION⁴ • CONTROL¹
Transacciones	<ul style="list-style-type: none"> • CONNHEADER • XACT • CONTROL¹
Actividades	<ul style="list-style-type: none"> • ACTIVITY • ACTIVITYMETRICS • ACTIVITYSTMT • ACTIVITYVALS • CONTROL¹
Estadísticas	<ul style="list-style-type: none"> • QSTATS • SCSTATS • SCMETRICS • WCSTATS • WLSTATS • WLMETRICS • HISTOGRAMBIN • CONTROL¹

Tabla 16. Valores para grupo-*evm* basados en el tipo de supervisor de sucesos (continuación)

Tipo de supervisor de sucesos	Valor grupo- <i>evm</i>
Violaciones de umbral	<ul style="list-style-type: none"> • THRESHOLDVIOLATIONS • CONTROL¹
Bloqueo ⁵	<ul style="list-style-type: none"> • LOCK • LOCK_PARTICIPANTS • LOCK_PARTICIPANT_ACTIVITIES • LOCK_ACTIVITY_VALUES • CONTROL¹
Antememoria de paquetes ⁵	<ul style="list-style-type: none"> • PKGCACHE • PKGCACHE_METRICS • CONTROL¹
Unidad de trabajo ⁵	<ul style="list-style-type: none"> • UOW • UOW_METRICS • UOW_PACKGE_LIST • UOW_EXECUTABLE_LIST • CONTROL¹
Historial de cambios	<ul style="list-style-type: none"> • CHANGESUMMARY • EVMONSTART • TXNCOMPLETION • DDLSTMTEXEC • DBDBMCFG • REGVAR • UTILSTART • UTILSTOP • UTILPHASE • UTILLOCATION • CONTROL¹

¹ La cabecera de base de datos de los grupos de datos lógicos (sólo el elemento *conn_time*), el inicio y el desbordamiento se graban en el grupo CONTROL. El desbordamiento se graba si el supervisor de sucesos está en estado no bloqueado y se han descartado los sucesos.

² Corresponde al suceso DETAILED_DLCONN.

³ Corresponde a los grupos de datos lógicos LOCK que se producen dentro de cada suceso DETAILED_DLCONN.

⁴ Sólo se crea para los entornos de bases de datos particionadas.

⁵ Se refiere a la versión de la tabla de sucesos con formato de este tipo de supervisor de sucesos.

opciones-tabla-destino

Identifica la tabla de destino de un grupo. Si no se especifica un valor para *opciones-tabla-destino*, el proceso CREATE EVENT MONITOR continúa como se indica a continuación:

CREATE EVENT MONITOR

- Se utiliza un nombre de tabla derivada (consulte la descripción de TABLE *nombre-tabla*).
- Se elige un espacio de tablas por omisión (consulte la descripción de IN *nombre-espacio-tablas*).
- Se incluyen todos los elementos.
- PCTDEACTIVATE y TRUNC no se especifican.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'  
CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

Se recomienda que, cuando un espacio de tablas de destino tenga el redimensionamiento automático habilitado, el parámetro PCTDEACTIVATE se establezca en 100.

TRUNC

Especifica que las columnas STMT_TEXT y STMT_VALUE_DATA están definidas como VARCHAR(*n*), donde *n* es el tamaño mayor que puede caber en la fila de la tabla. En este caso, los datos que tengan más de *n* bytes

se truncarán. En el ejemplo siguiente se muestra cómo se calcula el valor de *n*. Supongamos que:

- La tabla se ha creado en un espacio de tablas que utiliza páginas de 32K.
- La longitud total de todas las demás columnas de la tabla es de 357 bytes.

En este caso, el tamaño de fila máximo para la tabla es de 32677 bytes. Por lo tanto, el elemento se definirá como VARCHAR(32316); es decir, 32677 - 357 - 4. Si no se especifica TRUNC, la columna se definirá como CLOB(2M). Tenga en cuenta que STMT_TEXT se encuentra en el grupo de sucesos STMT, en el grupo de sucesos STMT_HISTORY y en el grupo de sucesos DLCONN (para los supervisores de sucesos de puntos muertos con detalles). STMT_VALUE_DATA se encuentra en el grupo de sucesos DATA_VALUE.

INCLUDES

Especifica que los elementos siguientes van a incluirse en la tabla.

EXCLUDES

Especifica que los elementos siguientes *no* van a incluirse en la tabla.

elemento

Identifica un elemento del supervisor. La información de los elementos puede proporcionarse de una de las formas siguientes:

- No especificar información de los elementos. En este caso, todos los elementos se incluyen en la sentencia CREATE TABLE.
- Especificar los elementos que deben incluirse en el formato: INCLUDES (elemento1, elemento2, ..., elemento*n*). Para estos elementos, sólo se crean columnas de tabla.
- Especificar los elementos que deben excluirse en el formato: EXCLUDES (elemento1, elemento2, ..., elemento*n*). Sólo se crean columnas de tabla para todos los elementos que no se han indicado aquí.

Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.

BUFFERSIZE *páginas*

Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Los supervisores de sucesos de tabla insertan todos los datos de un almacenamiento intermedio y emiten COMMIT cuando se ha procesado el almacenamiento intermedio. Cuanto más grandes sean los almacenamientos intermedios, mayor será el ámbito de confirmación utilizado por 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 un supervisor, se asignan dos almacenamientos intermedios del tamaño especificado.

CREATE EVENT MONITOR

Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esa pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

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.

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 normas 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, están permitidos los nombres de conexión relativos y se tratan como nombres de vía de acceso relativos (consulte la descripción de *nombre-vía-acceso*). En Windows, no obstante, existe una sintaxis especial para un nombre de conexión y, como consecuencia de ello, se necesitan nombres de conexión 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 de SQL, dicha sentencia fallará (SQLSTATE 58030).

En un entorno DB2 pureScale, *nombre-conexión* debe estar en un sistema de archivos compartido tanto si se trata de un supervisor de sucesos LOCAL o GLOBAL. Este requisito es para permitir que estos supervisores de sucesos funcionen correctamente en el caso de una migración tras error de un miembro. Si no se utiliza un *nombre-conexión* en un sistema de archivos compartidos, se generará un error (SQLSTATE 428A3) si el supervisor de sucesos se activa durante una migración tras error de miembro.

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-acceso

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; no obstante, la vía de acceso propiamente dicha puede residir en otra partición de base de datos (por ejemplo, en un sistema 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 por el directorio raíz en AIX, o por un identificador de disco en Windows), se utilizará la vía de acceso especificada. En entornos que no sean DB2 pureScale, si se especifica una vía de acceso relativa (vía de acceso que no empieza por la raíz), se utilizará la vía de acceso relativa al directorio DB2EVENT del directorio de base de datos. En un entorno DB2 pureScale, si se especifica una vía de acceso relativa, se utilizará la vía de acceso relativa a la base de datos propietaria del directorio dentro del directorio de base de datos.

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.

En un entorno DB2 pureScale, *nombre-vía-acceso* debe estar en un sistema de archivos compartido tanto si se trata de un supervisor de sucesos LOCAL o GLOBAL. Este requisito es para permitir que estos supervisores de sucesos funcionen correctamente en el caso de una migración tras error de un miembro. Si no se utiliza un *nombre-vía-acceso* en un sistema de archivos compartidos, se generará un error (SQLSTATE 428A3) si el supervisor de sucesos se activa durante una migración tras error de miembro.

opciones-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. 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:

- Windows - 200 páginas de 4 K
- 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 por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el valor del parámetro MAXFILESIZE, así como por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esta pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

Los supervisores de sucesos que graban sus datos en una conexión ("pipe") también tienen dos almacenamientos intermedios internos (no configurables) que tienen un tamaño de 1 página cada uno. Estos almacenamientos intermedios también se asignan a partir de la pila del supervisor (MON_HEAP). Para cada supervisor de sucesos activo que tiene un destino de conexión, aumente el tamaño de la pila de la base de datos en 2 páginas.

BLOCKED

Especifica que cada agente que genera un suceso debe esperar a que se grave 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 grave 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, este añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se reactiva, reanudará la grabación en los

CREATE EVENT MONITOR

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, este borrará todos los archivos de sucesos y empezará a grabar los datos en el archivo 00000000.evt.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia. Es el valor por omisión.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada.

ON DBPARTITIONNUM *núm-partición-bd*

Especifica la partición de base de datos (en un entorno de base de datos particionada) o el miembro (en un entorno DB2 pureScale) en el que se debe ejecutar un supervisor de sucesos de archivo o conexión. Cuando el ámbito de supervisión se define como LOCAL, los datos se recopilan sólo en la partición o el miembro especificado. Cuando el ámbito de supervisión se define como GLOBAL, todas las particiones de base de datos o los miembros recopilan datos y los notifican a la partición de base de datos o al miembro que tiene el número especificado. El componente de E/S se ejecutará físicamente en la partición de base de datos o el miembro especificado y grabará los registros en el archivo o la conexión que se ha especificado. Cuando DB2 pureScale está habilitado, se puede especificar -1, que permite que el componente de E/S se ejecute desde cualquier miembro activo. Además, en el caso de que el componente de E/S ya no pueda ejecutarse en un miembro determinado, el supervisor de sucesos se reiniciará con el componente de E/S ejecutándose en otro miembro activo disponible.

Esta cláusula no es válida para supervisores de sucesos de tabla. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se ejecutarán y grabarán los sucesos en todas las particiones de base de datos en las que se hayan definido espacios de tablas para tablas de destino. En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla registrarán los sucesos de todos los miembros activos.

Si no se especifica esta cláusula y DB2 pureScale no está habilitado, se utiliza el número de partición de base de datos conectado actualmente (para la aplicación). Si no se especifica esta cláusula y DB2 pureScale está habilitado, el componente de E/S puede ejecutarse en cualquier número de partición de base de datos conectada actualmente.

LOCAL

El supervisor de sucesos sólo informa acerca de la partición de base de datos que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Es el valor por omisión.

Esta cláusula es válida para los supervisores de conexión o de archivos. No es válida para supervisores de sucesos de tabla.

GLOBAL

El supervisor de sucesos informa acerca de todas las particiones de base de datos. Para una base de datos particionada, sólo los supervisores de sucesos DEADLOCKS pueden definirse como GLOBAL.

Esta cláusula es válida para los supervisores de conexión o de archivos. No es válida para supervisores de sucesos de tabla.

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. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Cuando se utiliza DEADLOCKS WITH DETAILS en lugar de DEADLOCKS, ello afecta al rendimiento. Cuando se produce un punto muerto, el gestor de bases de datos necesita más tiempo para grabar la información adicional del punto muerto.
- Normalmente, siempre que se establece una conexión se graba un suceso CONNHEADER. Sin embargo, si se ha creado un supervisor de sucesos sólo para DEADLOCKS WITH, sólo se grabará un suceso CONNHEADER la primera vez que la conexión participe en un punto muerto.
- En una base de datos con múltiples particiones de base de datos, la cláusula ON DBPARTITIONNUM puede utilizarse con los supervisores de sucesos FILE y PIPE que tengan un tipo de suceso DEADLOCKS para indicar dónde debe residir el supervisor de sucesos propiamente dicho; la información de las otras particiones de base de datos, si es relevante, se enviará a aquella ubicación para el proceso.
- En una base de datos con múltiples particiones de base de datos, un supervisor de sucesos de punto muerto recibirá información sobre las aplicaciones que tengan bloqueos que participen en el punto muerto desde todas las particiones de base de datos en las que existían dichos bloqueos participantes. Si la partición de base de datos a la que está conectada la aplicación (la partición coordinadora de la aplicación) no es una de las particiones de base de datos participantes, no se recibirá información sobre ningún suceso de punto muerto desde esa partición de base de datos.
- El parámetro BUFFERSIZE restringe el tamaño de los sucesos STMT, STMT_HISTORY, DATA_VALUE y DETAILED_DLCONN. Si un suceso STMT o STMT_HISTORY no cabe en un almacenamiento intermedio, se trunca mediante el truncamiento del texto de la sentencia. Si un suceso DETAILED_DLCONN no puede caber dentro de un almacenamiento intermedio, este se trunca eliminando bloqueos. Si todavía no tiene cabida, el texto de la sentencia se trunca. Si un suceso DATA_VAL no cabe en un almacenamiento intermedio, se trunca el valor de datos.

Los supervisores de sucesos WITH DETAILS HISTORY VALUES (y, en menor grado, WITH DETAILS HISTORY) utilizan una cantidad significativa de espacio de pila de supervisor para efectuar el seguimiento de las sentencias y sus valores

CREATE EVENT MONITOR

de datos. Si desea más información, consulte la descripción del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

- Si la partición de base de datos en la que se debe ejecutar el supervisor de sucesos no está activa, la activación de este se produce la próxima vez que se activa la partición de base de datos.
- Una vez activado un supervisor de sucesos, este se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva una partición de base de datos y esta se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- *Supervisores de sucesos de grabación en tabla:* Notas generales:
 - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
 - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
 - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.
 - Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
 - Deberán existir espacios de tablas para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tablas.
 - Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada partición de base de datos de la instancia, y cada uno de estos procesos sólo informa datos a la partición de base de datos en la que está ejecutándose.
 - Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
 - LOG_STREAM_HEADER
 - LOG_HEADER
 - DB_HEADER (Los elementos db_name y db_path no se graban. El elemento conn_time se graba en CONTROL.)
 - En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base de datos con el fin de supervisarlas,

creando un espacio de tablas que sólo exista en determinadas particiones de base de datos. En un entorno DB2 pureScale, los datos se grabarán desde cada miembro.

En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.

- Los usuarios pueden podar manualmente todas las tablas de destino.

Columnas de tabla:

- Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. Cualquier elemento del supervisor de sucesos que no tenga una columna de tabla de destino correspondiente se pasa por alto.
- Utilice el mandato `db2evtbl` para crear una sentencia `CREATE EVENT MONITOR` que incluya una lista completa de elementos para un grupo.
- Los tipos de columnas que se utilizan para los elementos del supervisor corresponden a la correlación siguiente:

SQLM_TYPE_STRING	CHAR[n], VARCHAR[n] o CLOB(n) (Si los datos del registro del supervisor de sucesos exceden de <i>n</i> bytes, se truncan.)
SQLM_TYPE_U8BIT y SQLM_TYPE_8BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_16BIT y SQLM_TYPE_U16BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_32BIT y SQLM_TYPE_U32BIT	INTEGER o BIGINT
SQLM_TYPE_U64BIT y SQLM_TYPE_64BIT	BIGINT
sqlm_timestamp	TIMESTAMP
sqlm_time(tiempo transcurrido)	BIGINT
sqlca:	
sqlerrmc	VARCHAR[72]
sqlstate	CHAR[5]
sqlwarn	CHAR[11]
otros campos	INTEGER o BIGINT

- Las columnas se han definido para que sean NOT NULL.
- Dado que el rendimiento de las tablas con columnas CLOB es inferior al de las tablas que tienen columnas VARCHAR, considere la posibilidad de utilizar la palabra clave `TRUNC` al especificar el valor *grupo-evm* de `STMT` (o el valor *grupo-evm* de `DLCONN`, si se utiliza el tipo de suceso `DEADLOCKS WITH DETAILS`).
- A diferencia de otras tablas de destino, las columnas de la tabla `CONTROL` no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

Nombre columna	Tipo datos	Nulos	Descripción
-----	-----	-----	-----
PARTITION_KEY	INTEGER	N	Clave de distribución (sólo base de datos particionada)
PARTITION_NUMBER	INTEGER	N	Número de partición de base de datos (sólo base de datos particionada)
EVMONNAME	VARCHAR(128)	N	Nombre de supervisor de sucesos
MESSAGE	VARCHAR(128)	N	Describe la naturaleza de la columna MESSAGE_TIME. Puede ser uno de los valores siguiente: - FIRST_CONNECT (la hora de la primera conexión con la base de datos tras la

CREATE EVENT MONITOR

- activación)
 - EVMON_START (la hora en que se inició el supervisor de sucesos listado en EVMONNAME)
 - OVERFLOWS:*n* (indica que se han descartado *n* registros por desbordamiento del almacenamiento intermedio)
 - LAST_DROPPED_RECORD (última vez en que se ha producido un desbordamiento)
- MESSAGE_TIME TIMESTAMP N Indicación de fecha y hora
- En un entorno de base de datos particionada, la primera columna de cada tabla se denomina PARTITION_KEY, es NOT NULL y es de tipo INTEGER. Esta columna se utiliza como clave de distribución para la tabla. El valor de esta columna se elige de forma que cada proceso del supervisor de sucesos inserte datos en la partición de base de datos en la que está ejecutándose el proceso; es decir, se realizan operaciones de inserción localmente en la partición de base de datos en la que está ejecutándose el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se descarta una partición de base de datos y se realiza la redistribución de los datos, todos los datos de la partición de base de datos que se ha descartado se dirigirán a otra partición de base de datos en lugar de distribuirse equitativamente. Por lo tanto, antes de eliminar una partición de base de datos, considere la supresión de todas las filas de tabla de esa partición de base de datos.
 - En un entorno de base de datos particionada, puede definirse una columna denominada PARTITION_NUMBER para cada tabla. Esta columna es NOT NULL y es de tipo INTEGER. Contiene el número de la partición de base de datos en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de base de datos no particionada.

Atributos de tabla:

- Se utilizan atributos de tabla por omisión. Además de la clave de distribución (sólo bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
- En la tabla puede crearse índices.
- Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
- Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.

Activación del supervisor de sucesos:

- Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.
- En un entorno de base de datos particionada, el proceso de activación se produce en cada partición de base de datos de la instancia. En una partición de base de datos en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en una partición de base de datos si existe, como mínimo, una tabla de destino en esa partición de base de datos. Además, si no se encuentra alguna tabla de destino en una partición de

base de datos, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.

- Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de base de datos particionada, si el espacio de tablas no reside en una partición de base de datos), la activación continúa y los datos que, de otro modo, se insertarían en esta tabla, se pasan por alto.
- El proceso de activación valida cada tabla de destino. Si la validación no se realiza satisfactoriamente, la activación del supervisor de sucesos no tiene lugar y se graban mensajes en el archivo de anotaciones cronológicas de administración.
- Durante la activación en un entorno de base de datos particionada, las filas de la tabla CONTROL de FIRST_CONNECT y de EVMON_START sólo se insertan en la partición de base de datos de catálogo. Para ello es necesario que el espacio de tablas de la tabla CONTROL exista en la partición de base de datos de catálogo. Si no existe en la partición de base de datos de catálogo, dichas inserciones no se realizan.
- En un entorno de base de datos particionada, si una partición todavía no está activa al activarse el supervisor de sucesos de grabación en tabla, el supervisor de sucesos se activará la próxima vez que se active la partición.

Tiempo de ejecución:

- El supervisor de sucesos se ejecuta con autorización DATAACCESS.
- Si, mientras existe un supervisor de sucesos activo, no se ejecuta satisfactoriamente una operación de inserción en una tabla de destino:
 - Los cambios no confirmados se retrotraen.
 - Se graba un mensaje en el archivo de anotaciones cronológicas de administración.
 - El supervisor de sucesos se desactiva.
- Si existe un supervisor de sucesos activo, ejecuta COMMIT localmente cuando ha terminado de procesar un almacenamiento intermedio del supervisor de sucesos.
- En un entorno de base de datos particionada, el texto de la sentencia real, que puede tener una longitud de hasta 2 MB, sólo lo almacena (en la tabla STMT o DLCONN) el proceso de supervisor de sucesos que se ejecuta en la partición de base de datos coordinadora de la aplicación. En otras particiones de base de datos, este valor tiene una longitud cero.
- En un entorno que no sea una base de datos particionada o una base de datos DB2 pureScale, todos los supervisores de sucesos de grabación en tabla se desactivarán cuando finalice la última aplicación (y la base de datos no se haya activado de forma explícita). En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla de un miembro determinado cuando se desactiva la base de datos de ese miembro y se vuelve a activar cuando la base de datos se vuelve a activar en ese miembro. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se desactivan al desactivarse la partición de catálogo.
- La sentencia DROP EVENT MONITOR no descarta tablas de destino.
- Siempre que se active un supervisor de sucesos de grabación en tabla, obtendrá bloqueos de tabla IN en cada tabla de destino para impedir que se modifiquen mientras el supervisor de sucesos esté activo. Los bloqueos de tabla se mantienen en todas las tablas mientras el supervisor de sucesos está activo. Si en cualquiera de estas tablas de destino es necesario un acceso

CREATE EVENT MONITOR

exclusivo (por ejemplo, cuando se va a ejecutar un programa de utilidad), primero desactive el supervisor de sucesos para liberar los bloqueos de tabla antes de intentar dicho acceso.

- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODE puede especificarse en lugar de DBPARTITIONNUM
 - Las comas se pueden utilizar para separar varias opciones en la cláusula *opciones-tabla-destino*.

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 debe tener 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 ejemplo siguiente 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 DB_APPLS
FOR CONNECTIONS
WRITE TO PIPE '/home/jsmith/applpipe'
```

- *Ejemplo 4:* Este ejemplo, en el que se da por supuesto un entorno de base de datos particionada, crea un supervisor de sucesos denominado FOO. Este supervisor de sucesos recopila los sucesos de la sentencia de SQL y los graba en tablas de SQL con los siguientes nombres que ha obtenido:
 - CONNHEADER_FOO
 - STMT_FOO
 - SUBSECTION_FOO
 - CONTROL_FOO

Puesto que no se ha proporcionado información de espacio de tablas, todas las tablas se crearán en un espacio de tablas seleccionado por el sistema, basándose en las normas que se describen en la cláusula *IN nombre-espacio-tablas*. Todas las tablas incluyen todos los elementos de su grupo (es decir, se definen columnas cuyos nombres son equivalentes a los nombres de los elementos.)

```
CREATE EVENT MONITOR F00
FOR STATEMENTS
WRITE TO TABLE
```

- *Ejemplo 5:* En este ejemplo, en el que se da por supuesto un entorno de base de datos particionada, se crea un supervisor de sucesos denominado BAR. Este supervisor de sucesos recopila la sentencia de SQL y los sucesos de la transacción y los graba en tablas, como se indica a continuación:
 - Los datos del grupo STMT se graban en la tabla MYDEPT.MYSTMTINFO. La tabla se crea en el espacio de tablas MYTABLESPACE. Cree columnas sólo para los elementos siguientes: ROWS_READ, ROWS_WRITTEN y STMT_TEXT. Los demás elementos del grupo se descartarán.
 - Los datos del grupo SUBSECTION se graban en la tabla MYDEPT.MYSUBSECTIONINFO. La tabla se crea en el espacio de tablas MYTABLESPACE. La tabla incluye todas las columnas, excepto START_TIME, STOP_TIME y PARTIAL_RECORD.
 - Los datos del grupo XACT se graban en la tabla XACT_BAR. Puesto que no se ha proporcionado información de espacio de tablas, la tabla se creará en un espacio de tablas seleccionado por el sistema, basándose en las normas descritas en la cláusula *IN nombre-espacio-tablas*. Esta tabla incluye todos los elementos que están contenidos en el grupo XACT.
 - No se crea ninguna tabla para connheader o control; todos los datos de esos grupos se descartan.

```
CREATE EVENT MONITOR BAR
FOR STATEMENTS, TRANSACTIONS
WRITE TO TABLE
STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
XACT
```

CREATE EVENT MONITOR (actividades)

La sentencia CREATE EVENT MONITOR (actividades) define un supervisor que registrará sucesos de actividad que se produzcan cuando se utilice la base de datos. La definición del supervisor de sucesos de actividad especifica también el lugar donde la base de datos debe registrar los sucesos.

Invocación

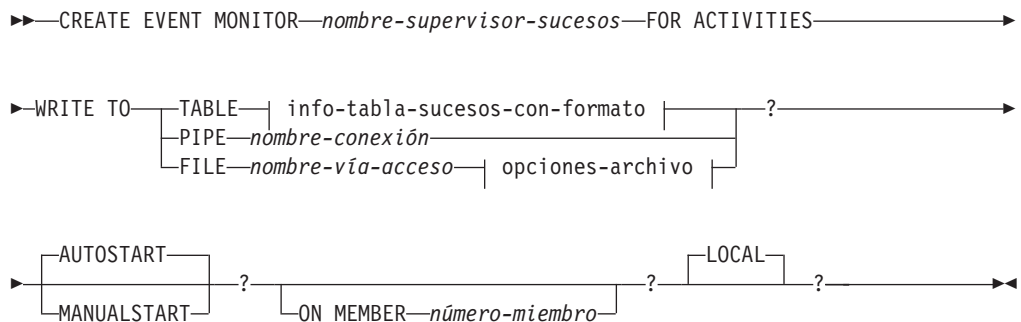
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

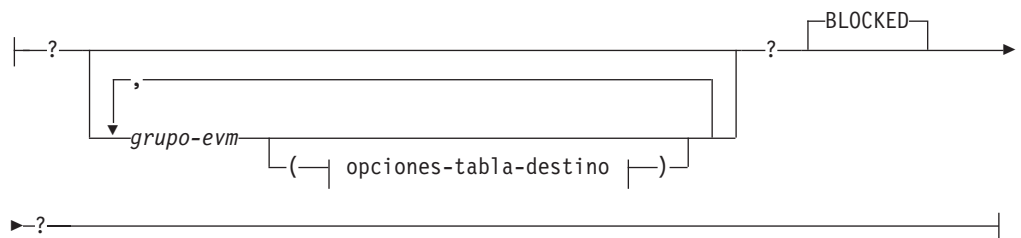
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM
- Autorización WLMADM

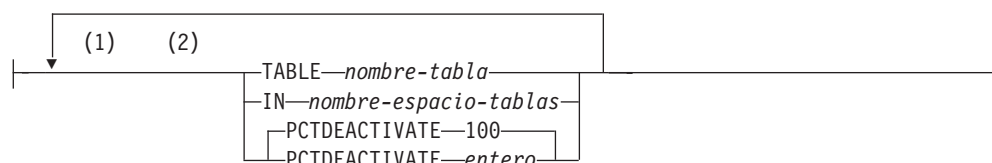
Sintaxis



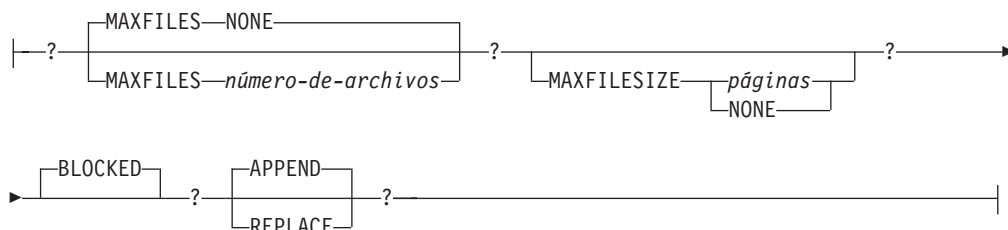
info-tabla-sucesos-con-formato:



opciones-tabla-destino:



opciones-archivo:



Notas:

- 1 Cada cláusula sólo puede especificarse una vez.
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

ACTIVITIES

Especifica que el supervisor de sucesos registra un suceso de actividad cuando una actividad termina de ejecutarse o antes de completarse la ejecución si el procedimiento `CAPTURE_ACTIVITY_IN_PROGRESS` activa el suceso. La actividad debe:

- Pertener a una clase de servicio o carga de trabajo que tenga el conjunto `COLLECT ACTIVITY DATA`, o
- Pertener a una clase de trabajo cuya acción de trabajo asociada sea `COLLECT ACTIVITY DATA`, o
- Identificarse como la actividad que ha violado el umbral cuya cláusula `COLLECT ACTIVITY DATA` estaba especificada, o
- Haberse identificado en una llamada al procedimiento `CAPTURE_ACTIVITY_IN_PROGRESS` antes de completarse

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una

CREATE EVENT MONITOR (actividades)

tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-tabla-sucesos-con-formato

Define las tablas de destino para un supervisor de sucesos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula *info-grupo-evm*, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-evm

Identifica el grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-evm
Actividades	<ul style="list-style-type: none">• ACTIVITY• ACTIVITYMETRICS• ACTIVITYSTMT• ACTIVITYVALS• CONTROL

opciones-tabla-destino

Identifica la tabla de destino de un grupo.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'  
CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

Puesto que el tamaño de la página afecta a las longitudes de INLINE LOB utilizadas, considere la posibilidad de especificar un espacio de tablas con un tamaño de página lo más grande posible para mejorar el rendimiento de INSERT del supervisor de sucesos.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE

CREATE EVENT MONITOR (actividades)

especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

Si no se especifica un valor para *opciones-tabla-destino*, el proceso CREATE EVENT MONITOR continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado.
- Se elige un espacio de tablas por omisión.
- El parámetro PCTDEACTIVATE se establece por omisión en 100.

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.

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 normas 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, están permitidos los nombres de conexión relativos y se tratan como nombres-vía relativos (consulte la descripción de *nombre-vía*). En Windows, no obstante, existe una sintaxis especial para un nombre de conexión y, como consecuencia de ello, se necesitan nombres de conexión absolutos.

CREATE EVENT MONITOR (actividades)

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-acceso

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; no obstante, la vía de acceso propiamente dicha puede residir en otra partición de base de datos (por ejemplo, en un sistema 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 por el directorio raíz en AIX, o por un identificador de disco en Windows), se utilizará la vía de acceso especificada. En entornos que no sean DB2 pureScale, si se especifica una vía de acceso relativa (vía de acceso que no empieza por la raíz), se utilizará la vía de acceso relativa al directorio DB2EVENT del directorio de base de datos. En un entorno DB2 pureScale, si se especifica una vía de acceso relativa, se utilizará la vía de acceso relativa a la base de datos propietaria del directorio dentro del directorio de base de datos.

Es posible especificar dos o más supervisores de sucesos que tengan la misma vía de acceso de destino. Sin embargo, cuando uno de los

CREATE EVENT MONITOR (actividades)

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-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. 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:

- Windows - 200 páginas de 4 K
- 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.

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.

CREATE EVENT MONITOR (actividades)

APPEND

Especifica que si ya existen archivos de datos de sucesos cuando se activa el supervisor de sucesos, este añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se reactiva, reanudará la grabación 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 debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Es el comportamiento por omisión del supervisor de sucesos de actividades.

ON MEMBER *número-miembro*

Especifica el miembro en el que se debe ejecutar un supervisor de sucesos de archivo o conexión. Cuando el ámbito de supervisión se define como LOCAL, los datos se recopilan sólo en el miembro especificado. El componente de E/S se ejecutará físicamente en el miembro especificado y grabará los registros en el archivo o la conexión especificada. Cuando se habilita la función DB2 pureScale, el valor por omisión es -1. Si se especifica un valor -1, permite que el componente de E/S se ejecute desde cualquier miembro activo. Además, en el caso de que el componente de E/S ya no pueda ejecutarse en un miembro determinado, el supervisor de sucesos se reiniciará con el componente de E/S ejecutándose en otro miembro activo disponible.

Esta cláusula no es válida para supervisores de sucesos de tabla. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se ejecutarán y grabarán los sucesos en todas las particiones de base de datos en las que se hayan definido espacios de tablas para tablas de destino. En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla registrarán los sucesos de todos los miembros activos.

Si no se especifica esta cláusula y DB2 pureScale no está habilitado, se utiliza el miembro conectado actualmente (para la aplicación). Si no se especifica esta cláusula y DB2 pureScale está habilitado, el componente de E/S puede ejecutarse en cualquier miembro conectado actualmente.

LOCAL

El supervisor de sucesos sólo informa acerca del miembro que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Es el valor por omisión.

CREATE EVENT MONITOR (actividades)

Esta cláusula es válida para los supervisores de conexión o de archivos. No es válida para supervisores de sucesos de tabla.

GLOBAL no es un ámbito válido para este tipo de supervisor de sucesos.

Normas

- El tipo de suceso ACTIVITIES no puede combinarse con ningún otro tipo de suceso de una definición de supervisor de sucesos concreta.

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. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Si el miembro en el que se debe ejecutar el supervisor de sucesos no está activo, la activación del supervisor de sucesos tiene lugar cuando se vuelve a activar ese miembro.
- Una vez activado un supervisor de sucesos, este se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva un miembro y ese miembro se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- La sentencia FLUSH EVENT MONITOR no es aplicable a este supervisor de sucesos y no se aplicará cuando se emita contra el mismo.
- *Supervisores de sucesos de grabación en tabla:* Notas generales:
 - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
 - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
 - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.
 - Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
 - Deberán existir espacios de tablas para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tablas.
 - Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada miembro de base de datos de la instancia, y cada uno de estos procesos sólo notifica datos al miembro en el que está ejecutándose.

CREATE EVENT MONITOR (actividades)

- Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
 - LOG_STREAM_HEADER
 - LOG_HEADER
 - DB_HEADER (Los elementos **db_name** y **db_path** no se graban. El elemento **conn_time** se graba en CONTROL.)
- En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base de datos con el fin de supervisarlas, creando un espacio de tablas que sólo exista en determinadas particiones de base de datos. En un entorno DB2 pureScale, los datos se grabarán desde cada miembro.

En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.
- Los usuarios pueden podar manualmente todas las tablas de destino.

Columnas de tabla:

- Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. Cualquier elemento del supervisor de sucesos que no tenga una columna de tabla de destino correspondiente se pasa por alto.
- Utilice el mandato **db2evtbl** para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.
- Los tipos de columnas que se utilizan para los elementos del supervisor corresponden a la correlación siguiente:

SQLM_TYPE_STRING	CHAR[n], VARCHAR[n] o CLOB(n) (Si los datos del registro del supervisor de sucesos exceden de <i>n</i> bytes, se truncan.)
SQLM_TYPE_U8BIT y SQLM_TYPE_8BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_16BIT y SQLM_TYPE_U16BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_32BIT y SQLM_TYPE_U32BIT	INTEGER o BIGINT
SQLM_TYPE_U64BIT y SQLM_TYPE_64BIT	BIGINT
sqlm_timestamp	TIMESTAMP
sqlm_time(tiempo transcurrido)	BIGINT
sqlca:	
sqlerrmc	VARCHAR[72]
sqlstate	CHAR[5]
sqlwarn	CHAR[11]
otros campos	INTEGER o BIGINT

- Las columnas se han definido para que sean NOT NULL.
- A diferencia de otras tablas de destino, las columnas de la tabla CONTROL no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

Nombre columna	Tipo datos	Nulos	Descripción
-----	-----	-----	-----
PARTITION_KEY	INTEGER	N	Clave de distribución (sólo base de datos particionada)
PARTITION_NUMBER	INTEGER	N	Número de partición de base de datos (sólo base de datos)

CREATE EVENT MONITOR (actividades)

EVMONNAME	VARCHAR(128)	N	particionada)
MESSAGE	VARCHAR(128)	N	Nombre de supervisor de sucesos
			Describe la naturaleza de la columna MESSAGE_TIME. Puede ser uno de los valores siguiente:
			- FIRST_CONNECT (la hora de la primera conexión con la base de datos tras la activación)
			- EVMON_START (la hora en que se inició el supervisor de sucesos listado en EVMONNAME)
			- OVERFLOWS:n (indica que se han descartado n registros por desbordamiento del almacenamiento intermedio)
			- LAST_DROPPED_RECORD (última vez en que se ha producido un desbordamiento)
MESSAGE_TIME	TIMESTAMP	N	Indicación de fecha y hora

- En un entorno de base de datos particionada, la primera columna de cada tabla se denomina PARTITION_KEY, es NOT NULL y es de tipo INTEGER. Esta columna se utiliza como clave de distribución para la tabla. El valor de esta columna se elige para que cada proceso de supervisor de sucesos inserte datos en el miembro en el que se ejecute el proceso, es decir, las operaciones de inserción se llevan a cabo localmente en el miembro donde se está ejecutando el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se descarta una partición de base de datos y se realiza la redistribución de los datos, todos los datos de la partición de base de datos que se ha descartado se dirigirán a otra partición de base de datos en lugar de distribuirse equitativamente. Por lo tanto, antes de eliminar una partición de base de datos, considere la supresión de todas las filas de tabla de esa partición de base de datos.
- En un entorno de base de datos particionada, puede definirse una columna denominada PARTITION_NUMBER para cada tabla. Esta columna es NOT NULL y es de tipo INTEGER. Contiene el número de la partición de base de datos en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de base de datos no particionada.

Atributos de tabla:

- Se utilizan atributos de tabla por omisión. Además de la clave de distribución (sólo bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
- En la tabla puede crearse índices.
- Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
- Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.

Activación del supervisor de sucesos:

- Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.

CREATE EVENT MONITOR (actividades)

- En un entorno de base de datos particionada, el proceso de activación se produce en cada miembro de la instancia. En un miembro en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en una partición de base de datos si existe, como mínimo, una tabla de destino en esa partición de base de datos. Además, si no se encuentra alguna tabla de destino en una partición de base de datos, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.
- Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de base de datos particionada, si el espacio de tablas no reside en una partición de base de datos), la activación continúa y los datos que, de otro modo, se insertarían en esta tabla, se pasan por alto.
- El proceso de activación valida cada tabla de destino. Si la validación no se realiza satisfactoriamente, la activación del supervisor de sucesos no tiene lugar y se graban mensajes en el archivo de anotaciones cronológicas de administración.
- Durante la activación en un entorno de base de datos particionada, las filas de la tabla CONTROL de FIRST_CONNECT y de EVMON_START sólo se insertan en la partición de base de datos de catálogo. Para ello es necesario que el espacio de tablas de la tabla CONTROL exista en la partición de base de datos de catálogo. Si no existe en la partición de base de datos de catálogo, dichas inserciones no se realizan.
- En un entorno de base de datos particionada, si un miembro todavía no está activo al activarse el supervisor de sucesos de grabación en tabla, el supervisor de sucesos se activará la próxima vez que se active el miembro.

Tiempo de ejecución:

- El supervisor de sucesos se ejecuta con autorización DATAACCESS.
- Si, mientras existe un supervisor de sucesos activo, no se ejecuta satisfactoriamente una operación de inserción en una tabla de destino:
 - Los cambios no confirmados se retrotraen.
 - Se graba un mensaje en el archivo de anotaciones cronológicas de administración.
 - El supervisor de sucesos se desactiva.
- Si existe un supervisor de sucesos activo, ejecuta COMMIT localmente cuando ha terminado de procesar un almacenamiento intermedio del supervisor de sucesos.
- En un entorno que no sea una base de datos particionada o un entorno DB2 pureScale, todos los supervisores de sucesos de grabación en tabla se desactivarán cuando finalice la última aplicación (y la base de datos no se haya activado de forma explícita). En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla se desactivan en un miembro concreto cuando el miembro se detiene y se vuelve a activar cuando el miembro se reinicia. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se desactivan al desactivarse la partición de catálogo.
- La sentencia DROP EVENT MONITOR no descarta tablas de destino.
- Siempre que se active un supervisor de sucesos de grabación en tabla, obtendrá bloqueos de tabla IN en cada tabla de destino para impedir que se modifiquen mientras el supervisor de sucesos esté activo. Los bloqueos de tabla se mantienen en todas las tablas mientras el supervisor de sucesos está activo. Si en cualquiera de estas tablas de destino es necesario un acceso

CREATE EVENT MONITOR (actividades)

exclusivo (por ejemplo, cuando se va a ejecutar un programa de utilidad), primero desactive el supervisor de sucesos para liberar los bloqueos de tabla antes de intentar dicho acceso.

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Se pueden especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Las comas se pueden utilizar para separar varias opciones en la cláusula *opciones-tabla-destino*

Ejemplo

Definir un supervisor de sucesos de actividad denominado DB2ACTIVITIES

```
CREATE EVENT MONITOR DB2ACTIVITIES
  FOR ACTIVITIES
  WRITE TO TABLE
  ACTIVITY (TABLE ACTIVITY_DB2ACTIVITIES
            IN USERSPACE1
            PCTDEACTIVATE 100),
  ACTIVITYMETRICS (TABLE ACTIVITYMETRICS_DB2ACTIVITIES
                   IN USERSPACE1
                   PCTDEACTIVATE 100),
  ACTIVITYSTMT (TABLE ACTIVITYSTMT_DB2ACTIVITIES
                IN USERSPACE1
                PCTDEACTIVATE 100),
  ACTIVITYVALS (TABLE ACTIVITYVALS_DB2ACTIVITIES
                IN USERSPACE1
                PCTDEACTIVATE 100),
  CONTROL (TABLE CONTROL_DB2ACTIVITIES
            IN USERSPACE1
            PCTDEACTIVATE 100)
AUTOSTART;
```

CREATE EVENT MONITOR (historial de cambios)

La sentencia CREATE EVENT MONITOR (historial de cambios) crea un supervisor de sucesos que puede registrar sucesos para cambios en los parámetros de configuración, cambios en las variables del registro y la ejecución de sentencias DDL y programas de utilidad.

El supervisor de sucesos creado por la sentencia CREATE EVENT MONITOR (historial de cambios) también puede registrar la configuración inicial y los valores de registro en el momento del inicio del supervisor de sucesos. El conjunto de sucesos registrados depende de los controles de suceso especificados en la sentencia CREATE EVENT MONITOR.

Invocación

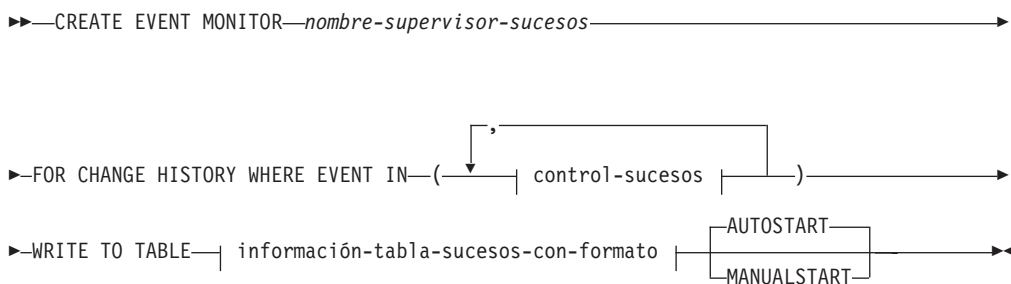
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización SQLADM
- Autorización DBADM

Sintaxis

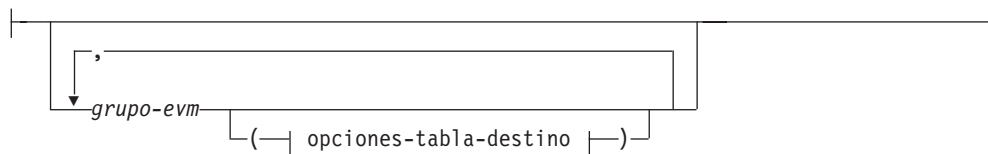


control-sucesos:

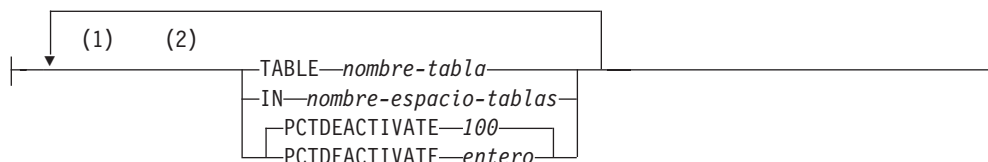
CREATE EVENT MONITOR (historial de cambios)

ALL
BACKUP
CFGALL
DBCFG
DBCFGVALUES
DBMCFG
DBMCFGVALUES
DDLALL
DDLDATA
DDLFEDERATED
DDLMONITOR
DDLSECURITY
DDLSQL
DDLSTORAGE
DDLWLM
DDLXML
LOAD
MOVETABLE
REDISTRIBUTE
REGVAR
REGVARVALUES
REORG
RESTORE
ROLLFORWARD
RUNSTATS
UTILALL

info-tabla-sucesos-con-formato:



opciones-tabla-destino:



Notas:

- 1 Cada condición sólo puede especificarse una vez (SQLSTATE 42613).
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que exista en el catálogo (SQLSTATE 42710).

CREATE EVENT MONITOR (historial de cambios)

FOR

Especifica el tipo de suceso que se debe registrar.

CHANGE HISTORY

Especifica que este supervisor de sucesos puede registrar sucesos para cambios en la configuración, cambios en el registro y para la ejecución de sentencias DDL y programas de utilidad. También puede registrar la configuración inicial y los valores de registro en el momento del inicio del supervisor de sucesos. El conjunto de sucesos registrados depende de los controles de sucesos especificados en la cláusula WHERE EVENT IN.

WHERE EVENT IN (control-sucesos, ...)

Especifica uno o más controles de suceso utilizados para identificar qué sucesos captura el supervisor de sucesos.

control-sucesos

ALL Captura todos los tipos de sucesos.

BACKUP

Captura la ejecución del programa de utilidad de copia de seguridad en línea.

CFGALL

Captura todos los parámetros de configuración y los tipos de sucesos de variable de registro.

DBCFG

Captura los cambios en los parámetros de configuración de la base de datos.

DBCFGVALUES

Registra los valores iniciales para todos los parámetros de configuración de la base de datos en el momento del inicio del supervisor de sucesos si el supervisor de sucesos no ha capturado la actualización de alguno de los parámetros de configuración de la base de datos.

DBMCFG

Captura los cambios en los parámetros de configuración del gestor de la base de datos.

DBMCFGVALUES

Registra los valores iniciales para todos los parámetros de configuración del gestor de la base de datos en el momento del inicio del supervisor de sucesos si el supervisor de sucesos no ha capturado la actualización de alguno de los parámetros de configuración del gestor de la base de datos.

DDLALL

Captura la ejecución para todos los tipos de sentencias DDL.

DDLDATA

Captura la ejecución de índices, secuencias, tablas y DDL de tablas temporales.

DDLFEDERATED

Captura la ejecución de apodos, servidores, correlación de usuarios y DDL de derivadores.

DDLMONITOR

Captura la ejecución del supervisor de sucesos y DDL de lista de uso.

CREATE EVENT MONITOR (historial de cambios)

DDLSECURITY

Captura la ejecución de políticas de auditoría, asignaciones, máscaras, roles de permiso, revocaciones, etiquetas de seguridad, componentes de etiquetas de seguridad, políticas de seguridad y DDL de contexto fiable.

DDLSQL

Captura la ejecución de alias, funciones, métodos, módulos, paquetes, procedimientos, esquemas, sinónimos, transformaciones, activadores, tipos, variables y DDL de visualización.

DDLSTORAGE

Captura la ejecución de la sentencia ALTER DATABASE, agrupaciones de almacenamiento intermedio, grupos de particiones, grupos de almacenamientos y DDL de espacios de tablas.

DDLWLM

Captura la ejecución de histogramas, clases de servicio, umbrales, conjuntos de acciones de trabajo, conjuntos de clases de trabajo y DDL de cargas de trabajo.

DDLXML

Captura la ejecución de XSROBJECT DDL.

LOAD

Captura la ejecución del programa de utilidad de carga.

MOVETABLE

Captura la ejecución del programa de utilidad de movimiento de tablas (invocaciones del procedimiento almacenado ADMIN_MOVE_TABLE).

REDISTRIBUTE

Captura la ejecución del programa de utilidad del grupo de particiones de redistribución.

REGVAR

Captura los cambios en las variables del registro inmediato.

REGVARVALUES

Registra los valores iniciales de las variables de registro en el momento del inicio del supervisor de sucesos.

REORG

Captura la ejecución del programa de utilidad de reorganización.

RESTORE

Captura la ejecución del programa de utilidad de restauración en línea.

ROLLFORWARD

Captura la ejecución del programa de utilidad de recuperación en avance en línea.

RUNSTATS

Captura la ejecución del programa de utilidad runstats.

UTILALL

Captura la ejecución de los programas de utilidad de carga, de movimiento de tablas, de copia de seguridad en línea, de restauración en línea, de recuperación en avance en línea, de redistribución, de reorganización y runstats.

CREATE EVENT MONITOR (historial de cambios)

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente.

info-tabla-sucesos-con-formato

Define la tabla de destino de un grupo de datos lógicos. Especifique esta cláusula para cada agrupación que ha de registrarse. Sin embargo, si no se especifica ninguna cláusula *grupo-evm*, los grupos necesarios para las opciones control-suceso especificadas se crearán junto con los grupos lógicos CONTROL, CHANGESUMMARY y EVMONSTART.

grupo-evm

Identifica el grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-evm
Historial de cambios	<ul style="list-style-type: none">• CONTROL• CHANGESUMMARY• EVMONSTART• TXNCOMPLETION• DDLSTMTEXEC• DBDBMCFG• REGVAR• UTILSTART• UTILSTOP• UTILPHASE• UTILLOCATION

opciones-tabla-destino

Identifica la tabla de destino de un grupo.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
SUBSTRING(grupo-evm CONCAT ' '
          CONCAT nombre-supervisor-suceso, 1, 128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

CREATE EVENT MONITOR (historial de cambios)

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llena. El valor por omisión es 100 si no se ha especificado PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

Si no se especifica un valor de info-tabla-destino, el proceso de CREATE EVENT MONITOR continúa de la manera siguiente:

- Se utiliza un nombre de tabla derivado.
- Se elige un espacio de tablas por omisión.
- El parámetro PCTDEACTIVATE se establece por omisión en 100.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Éste es el comportamiento por omisión.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

Notas

- **Creación de tablas de sucesos de destino:** Las tablas de sucesos de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR FOR CHANGE HISTORY si las tablas de destino no existen.
- **Tablas de sucesos creadas anteriormente:** Durante el proceso de CREATE EVENT MONITOR FOR CHANGE HISTORY, si ya se ha definido una tabla de sucesos para que la utilice otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR FOR CHANGE HISTORY falla y se devuelve un error al programa de aplicación. Una tabla de sucesos se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTTABLES. Si la tabla de sucesos existe y ésta no se ha definido para que la utilice otro supervisor de sucesos, no se creará una tabla, se pasará por alto cualquier otro parámetro de opciones de tabla y el proceso continuará. Se devolverá un aviso al programa de aplicación.
- **Descarte de supervisores de sucesos:** Al descartar el supervisor de sucesos no se descarta la tabla de sucesos. Las tablas de sucesos asociadas deben descartarse manualmente después de que se descarte el supervisor de sucesos.
- **Podar:** las tablas de sucesos deben podarse manualmente.

CREATE EVENT MONITOR (historial de cambios)

- *Comportamiento en un entorno particionado:* En un entorno particionado, si varias tablas de sucesos de destino no existen en una partición, pero otras tablas de sucesos de destino sí existen en esa misma partición, sólo se registran los datos para las tablas de sucesos de destino que existen en esa partición.
- **FLUSH EVENT MONITOR:** La sentencia FLUSH EVENT MONITOR no es aplicable a este supervisor de sucesos y no se aplica cuando se emite contra este.
- *Modificación de controles de suceso después de crear un supervisor:* después de crear el supervisor de sucesos del historial de cambios, los controles de suceso especificados utilizando la cláusula WHERE EVENT IN de la sentencia CREATE EVENT MONITOR no se pueden cambiar ni modificar. Para cambiar los controles de suceso, el supervisor de sucesos debe desactivarse, descartarse y, a continuación, volver a crearse con el nuevo conjunto de controles de suceso utilizando la cláusula WHERE EVENT IN.

Ejemplos

- *Ejemplo 1:* Este ejemplo crea un supervisor de sucesos del historial de cambios denominado CFG_WITH_OFFLINE que registra los cambios y los valores iniciales de la configuración.

```
CREATE EVENT MONITOR CFG_WITH_OFFLINE
  FOR CHANGE HISTORY WHERE EVENT IN (CFGALL)
  WRITE TO TABLE
    CHANGESUMMARY (TABLE CHG_SUMMARY_HISTORY),
    DBDBMCFG (TABLE DB_DBM_HISTORY),
    REGVAR (TABLE REGVAR_HISTORY)
  AUTOSTART
```

En este ejemplo, las tablas de destino se especifican de forma explícita. La sentencia anterior crea las tablas siguientes:

```
CHG_SUMMARY_HISTORY
DB_DBM_HISTORY
REGVAR_HISTORY
```

- *Ejemplo 2:* Este ejemplo crea un supervisor de sucesos del historial de cambios denominado BKP_REST que recopila los sucesos que describen toda la copia de seguridad en línea y restaura las ejecuciones de los programas de utilidad.

```
CREATE EVENT MONITOR BKP_REST
  FOR CHANGE HISTORY WHERE EVENT IN (BACKUP, RESTORE)
  WRITE TO TABLE
```

En este ejemplo, las tablas de destino no se especifican de forma explícita. La sentencia CREATE EVENT MONITOR sólo crea las tablas de destino necesarias en función de los controles especificados en la cláusula WHERE EVENT IN, junto con las tablas para los grupos de datos lógicos CONTROL, CHANGESUMMARY y EVMONSTART. Los controles BACKUP y RESTORE permiten la recopilación de sucesos de programas de utilidad para realizar copias de seguridad en línea y restaurar los grupos de datos lógicos UTILSTART, UTILSTOP, UTILLOCATION y UTILPHASE. La sentencia anterior crea las tablas siguientes:

```
CONTROL_BKP_REST
CHANGESUMMARY_BKP_REST
EVMONSTART_BKP_REST
UTILSTART_BKP_REST
UTILSTOP_BKP_REST
UTILLOCATION_BKP_REST
UTILPHASE_BKP_REST
```

CREATE EVENT MONITOR (bloqueo)

La sentencia CREATE EVENT MONITOR (bloqueo) crea un supervisor que registrará los sucesos relacionados con el bloqueo que se produzcan cuando se utilice la base de datos.

Invocación

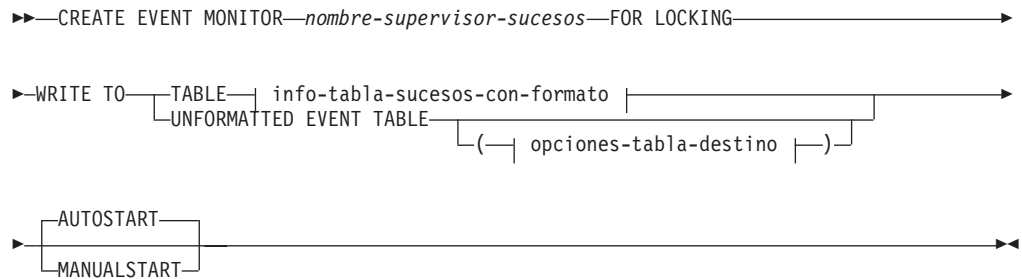
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

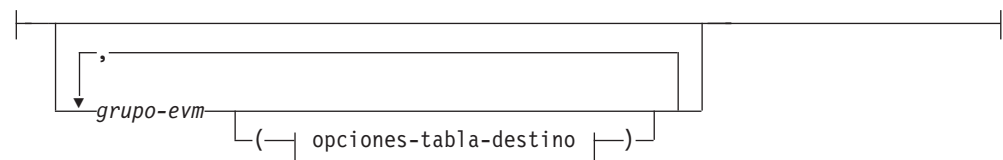
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM

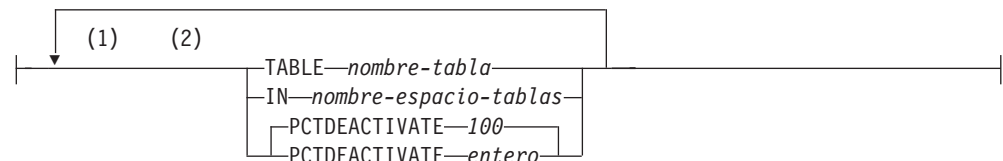
Sintaxis



info-tabla-sucesos-con-formato:



opciones-tabla-destino:



Notas:

- 1 Cada opción de tabla se puede especificar una vez como máximo (SQLSTATE 42613).
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

CREATE EVENT MONITOR (bloqueo)

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

LOCKING

Especifica que este supervisor de sucesos pasivo registrará cualquier bloqueo que se genere cuando DB2 se ejecute en una o más de estas condiciones:

- LOCKTIMEOUT: el bloqueo ha superado el tiempo de espera.
- DEADLOCK: el bloqueo estaba implicado en un punto muerto (víctima y participantes).
- LOCKWAIT: los bloqueos no se adquieren en la duración especificada.

La creación del supervisor de sucesos de bloqueo no indica que los datos de bloqueo se recopilarán de manera inmediata. El suceso de bloqueo real de interés se controla en el nivel de la carga de trabajo o de la base de datos.

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de sucesos con formato. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-tabla-sucesos-con-formato

Define las tablas de sucesos con formato de destino para un supervisor de sucesos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula *grupo-evm*, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-evm

Identifica un grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-evm
Bloqueo	<ul style="list-style-type: none"> • LOCK • LOCK_PARTICIPANTS • LOCK_PARTICIPANT_ACTIVITIES • LOCK_ACTIVITY_VALUES • CONTROL

UNFORMATTED EVENT TABLE

Especifica que el destino del supervisor de sucesos es una tabla de sucesos sin formato. La tabla de sucesos sin formato se utiliza para almacenar los datos de supervisor de sucesos de bloqueo recopilados. Los datos se almacenan en un formato binario interno dentro de una columna BLOB en línea. Cada suceso puede insertar múltiples registros en esta tabla, cada uno de los cuales puede ser de distinto tipo y tener también contenido BLOB asociado diferente. Los datos de la columna BLOB no tienen formato legible y es necesaria su conversión, mediante la herramienta **db2evmonfmt** basada en Java, la función de tabla `EVMON_FORMAT_UE_TO_XML` o el procedimiento `EVMON_FORMAT_UE_TO_TABLES`, a un formato utilizable, como un documento XML o una tabla relacional.

opciones-tabla-destino

Identifica opciones para la tabla de destino. Si no se especifica un valor para **opciones-tabla-destino**, el proceso `CREATE EVENT MONITOR LOCKING` continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado (tal y como se explica en la descripción de `TABLE nombre-tabla`).
- Un espacio de tablas por omisión se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando `CREATE TABLE`.
- `PCTDEACTIVATE` se establece en 100.

TABLE nombre-tabla

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial `CURRENT SCHEMA`. Si no se proporciona un nombre para una tabla de sucesos sin formato, el nombre no calificado es igual al *nombre-supervisor-sucesos*, es decir, a la tabla de sucesos sin formato se le asignará un nombre según el supervisor de sucesos. Si no se proporciona ningún nombre para una tabla de sucesos sin formato, el nombre no calificado se obtiene de *grupo-evm* y *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'
          CONCAT nombre-supervisor-sucesos,1,128)
```

IN nombre-espacio-tablas

Define el espacio de tablas en el que va a crearse la tabla. La sentencia `CREATE EVENT MONITOR FOR LOCKING` no crea espacios de tablas.

Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando `CREATE TABLE`.

CREATE EVENT MONITOR (bloqueo)

Al especificar un nombre de espacio de tablas para una tabla de sucesos con formato, el tamaño de la página del espacio de tablas afecta a las longitudes de INLINE LOB utilizadas. Considere la posibilidad de especificar un espacio de tablas con un tamaño de página lo más grande posible para mejorar el rendimiento de INSERT del supervisor de sucesos.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100. El valor por omisión es 100, donde 100 significa que el supervisor de sucesos se desactivará cuando el espacio de tablas se haya llenado por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Es el comportamiento por omisión del supervisor de sucesos de bloqueo.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

Notas

- La tabla de destino se crea cuando la sentencia CREATE EVENT MONITOR FOR LOCKING se ejecuta, si no existe ya.
- Durante el proceso de CREATE EVENT MONITOR FOR LOCKING, si se encuentra una tabla que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR FOR LOCKING no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES. Si la tabla existe y no se define para que la utilice otro supervisor de sucesos, el supervisor de sucesos volverá a utilizar la tabla.
- Al descartar el supervisor de sucesos no se descartará ninguna tabla. Las tablas asociadas deben descartarse después de que se descarte el supervisor de sucesos.
- Los datos de suceso de bloqueo no se depurarán de forma automática desde las tablas de sucesos sin formato o desde las tablas comunes creadas por el

supervisor de sucesos. Al utilizar el proceso EVMON_FORMAT_UE_TO_TABLES dispondrá de una opción para depurar datos desde las tablas UE. Para los supervisores de sucesos que graban tablas comunes, los datos de suceso se deben depurar de forma manual.

- La sentencia FLUSH EVENT MONITOR no es aplicable a este supervisor de sucesos y no se aplicará cuando se emita contra el mismo.
- Los datos de sucesos de tablas de sucesos sin formato se insertan en la tabla en una columna de datos BLOB en línea. Normalmente, los datos BLOB se almacenan en un espacio de tablas LOB independiente y pueden experimentar como resultado una actividad general adicional del rendimiento. Cuando los datos BLOB se colocan en línea en la página de datos de la tabla base, no experimentan dicha actividad general. El gestor de bases de datos DB2 colocará automáticamente en línea la parte de datos BLOB de una tabla de sucesos sin formato si el tamaño de los datos BLOB es inferior al tamaño de página del espacio de tablas menos el prefijo del registro. Por lo tanto, para obtener una eficacia y un rendimiento elevados de las aplicaciones, se sugiere crear el supervisor de sucesos en una tabla de sucesos lo más grande posible hasta 32 KB inclusive para el espacio de tablas y la agrupación de almacenamientos intermedios asociada.

Ejemplo

El supervisor de sucesos de bloqueo cuenta actualmente con los dos tipos de registro siguientes:

- Registro de información de la aplicación
- Registro de actividad de la aplicación

Registro de información de la aplicación = tamaño máximo de 3,5 KB

Registro de actividad de la aplicación = 3 KB + tamaño de texto de la sentencia de SQL (siendo el tamaño de texto de la sentencia de SQL de 2 MB como máximo)

El registro de información de la aplicación es muy pequeño y siempre debería colocarse en línea, siempre y cuando se esté utilizando un tamaño de página de 4 KB. El registro de actividad de la aplicación estará en línea de acuerdo con las fórmulas siguientes:

Informe de actividad de la aplicación < longitud en línea
(Tamaño de página - columnas no LOB de actividad general (0,5 KB))
3 KB + texto de sentencia de SQL < longitud en línea
(Tamaño de página - columnas no LOB de actividad general (0,5 KB))

Texto de sentencia de SQL < Tamaño de página - actividad general no LOB (1 K) - 3 KB
Texto de sentencia de SQL < 16 KB - 1 KB - 3 KB
< 12 KB

Por consiguiente, si se utiliza un tamaño de página de 16 KB, los registros del supervisor de sucesos de bloqueo sólo estarán en línea si la sentencia de SQL que se está capturando tiene un tamaño inferior a 12 KB.

- Sólo se debe crear un supervisor de sucesos de bloqueo por base de datos. La creación de más de un supervisor de sucesos de bloqueo utiliza ciclos de procesador y almacenamiento adicionales, sin proporcionar datos adicionales.

Importante: Para mantener la compatibilidad con las versiones anteriores del producto, todas las bases de datos se crean con el supervisor de sucesos DB2DETAILDEADLOCK habilitado. El supervisor de sucesos de bloqueo presentado en DB2 Versión 9.7 es el mecanismo preferido para la recopilación de datos relacionados con los bloqueos; el supervisor de sucesos DB2DETAILEDDEALOCK está en desuso y podría eliminarse en un futuro

CREATE EVENT MONITOR (bloqueo)

release. Cuando cree un supervisor de sucesos de bloqueo, inhabilite y descarte el supervisor de sucesos DB2DETAILEDDEADLOCK para evitar la recopilación de información duplicada innecesaria.

Para eliminar el supervisor de sucesos DB2DETAILDEADLOCK, emita las sentencias de SQL siguientes:

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

- En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base de datos seleccionables, creando un espacio de tablas que sólo exista en determinadas particiones de base de datos.
- En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de sucesos de destino sin formato que residan en esa partición de base de datos.

Ejemplos

- *Ejemplo 1:* en este ejemplo se crea un supervisor de sucesos de bloqueo LOCKEVMON que recopilará los sucesos de bloqueo que tengan lugar en la base de datos de creación.

```
CREATE EVENT MONITOR LOCKEVMON
FOR LOCKING
WRITE TO TABLE
```

Este supervisor de sucesos graba su salida en las tablas siguientes:

```
LOCK_LOCKEVMON
LOCK_PARTICIPANTS_LOCKEVMON
LOCK_PARTICIPANT_ACTIVITIES_LOCKEVMON
LOCK_ACTIVITY_VALUES_LOCKEVMON
CONTROL_LOCKEVMON
```

- *Ejemplo 2:* en este ejemplo se crea un supervisor de sucesos de bloqueo LOCKEVMON que recopilará los sucesos de bloqueo que tengan lugar en la base de datos de creación y los almacenará en la tabla de sucesos sin formato IMRAN.LOCKEVENTS.

```
CREATE EVENT MONITOR LOCKEVMON
FOR LOCKING
WRITE TO UNFORMATTED EVENT TABLE (TABLE IMRAN.LOCKEVENTS)
```

- *Ejemplo 3:* en este ejemplo se crea un supervisor de sucesos de bloqueo LOCKEVMON que recopilará los sucesos de bloqueo que tengan lugar en la base de datos de creación y los almacenará en la tabla de sucesos sin formato IMRAN.LOCKEVENTS en el espacio de tablas APPSPACE. El supervisor de sucesos se desactivará cuando el espacio de tablas esté un 85% lleno.

```
CREATE EVENT MONITOR LOCKEVMON
FOR LOCKING
WRITE TO UNFORMATTED EVENT TABLE
(TABLE IMRAN.LOCKEVENTS IN APPSPACE PCTDEACTIVATE 85)
```

Sentencia CREATE EVENT MONITOR (antememoria de paquete)

La sentencia CREATE EVENT MONITOR (antememoria de paquete) crea un supervisor de sucesos que registrará los sucesos cuando la entrada de antememoria de una sección se vacíe de la antememoria de paquete.

Invocación

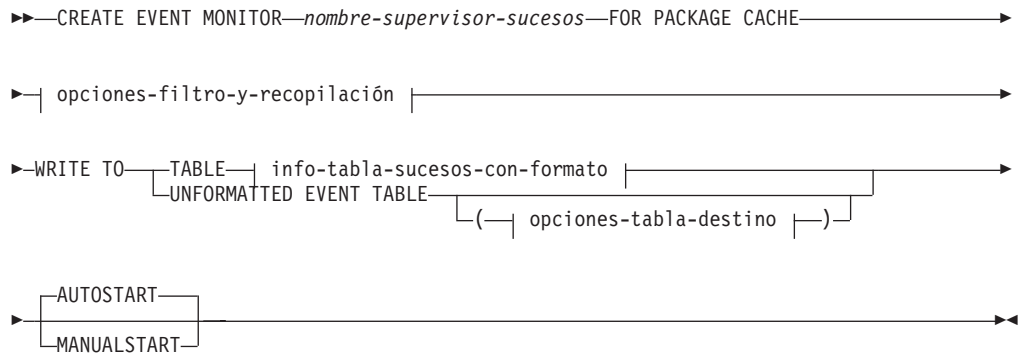
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

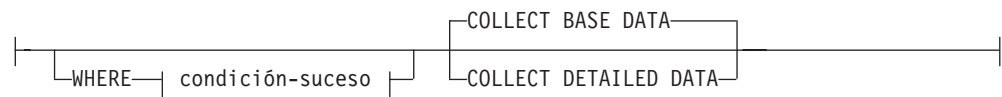
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM

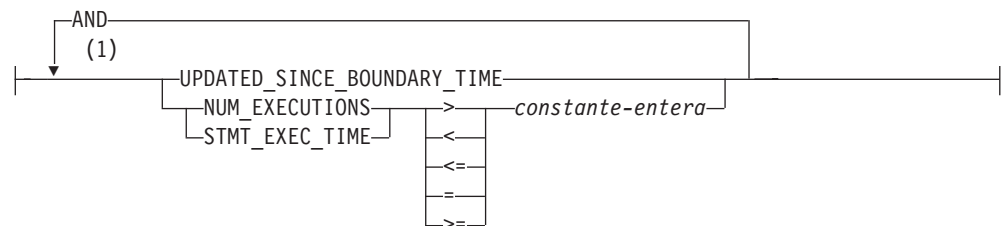
Sintaxis



opciones-filtro-y-recopilación:

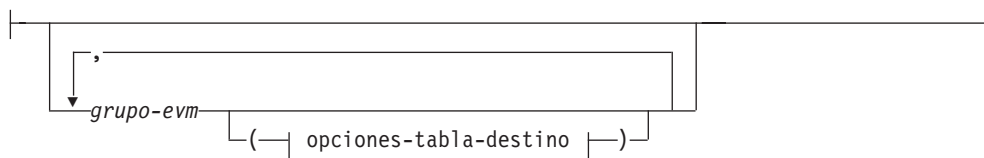


condición-suceso:

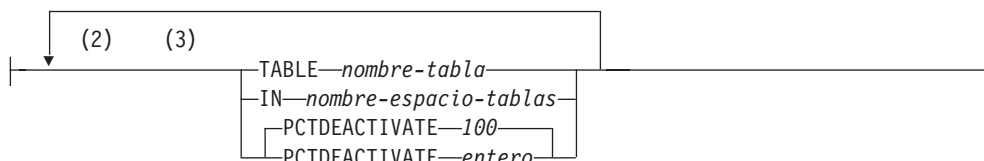


Sentencia CREATE EVENT MONITOR (antememoria de paquete)

info-tabla-sucesos-con-formato:



opciones-tabla-destino:



Notas:

- 1 Cada condición sólo puede especificarse una vez (SQLSTATE 42613).
- 2 Cada opción de tabla se puede especificar una vez como máximo (SQLSTATE 42613).
- 3 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

PACKAGE CACHE

Especifica que este supervisor de sucesos registrará un suceso cuando la entrada de antememoria de una sentencia de SQL estático o dinámico se vacíe de la antememoria de paquete. Este supervisor de sucesos no es un supervisor pasivo e iniciará el registro de sucesos en cuanto se active.

opciones-filtro-y-recopilación

Especifique un conjunto de opciones de filtro y recopilación.

WHERE

condición-suceso

Define un filtro que determina si las entradas que se vacían de la antememoria de paquete deben dar lugar a que se produzca un suceso. Si la condición del suceso es TRUE para una entrada en particular que está vaciándose de la antememoria de paquete, esa entrada se registrará como suceso.

Esta cláusula es una forma especial de la cláusula WHERE que no debe confundirse con una condición de búsqueda estándar. Se trata de una simple cláusula WHERE que incluye la utilización de los

Sentencia CREATE EVENT MONITOR (antememoria de paquete)

operadores NOT, OR y LIKE, a diferencia de la cláusula WHERE que se especifica para los supervisores de sucesos CONNECTIONS, TRANSACTIONS y STATEMENTS.

Si no se especifica la cláusula WHERE, se supervisarán todas las entradas que se vacíen de la antememoria de paquete.

UPDATED_SINCE_BOUNDARY_TIME

Especifica que este supervisor de sucesos ha de recopilar las entradas desalojadas, cuyas métricas se han actualizado después de haber transcurrido el tiempo límite. El tiempo límite se establece mediante una llamada a la función de tabla MON_GET_PKG_CACHE_STMT con el valor de la clave de entrada "tiempo_límite_actualizado" establecido como el nombre de este supervisor de sucesos.

Inicialmente, el tiempo límite se establece en la indicación de fecha y hora de activación del supervisor de sucesos.

NUM_EXECUTIONS > | < | <= | = | >= *constante-entero*

Especifica que el elemento de supervisor **num_executions** debe compararse con el valor *constante-entero* para determinar si ha de generarse o no un suceso. NUM_EXECUTIONS es el número de veces que se ha ejecutado la sección de la entrada desalojada.

Nota: El elemento de supervisor **num_executions** cuenta todas las ejecuciones de una sentencia, con independencia de si la ejecución de la sentencia ha pasado a formar parte de las métricas de actividad de las que se informa.

STMT_EXEC_TIME > | < | <= | = | >= *constante-entero*

Especifica que el elemento de supervisor **stmt_exec_time** debe compararse con el valor *constante-entero* para determinar si ha de generarse o no un suceso. STMT_EXEC_TIME es el tiempo agregado total que se ha utilizado para ejecutar la sentencia de la entrada desalojada. La unidad de tiempo de *constante-entero* debe especificarse en milisegundos.

COLLECT BASE DATA

Especifica que debe capturarse el mismo nivel de información que el que devuelve la función de tabla MON_GET_PKG_CACHE_STMT. Es la opción de recopilación por omisión.

COLLECT DETAILED DATA

Especifica que debe recopilarse la información de nivel BASE, así como la sección ejecutable en tiempo de ejecución de la entrada desalojada.

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

Sentencia CREATE EVENT MONITOR (antememoria de paquete)

info-tabla-sucesos-con-formato

Define las tablas de sucesos con formato de destino para un supervisor de sucesos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula *grupo-vm*, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-vm

Identifica un grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-vm
Antememoria de paquete	<ul style="list-style-type: none">• PKGCACHE• PKGCACHE_METRICS• PKGCACHE_STMT_ARGS• CONTROL

UNFORMATTED EVENT TABLE

Especifica que el destino del supervisor de sucesos es una tabla de sucesos sin formato. La tabla de sucesos sin formato se utiliza para almacenar los datos de supervisor de sucesos de antememoria de paquete que se han recopilado. Los datos se almacenan en su formato binario original dentro de una columna BLOB en línea. La columna BLOB puede contener varios registros binarios de tipos distintos. Los datos de la columna BLOB no tienen formato legible y es necesaria su conversión, mediante la herramienta **db2evmonfmt** basada en Java, la función de tabla `EVMON_FORMAT_UE_TO_XML` o el procedimiento `EVMON_FORMAT_UE_TO_TABLES`, a un formato utilizable, como un documento XML o una tabla relacional.

opciones-tabla-destino

Identifica opciones para la tabla de destino. Si no se especifica un valor para **opciones-tabla-destino**, el proceso CREATE EVENT MONITOR FOR PACKAGE CACHE continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado (tal y como se explica en la descripción de `TABLE nombre-tabla`).
- Un espacio de tablas por omisión se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando `CREATE TABLE`.
- `PCTDEACTIVATE` se establece en 100.

`TABLE nombre-tabla`

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial `CURRENT SCHEMA`. Si no se proporciona un nombre para una tabla de sucesos sin formato, el nombre no calificado es igual al *nombre-supervisor-sucesos*, es decir, a la tabla de sucesos sin formato se le asignará un nombre según el supervisor de sucesos. Si no se proporciona ningún nombre para una tabla de sucesos sin formato, el nombre no calificado se obtiene de *grupo-vm* y *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-vm CONCAT '_'  
CONCAT nombre-supervisor-sucesos,1,128)
```


Sentencia CREATE EVENT MONITOR (antememoria de paquete)

IN *nombre-espacio-tablas*

Especifica el espacio de tablas en el que debe crearse la tabla. La sentencia CREATE EVENT MONITOR FOR PACKAGE CACHE no crea espacios de tablas.

Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

El tamaño de la página del espacio de tablas afecta a las longitudes de INLINE LOB utilizadas. Considere la posibilidad de especificar un espacio de tablas con un tamaño de página lo más grande posible para mejorar el rendimiento de INSERT del supervisor de sucesos.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Es el comportamiento por omisión del supervisor de sucesos de antememoria de paquete.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que la ejecución de MANUALSTART haya activado el supervisor de sucesos, éste puede desactivarse mediante la utilización de la sentencia SET EVENT MONITOR STATE o bien mediante la detención de la instancia.

Notas

- La tabla de destino se crea al ejecutarse la sentencia CREATE EVENT MONITOR FOR PACKAGE CACHE, si todavía no existe.
- Durante el proceso CREATE EVENT MONITOR FOR PACKAGE CACHE, si se detecta que ya se había definido una tabla de destino para que la utilice otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR FOR PACKAGE CACHE no se ejecutará correctamente y se devolverá un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la

Sentencia CREATE EVENT MONITOR (antememoria de paquete)

vista de catálogo SYSCAT.EVENTTABLES. Si la tabla existe y no se define para que la utilice otro supervisor de sucesos, el supervisor de sucesos volverá a utilizar la tabla.

- Al descartar el supervisor de sucesos no se descartará ninguna tabla. Las tablas asociadas deben descartarse después de que se descarte el supervisor de sucesos.
- Los datos de suceso de bloqueo no se depurarán de forma automática desde las tablas de sucesos sin formato o desde las tablas comunes creadas por el supervisor de sucesos. Al utilizar el proceso EVMON_FORMAT_UE_TO_TABLES dispondrá de una opción para depurar datos desde las tablas UE. Para los supervisores de sucesos que graban tablas comunes, los datos de suceso se deben depurar de forma manual.
- En un entorno de base de datos particionada, los datos se graban en las tablas de destino únicamente de los miembros en los que existen sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en un miembro, en este miembro no se capturan los datos de suceso que se grabarían en esta tabla de destino. Este comportamiento permite a los usuarios seleccionar un subconjunto de los miembros que han de supervisarse, mediante la creación de un espacio de tablas que sólo exista en determinados miembros.
- En un entorno de base de datos particionada, los datos se graban en las tablas de destino únicamente del miembro en el que las entradas se han desalojado de la antememoria de paquetes de base de datos.
- En un entorno de base de datos particionada, si alguna tabla de destino no reside en un miembro, pero otras tablas de destino sí residen en ese mismo miembro, sólo se registrarán los datos para las tablas de sucesos sin formato de destino que residan en ese miembro.
- La sentencia FLUSH EVENT MONITOR no es aplicable a este supervisor de sucesos y no se aplicará cuando se emita contra el mismo.
- Después de haberse creado el supervisor de sucesos de antememoria de paquete, las opciones de filtro y de control no pueden cambiarse ni alterarse. Para cambiar las opciones de filtro y de control, el supervisor de sucesos debe desactivarse, descartarse y, a continuación, volver a crearse con las nuevas opciones de filtro y de control.

Utilización de espacios de tablas grandes para mejorar el rendimiento

Los datos de sucesos se insertan en la tabla de sucesos sin formato en una columna de datos BLOB en línea. Normalmente, los datos BLOB se almacenan en un espacio de tablas LOB independiente y pueden experimentar como resultado una actividad general adicional del rendimiento. Cuando los datos BLOB se colocan en línea en la página de datos de la tabla base, no experimentan dicha actividad general. El gestor de bases de datos DB2 colocará automáticamente en línea la parte de datos BLOB de una tabla de sucesos sin formato si el tamaño de los datos BLOB es inferior al tamaño de página del espacio de tablas menos el prefijo del registro. Por lo tanto, para mejorar la eficacia y el rendimiento de las aplicaciones, se recomienda crear el supervisor de sucesos en un espacio de tablas lo más grande posible, en un espacio de tablas de hasta 32 KB inclusive, y la agrupación de almacenamientos intermedios asociada.

Establecimiento en línea de los registros de antememoria de paquete

Para el supervisor de sucesos de antememoria de paquete, el tamaño de los elementos de supervisor **stmt_text**, **comp_env_desc** y **section_env** determinará si el registro de la antememoria de paquete estará o no en línea. Si el total de estos campos excede el tamaño del espacio de tablas, el registro no estará en línea.

Sentencia CREATE EVENT MONITOR (antememoria de paquete)

Determinación de si EVENT_DATA está en línea

Utilice las funciones ADMIN_IS_INLINED y ADMIN_EST_INLINE_LENGTH para determinar si el registro está en línea y para obtener una estimación de la longitud en línea que se necesita.

Restricciones

- Durante la desactivación de la base de datos, el supervisor de sucesos de antememoria de paquete no recopilará las entradas desalojadas.

Ejemplos

- *Ejemplo 1:* en este ejemplo se crea un supervisor de sucesos de antememoria de paquete denominado CACHEEVMON que recopilará los datos relacionados con sucesos de desalojo de sección de antememoria de paquete y grabará los datos en las tablas.

```
CREATE EVENT MONITOR CACHEEVMON
FOR PACKAGE CACHE
WRITE TO TABLE
```

Este supervisor de sucesos graba su salida en las tablas siguientes:

```
PKGCACHE_CACHEEVMON
PKGCACHE_METRICS_CACHEEVMON
PKGCACHE_STMT_ARGS
CONTROL_CACHEEVMON
```

- *Ejemplo 2:* en este ejemplo se crea un supervisor de sucesos de antememoria de paquete denominado CACHESTMTEVMON que recopilará los datos relacionados con sucesos de desalojo de sección de antememoria de paquete y los almacenará en la tabla de sucesos sin formato ALAN.STMTEVENTS.

```
CREATE EVENT MONITOR CACHESTMTEVMON
FOR PACKAGE CACHE
WRITE TO UNFORMATTED EVENT TABLE (TABLE ALAN.STMTEVENTS)
```

- *Ejemplo 3:* en este ejemplo se crea un supervisor de sucesos de antememoria de paquete denominado CACHESTMTEVMON que recopilará los datos relacionados con sucesos de desalojo de sección de antememoria de paquete y los almacenará en la tabla de sucesos sin formato ALAN.STMTEVENTS, en el espacio de tablas APPSPACE. El supervisor de sucesos se desactivará cuando el espacio de tablas esté un 85% lleno.

```
CREATE EVENT MONITOR CACHESTMTEVMON
FOR PACKAGE CACHE
WRITE TO UNFORMATTED EVENT TABLE
(TABLE ALAN.STMTEVENTS IN APPSPACE PCTDEACTIVATE 85)
```

CREATE EVENT MONITOR (estadísticas)

La sentencia CREATE EVENT MONITOR (estadísticas) define un supervisor que registrará sucesos estadísticos que se produzcan cuando se utilice la base de datos. La definición del supervisor de sucesos estadísticos especifica también el lugar donde la base de datos debe registrar los sucesos.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM
- Autorización WLMADM

Sintaxis

►► CREATE EVENT MONITOR *nombre-supervisor-sucesos* FOR STATISTICS

► WRITE TO

TABLE		info-tabla-sucesos-con-formato		?
PIPE		<i>nombre-conexión</i>		
FILE		<i>nombre-vía-acceso</i>		opciones-archivo

►

AUTOSTART		?	ON MEMBER		<i>número-miembro</i>		?	LOCAL		?
MANUALSTART										

info-tabla-sucesos-con-formato:

| ?

	,		
	grupo-evm		
	(opciones-tabla-destino
)		

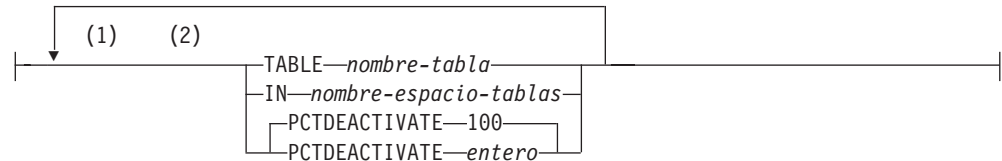
 | ?

►

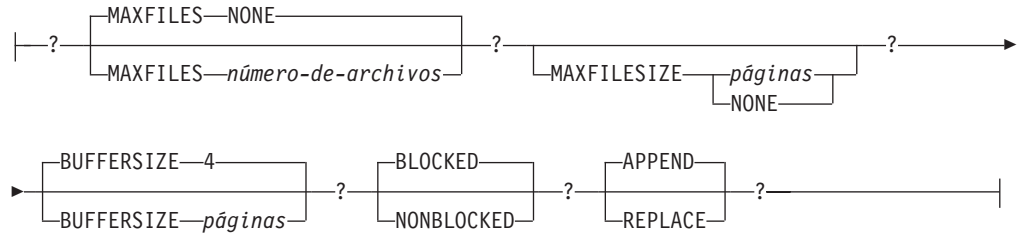
BUFFERSIZE	—	4		?	BLOCKED		?
BUFFERSIZE	—	<i>páginas</i>			NONBLOCKED		

opciones-tabla-destino:

CREATE EVENT MONITOR (estadísticas)



opciones-archivo:



Notas:

- 1 Cada cláusula sólo puede especificarse una vez.
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

STATISTICS

Especifica que el supervisor de sucesos registra un suceso de clase de trabajo, carga de trabajo o clase de servicio:

- Cada *período* minutos, donde *período* es el valor del parámetro de configuración de base de datos `wlm_collect_int`
- Cuando se llama al procedimiento `wlm_collect_stats`

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-tabla-sucesos-con-formato

Define las tablas de destino para un supervisor de sucesos. Esta

CREATE EVENT MONITOR (estadísticas)

cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula info-grupo-evm, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-evm

Identifica el grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-evm
Estadísticas	<ul style="list-style-type: none">• QSTATS• SCSTATS• SCMETRICS• WCSTATS• WLSTATS• WLMETRICS• HISTOGRAMBIN• CONTROL

opciones-tabla-destino

Identifica la tabla de destino de un grupo.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'  
CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el

CREATE EVENT MONITOR (estadísticas)

parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

Si no se especifica un valor para *opciones-tabla-destino*, el proceso CREATE EVENT MONITOR continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado.
- Se elige un espacio de tablas por omisión.
- PCTDEACTIVATE se establece por omisión en 100.

BUFFERSIZE *páginas*

Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Los supervisores de sucesos de tabla insertan todos los datos de un almacenamiento intermedio y emiten COMMIT cuando se ha procesado el almacenamiento intermedio. Cuanto más grandes sean los almacenamientos intermedios, mayor será el ámbito de confirmación utilizado por 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 un 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 por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esa pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

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.

CREATE EVENT MONITOR (estadísticas)

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 desecha 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 normas 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, están permitidos los nombres de conexión relativos y se tratan como nombres-vía relativos (consulte la descripción de *nombre-vía*). En Windows, no obstante, existe una sintaxis especial para un nombre de conexión y, como consecuencia de ello, se necesitan nombres de conexión 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 de 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-acceso

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; no obstante, la vía de acceso propiamente

dicha puede residir en otra partición de base de datos (por ejemplo, en un sistema 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 por el directorio raíz en AIX, o por un identificador de disco en Windows), se utilizará la vía de acceso especificada. En entornos que no sean DB2 pureScale, si se especifica una vía de acceso relativa (vía de acceso que no empieza por la raíz), se utilizará la vía de acceso relativa al directorio DB2EVENT del directorio de base de datos. En un entorno DB2 pureScale, si se especifica una vía de acceso relativa, se utilizará la vía de acceso relativa a la base de datos propietaria del directorio dentro del directorio de base de datos.

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-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. 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:

- Windows - 200 páginas de 4 K

CREATE EVENT MONITOR (estadísticas)

- 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 por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el valor del parámetro MAXFILESIZE, así como por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esta pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

Los supervisores de sucesos que graban sus datos en una conexión ("pipe") también tienen dos almacenamientos intermedios internos (no configurables) que tienen un tamaño de 1 página cada uno. Estos almacenamientos intermedios también se asignan a partir de la pila del supervisor (MON_HEAP). Para cada supervisor de sucesos activo que tiene un destino de conexión, aumente el tamaño de la pila de la base de datos en 2 páginas.

BLOCKED

Especifica que cada agente que genera un suceso debe esperar a que se grave 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 grave en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Los supervisores de sucesos NONBLOCKED

CREATE EVENT MONITOR (estadísticas)

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, este añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se reactiva, reanudará la grabación 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, este borrará todos los archivos de sucesos y empezará a grabar los datos en el archivo 00000000.evt.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que el miembro en el que se ejecuta el supervisor esté activado. Es el comportamiento por omisión del supervisor de sucesos estadísticos.

ON MEMBER *número-miembro*

Especifica el miembro en el que se debe ejecutar un supervisor de sucesos de archivo o conexión. Cuando el ámbito de supervisión se define como LOCAL, los datos se recopilan sólo en el miembro especificado. El componente de E/S se ejecutará físicamente en el miembro especificado y grabará los registros en el archivo o la conexión especificada. Cuando se habilita la función DB2 pureScale, el valor por omisión es -1. Si se especifica -1, permite que el componente de E/S se ejecute desde cualquier miembro activo. Además, en el caso de que el componente de E/S ya no pueda ejecutarse en un miembro determinado, el supervisor de sucesos se reiniciará con el componente de E/S ejecutándose en otro miembro activo disponible.

Esta cláusula no es válida para supervisores de sucesos de tabla. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se ejecutarán y grabarán los sucesos en todas las particiones de base de datos en las que se hayan definido espacios de tablas para tablas de destino. En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla registrarán los sucesos de todos los miembros activos.

Si no se especifica esta cláusula y la función DB2 pureScale no está habilitada, se utiliza el número de partición de base de datos conectado actualmente (para la aplicación). Si no se especifica esta cláusula y DB2 pureScale está habilitado, el componente de E/S puede ejecutarse en cualquier miembro conectado actualmente.

CREATE EVENT MONITOR (estadísticas)

LOCAL

El supervisor de sucesos sólo informa acerca del miembro que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Es el valor por omisión.

Esta cláusula es válida para los supervisores de conexión o de archivos. No es válida para supervisores de sucesos de tabla.

GLOBAL no es un ámbito válido para este tipo de supervisor de sucesos.

Normas

- El tipo de suceso STATISTICS no puede combinarse con ningún otro tipo de suceso de una definición de supervisor de sucesos concreta.

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. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Si el miembro en el que se debe ejecutar el supervisor de sucesos no está activo, la activación del supervisor de sucesos tiene lugar cuando se vuelve a activar ese miembro.
- Una vez activado un supervisor de sucesos, este se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva un miembro y ese miembro se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- Si crea el supervisor de sucesos de modo que los grupos de datos lógicos event_scstats o event_wlstats se incluyen en la salida del supervisor de sucesos, se informará acerca de las métricas en dos documentos XML, contenidos en la salida del supervisor de sucesos. Los elementos de métricas contenidos en el documento metrics volverán a establecerse en 0 tras cada intervalo de supervisión. Las medidas contenidas en details_xml no se restablece tras cada intervalo; en lugar de ello, siguen acumulándose hasta la siguiente activación de la base de datos.

Importante: A partir de la Versión 10.1 Fixpack 1, el documento XML details_xml ha quedado en desuso en el supervisor de sucesos estadísticos y podría eliminarse en un release futuro. Para obtener más información, consulte el tema Los informes de métricas en details_xml por el supervisor de sucesos estadísticos han quedado en desuso“Los informes de métricas en details_xml por el supervisor de sucesos estadísticos han quedado en desuso” en la publicación *Novedades en DB2 Versión 10.1*.

- **Supervisores de sucesos de grabación en tabla:** Notas generales:
 - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
 - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
 - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se

CREATE EVENT MONITOR (estadísticas)

habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.

- Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
- Deberán existir espacios de tablas para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tablas.
- Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada miembro de base de datos de la instancia, y cada uno de estos procesos sólo notifica datos al miembro en el que está ejecutándose.
- Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
 - LOG_STREAM_HEADER
 - LOG_HEADER
 - DB_HEADER (Los elementos db_name y db_path no se graban. El elemento conn_time se graba en CONTROL.)
- En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de miembro para supervisarlos mediante la creación de un espacio de tablas que exista sólo en algún miembro. En un entorno DB2 pureScale, los datos se grabarán desde cada miembro.

En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.
- Los usuarios pueden podar manualmente todas las tablas de destino.

Columnas de tabla:

- Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. Cualquier elemento del supervisor de sucesos que no tenga una columna de tabla de destino correspondiente se pasa por alto.
- Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.
- Los tipos de columnas que se utilizan para los elementos del supervisor corresponden a la correlación siguiente:

SQLM_TYPE_STRING	CHAR[n], VARCHAR[n] o CLOB(n) (Si los datos del registro del supervisor de sucesos exceden de <i>n</i> bytes, se truncan.)
SQLM_TYPE_U8BIT y SQLM_TYPE_8BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_16BIT y SQLM_TYPE_U16BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_32BIT y SQLM_TYPE_U32BIT	INTEGER o BIGINT
SQLM_TYPE_U64BIT y SQLM_TYPE_64BIT	BIGINT
sqlm_timestamp	TIMESTAMP
sqlm_time(tiempo transcurrido)	BIGINT

CREATE EVENT MONITOR (estadísticas)

```
sqlca:
sqlerrmc          VARCHAR[72]
sqlstate          CHAR[5]
sqlwarn           CHAR[11]
otros campos      INTEGER o BIGINT
```

- Las columnas se han definido para que sean NOT NULL.
- A diferencia de otras tablas de destino, las columnas de la tabla CONTROL no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

Nombre columna	Tipo datos	Nulos	Descripción
PARTITION_KEY	INTEGER	N	Clave de distribución (sólo base de datos particionada)
PARTITION_NUMBER	INTEGER	N	Número de partición de base de datos (sólo base de datos particionada)
EVMONNAME	VARCHAR(128)	N	Nombre de supervisor de sucesos
MESSAGE	VARCHAR(128)	N	Describe la naturaleza de la columna MESSAGE_TIME. Puede ser uno de los valores siguiente: <ul style="list-style-type: none">- FIRST_CONNECT (la hora de la primera conexión con la base de datos tras la activación)- EVMON_START (la hora en que se inició el supervisor de sucesos listado en EVMONNAME)- OVERFLOWS:n (indica que se han descartado n registros por desbordamiento del almacenamiento intermedio)- LAST_DROPPED_RECORD (última vez en que se ha producido un desbordamiento)
MESSAGE_TIME	TIMESTAMP	N	Indicación de fecha y hora

- En un entorno de base de datos particionada, la primera columna de cada tabla se denomina PARTITION_KEY, es NOT NULL y es de tipo INTEGER. Esta columna se utiliza como clave de distribución para la tabla. El valor de esta columna se elige para que cada proceso de supervisor de sucesos inserte datos en el miembro en el que se ejecute el proceso, es decir, las operaciones de inserción se llevan a cabo localmente en el miembro donde se está ejecutando el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se descarta una partición de base de datos y se realiza la redistribución de los datos, todos los datos de la partición de base de datos que se ha descartado se dirigirán a otra partición de base de datos en lugar de distribuirse equitativamente. Por lo tanto, antes de eliminar una partición de base de datos, considere la supresión de todas las filas de tabla de esa partición de base de datos.
- En un entorno de base de datos particionada, puede definirse una columna denominada PARTITION_NUMBER para cada tabla. Esta columna es NOT NULL y es de tipo INTEGER. Contiene el número de la partición de base de datos en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de base de datos no particionada.

Atributos de tabla:

CREATE EVENT MONITOR (estadísticas)

- Se utilizan atributos de tabla por omisión. Además de la clave de distribución (sólo bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
- En la tabla puede crearse índices.
- Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
- Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.

Activación del supervisor de sucesos:

- Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.
- En un entorno de base de datos particionada, el proceso de activación se produce en cada miembro de la instancia. En un miembro en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en un miembro si existe, como mínimo, una tabla de destino en esa partición de base de datos. Además, si no se encuentra alguna tabla de destino en una partición de base de datos, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.
- Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de base de datos particionada, si el espacio de tablas no reside en una partición de base de datos), la activación continúa y los datos que, de otro modo, se insertarían en esta tabla, se pasan por alto.
- El proceso de activación valida cada tabla de destino. Si la validación no se realiza satisfactoriamente, la activación del supervisor de sucesos no tiene lugar y se graban mensajes en el archivo de anotaciones cronológicas de administración.
- Durante la activación en un entorno de base de datos particionada, las filas de la tabla CONTROL de FIRST_CONNECT y de EVMON_START sólo se insertan en la partición de base de datos de catálogo. Para ello es necesario que el espacio de tablas de la tabla CONTROL exista en la partición de base de datos de catálogo. Si no existe en la partición de base de datos de catálogo, dichas inserciones no se realizan.
- En un entorno de base de datos particionada, si un miembro todavía no está activo al activarse el supervisor de sucesos de grabación en tabla, el supervisor de sucesos se activará la próxima vez que se active el miembro.

Tiempo de ejecución:

- El supervisor de sucesos se ejecuta con autorización DATAACCESS.
- Si, mientras existe un supervisor de sucesos activo, no se ejecuta satisfactoriamente una operación de inserción en una tabla de destino:
 - Los cambios no confirmados se retrotraen.
 - Se graba un mensaje en el archivo de anotaciones cronológicas de administración.
 - El supervisor de sucesos se desactiva.
- Si existe un supervisor de sucesos activo, ejecuta COMMIT localmente cuando ha terminado de procesar un almacenamiento intermedio del supervisor de sucesos.
- En un entorno que no sea una base de datos particionada o un entorno DB2 pureScale, todos los supervisores de sucesos de grabación en tabla se

CREATE EVENT MONITOR (estadísticas)

desactivarán cuando finalice la última aplicación (y la base de datos no se haya activado de forma explícita). En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla se desactivan en un miembro concreto cuando el miembro se detiene y se vuelve a activar cuando el miembro se reinicia. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se desactivan al desactivarse la partición de catálogo.

- La sentencia DROP EVENT MONITOR no descarta tablas de destino.
- Siempre que se active un supervisor de sucesos de grabación en tabla, obtendrá bloqueos de tabla IN en cada tabla de destino para impedir que se modifiquen mientras el supervisor de sucesos esté activo. Los bloqueos de tabla se mantienen en todas las tablas mientras el supervisor de sucesos está activo. Si en cualquiera de estas tablas de destino es necesario un acceso exclusivo (por ejemplo, cuando se va a ejecutar un programa de utilidad), primero desactive el supervisor de sucesos para liberar los bloqueos de tabla antes de intentar dicho acceso.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Se pueden especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Las comas se pueden utilizar para separar varias opciones en la cláusula *opciones-tabla-destino*

Ejemplo

Definir un supervisor de sucesos estadísticos denominado DB2STATISTICS

```
CREATE EVENT MONITOR DB2STATISTICS
FOR STATISTICS
WRITE TO TABLE
SCSTATS (TABLE SCSTATS_DB2STATISTICS
         IN USERSPACE1
         PCTDEACTIVATE 100),
WCSTATS (TABLE WCSTATS_DB2STATISTICS
         IN USERSPACE1
         PCTDEACTIVATE 100),
WLSTATS (TABLE WLSTATS_DB2STATISTICS
         IN USERSPACE1
         PCTDEACTIVATE 100),
QSTATS (TABLE QSTATS_DB2STATISTICS
        IN USERSPACE1
        PCTDEACTIVATE 100),
HISTOGRAMBIN (TABLE HISTOGRAMBIN_DB2STATISTICS
              IN USERSPACE1
              PCTDEACTIVATE 100),
CONTROL (TABLE CONTROL_DB2STATISTICS
         IN USERSPACE1
         PCTDEACTIVATE 100)
AUTOSTART;
```


CREATE EVENT MONITOR (violaciones de umbral)

La sentencia CREATE EVENT MONITOR (violaciones de umbral) define un supervisor que registrará sucesos de violaciones de umbral que se produzcan cuando se utilice la base de datos. La definición del supervisor de sucesos de violaciones de umbral especifica también el lugar donde la base de datos debe registrar los sucesos.

Invocación

A partir de la Versión 10 Fixpack 1 y fixpacks posteriores, el supervisor de sucesos de violación de umbral puede recopilar más información sobre la aplicación que ha violado el umbral. La adición de estos elementos de supervisor no afecta a los supervisores de sucesos de violación de umbral existentes, pero para recopilar información de aplicaciones adicional, los supervisores existentes deben descartarse y volver a crearse.

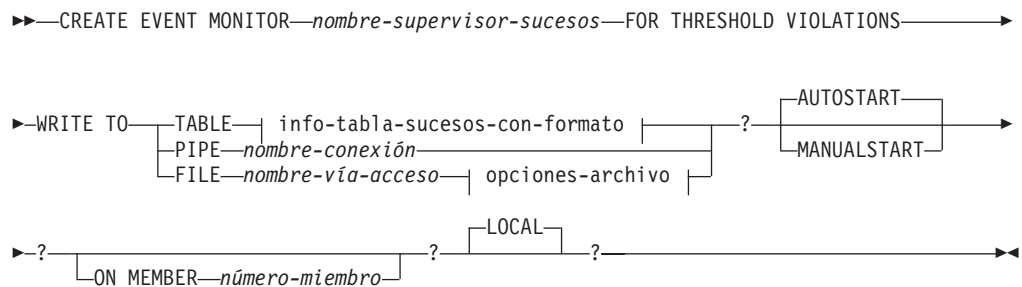
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

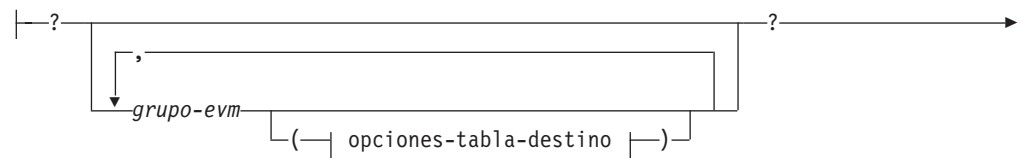
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM
- Autorización WLMADM

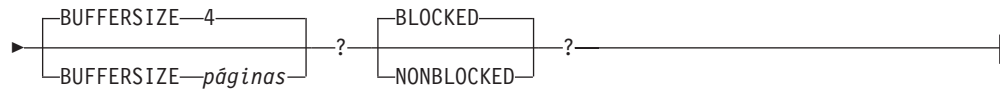
Sintaxis



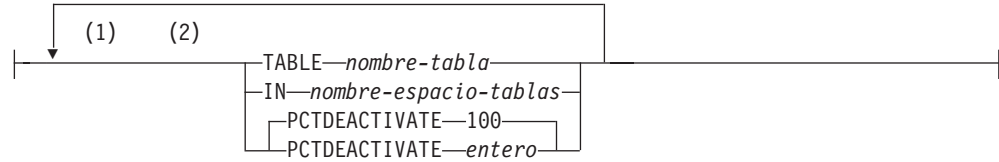
info-tabla-sucesos-con-formato:



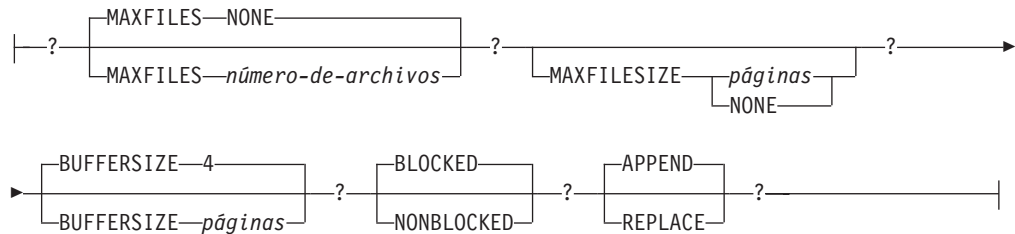
CREATE EVENT MONITOR (violaciones de umbral)



opciones-tabla-destino:



opciones-archivo:



Notas:

- 1 Cada cláusula sólo puede especificarse una vez.
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

THRESHOLD VIOLATIONS

Especifica que el supervisor de sucesos registra un suceso de violación de umbral cuando se viola un umbral. Dichos sucesos pueden registrarse en cualquier punto de la vida de una actividad y no sólo al completarse.

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla

CREATE EVENT MONITOR (violaciones de umbral)

que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-tabla-sucesos-con-formato

Define las tablas de destino para un supervisor de sucesos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula *info-grupo-evm*, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-evm

Identifica el grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-evm
Violaciones de umbral	<ul style="list-style-type: none">• THRESHOLDVIOLATIONS• CONTROL

opciones-tabla-destino

Identifica la tabla de destino de un grupo.

TABLE *nombre-tabla*

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial CURRENT SCHEMA. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *grupo-evm* y de *nombre-supervisor-sucesos* como se indica a continuación:

```
substring(grupo-evm CONCAT '_'  
CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

PCTDEACTIVATE *entero*

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el

CREATE EVENT MONITOR (violaciones de umbral)

valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

Si no se especifica un valor para *opciones-tabla-destino*, el proceso CREATE EVENT MONITOR continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado.
- Se elige un espacio de tablas por omisión.
- El parámetro PCTDEACTIVATE se establece por omisión en 100.

BUFFERSIZE *páginas*

Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Los supervisores de sucesos de tabla insertan todos los datos de un almacenamiento intermedio y emiten COMMIT cuando se ha procesado el almacenamiento intermedio. Cuanto más grandes sean los almacenamientos intermedios, mayor será el ámbito de confirmación utilizado por 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 un 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 por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esa pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

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.

PIPE

Especifica que el destino para los datos del supervisor de sucesos es una

CREATE EVENT MONITOR (violaciones de umbral)

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 desecha 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 normas 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, están permitidos los nombres de conexión relativos y se tratan como nombres-vía relativos (consulte la descripción de *nombre-vía*). En Windows, no obstante, existe una sintaxis especial para un nombre de conexión y, como consecuencia de ello, se necesitan nombres de conexión 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 de 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-acceso

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; no obstante, la vía de acceso propiamente dicha puede residir en otra partición de base de datos (por ejemplo, en

CREATE EVENT MONITOR (violaciones de umbral)

un sistema 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 por el directorio raíz en AIX, o por un identificador de disco en Windows), se utilizará la vía de acceso especificada. En entornos que no sean DB2 pureScale, si se especifica una vía de acceso relativa (vía de acceso que no empieza por la raíz), se utilizará la vía de acceso relativa al directorio DB2EVENT del directorio de base de datos. En un entorno DB2 pureScale, si se especifica una vía de acceso relativa, se utilizará la vía de acceso relativa a la base de datos propietaria del directorio dentro del directorio de base de datos.

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-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. 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:

- Windows - 200 páginas de 4 K
- UNIX - 1000 páginas de 4K

CREATE EVENT MONITOR (violaciones de umbral)

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 por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el valor del parámetro MAXFILESIZE, así como por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esta pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

Los supervisores de sucesos que graban sus datos en una conexión ("pipe") también tienen dos almacenamientos intermedios internos (no configurables) que tienen un tamaño de 1 página cada uno. Estos almacenamientos intermedios también se asignan a partir de la pila del supervisor (MON_HEAP). Para cada supervisor de sucesos activo que tiene un destino de conexión, aumente el tamaño de la pila 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

CREATE EVENT MONITOR (violaciones de umbral)

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, este añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se reactiva, reanudará la grabación 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, este borrará todos los archivos de sucesos y empezará a grabar los datos en el archivo 00000000.evt.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Es el comportamiento por omisión del supervisor de sucesos de violaciones de umbral.

ON MEMBER *número-miembro*

Especifica el miembro en el que se debe ejecutar un supervisor de sucesos de archivo o conexión. Cuando el ámbito de supervisión se define como LOCAL, los datos se recopilan sólo en el miembro. El componente de E/S se ejecutará físicamente en el miembro especificado y grabará los registros en el archivo o la conexión especificada. Cuando se habilita DB2 pureScale, el valor por omisión es -1. Si el valor es -1, permite que el componente de E/S se ejecute desde cualquier miembro activo. Además, en el caso de que el componente de E/S ya no pueda ejecutarse en un miembro determinado, el supervisor de sucesos se reiniciará con el componente de E/S ejecutándose en otro miembro activo disponible.

Esta cláusula no es válida para supervisores de sucesos de tabla. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se ejecutarán y grabarán los sucesos en todas las particiones de base de datos en las que se hayan definido espacios de tablas para tablas de destino. En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla registrarán los sucesos de todos los miembros activos.

Si no se especifica esta cláusula y DB2 pureScale no está habilitado, se utiliza el miembro conectado actualmente. Si no se especifica esta cláusula y DB2 pureScale está habilitado, el componente de E/S puede ejecutarse en cualquier miembro conectado actualmente.

CREATE EVENT MONITOR (violaciones de umbral)

LOCAL

El supervisor de sucesos sólo informa acerca del miembro que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Es el valor por omisión.

Esta cláusula es válida para los supervisores de conexión o de archivos. No es válida para supervisores de sucesos de tabla.

GLOBAL no es un ámbito válido para este tipo de supervisor de sucesos.

Normas

- El tipo de suceso THRESHOLD VIOLATIONS no puede combinarse con ningún otro tipo de suceso de una definición de supervisor de sucesos concreta.

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. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Si el miembro en el que se debe ejecutar el supervisor de sucesos no está activo, la activación del supervisor de sucesos tiene lugar cuando se vuelve a activar ese miembro.
- Una vez activado un supervisor de sucesos, este se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva un miembro y ese miembro se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- *Supervisores de sucesos de grabación en tabla:* Notas generales:
 - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
 - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
 - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.
 - Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
 - Deberán existir espacios de tablas para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tablas.
 - Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada miembro de base de datos de la instancia, y cada uno de estos procesos sólo notifica datos al miembro en el que está ejecutándose.

CREATE EVENT MONITOR (violaciones de umbral)

- Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
 - LOG_STREAM_HEADER
 - LOG_HEADER
 - DB_HEADER (Los elementos db_name y db_path no se graban. El elemento conn_time se graba en CONTROL.)
- En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base de datos con el fin de supervisarlas, creando un espacio de tablas que sólo exista en determinadas particiones de base de datos. En un entorno DB2 pureScale, los datos se grabarán desde cada miembro.

En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.
- Los usuarios pueden podar manualmente todas las tablas de destino.

Columnas de tabla:

- Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. Cualquier elemento del supervisor de sucesos que no tenga una columna de tabla de destino correspondiente se pasa por alto.
- Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.
- Los tipos de columnas que se utilizan para los elementos del supervisor corresponden a la correlación siguiente:

SQLM_TYPE_STRING	CHAR[n], VARCHAR[n] o CLOB(n) (Si los datos del registro del supervisor de sucesos exceden de <i>n</i> bytes, se truncan.)
SQLM_TYPE_U8BIT y SQLM_TYPE_8BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_16BIT y SQLM_TYPE_U16BIT	SMALLINT, INTEGER o BIGINT
SQLM_TYPE_32BIT y SQLM_TYPE_U32BIT	INTEGER o BIGINT
SQLM_TYPE_U64BIT y SQLM_TYPE_64BIT	BIGINT
sqlm_timestamp	TIMESTAMP
sqlm_time(tiempo transcurrido)	BIGINT
sqlca:	
sqlerrmc	VARCHAR[72]
sqlstate	CHAR[5]
sqlwarn	CHAR[11]
otros campos	INTEGER o BIGINT

- Las columnas se han definido para que sean NOT NULL.
- A diferencia de otras tablas de destino, las columnas de la tabla CONTROL no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

Nombre columna	Tipo datos	Nulos	Descripción
-----	-----	-----	-----
PARTITION_KEY	INTEGER	N	Clave de distribución (sólo base de datos particionada)
PARTITION_NUMBER	INTEGER	N	Número de partición de base de datos (sólo base de datos)

CREATE EVENT MONITOR (violaciones de umbral)

EVMONNAME	VARCHAR(128)	N	particionada)
MESSAGE	VARCHAR(128)	N	Nombre de supervisor de sucesos
			Describe la naturaleza de la columna MESSAGE_TIME. Puede ser uno de los valores siguiente:
			- FIRST_CONNECT (la hora de la primera conexión con la base de datos tras la activación)
			- EVMON_START (la hora en que se inició el supervisor de sucesos listado en EVMONNAME)
			- OVERFLOWS:n (indica que se han descartado n registros por desbordamiento del almacenamiento intermedio)
			- LAST_DROPPED_RECORD (última vez en que se ha producido un desbordamiento)
MESSAGE_TIME	TIMESTAMP	N	Indicación de fecha y hora

- En un entorno de base de datos particionada, la primera columna de cada tabla se denomina PARTITION_KEY, es NOT NULL y es de tipo INTEGER. Esta columna se utiliza como clave de distribución para la tabla. El valor de esta columna se elige de forma que cada proceso del supervisor de sucesos inserte datos en la partición de base de datos en la que está ejecutándose el proceso; es decir, se realizan operaciones de inserción localmente en la partición de base de datos en la que está ejecutándose el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se descarta una partición de base de datos y se realiza la redistribución de los datos, todos los datos de la partición de base de datos que se ha descartado se dirigirán a otra partición de base de datos en lugar de distribuirse equitativamente. Por lo tanto, antes de eliminar una partición de base de datos, considere la supresión de todas las filas de tabla de esa partición de base de datos.
- En un entorno de base de datos particionada, puede definirse una columna denominada PARTITION_NUMBER para cada tabla. Esta columna es NOT NULL y es de tipo INTEGER. Contiene el número de la partición de base de datos en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de base de datos no particionada.

Atributos de tabla:

- Se utilizan atributos de tabla por omisión. Además de la clave de distribución (sólo bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
- En la tabla puede crearse índices.
- Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
- Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.

Activación del supervisor de sucesos:

- Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.

CREATE EVENT MONITOR (violaciones de umbral)

- En un entorno de base de datos particionada, el proceso de activación se produce en cada partición de base de datos de la instancia. En una partición de base de datos en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en una partición de base de datos si existe, como mínimo, una tabla de destino en esa partición de base de datos. Además, si no se encuentra alguna tabla de destino en una partición de base de datos, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.
- Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de base de datos particionada, si el espacio de tablas no reside en una partición de base de datos), la activación continúa y los datos que, de otro modo, se insertarían en esta tabla, se pasan por alto.
- El proceso de activación valida cada tabla de destino. Si la validación no se realiza satisfactoriamente, la activación del supervisor de sucesos no tiene lugar y se graban mensajes en el archivo de anotaciones cronológicas de administración.
- Durante la activación en un entorno de base de datos particionada, las filas de la tabla CONTROL de FIRST_CONNECT y de EVMON_START sólo se insertan en la partición de base de datos de catálogo. Para ello es necesario que el espacio de tablas de la tabla CONTROL exista en la partición de base de datos de catálogo. Si no existe en la partición de base de datos de catálogo, dichas inserciones no se realizan.
- En un entorno de base de datos particionada, si una partición todavía no está activa al activarse el supervisor de sucesos de grabación en tabla, el supervisor de sucesos se activará la próxima vez que se active la partición.

Tiempo de ejecución:

- El supervisor de sucesos se ejecuta con autorización DATAACCESS.
- Si, mientras existe un supervisor de sucesos activo, no se ejecuta satisfactoriamente una operación de inserción en una tabla de destino:
 - Los cambios no confirmados se retrotraen.
 - Se graba un mensaje en el archivo de anotaciones cronológicas de administración.
 - El supervisor de sucesos se desactiva.
- Si existe un supervisor de sucesos activo, ejecuta COMMIT localmente cuando ha terminado de procesar un almacenamiento intermedio del supervisor de sucesos.
- En un entorno que no sea una base de datos particionada o un entorno DB2 pureScale, todos los supervisores de sucesos de grabación en tabla se desactivarán cuando finalice la última aplicación (y la base de datos no se haya activado de forma explícita). En un entorno DB2 pureScale, los supervisores de sucesos de grabación en tabla se desactivan en un miembro concreto cuando el miembro se detiene y se vuelve a activar cuando el miembro se reinicia. En un entorno de base de datos particionada, los supervisores de sucesos de grabación en tabla se desactivan al desactivarse la partición de catálogo.
- La sentencia DROP EVENT MONITOR no descarta tablas de destino.
- Siempre que se active un supervisor de sucesos de grabación en tabla, obtendrá bloqueos de tabla IN en cada tabla de destino para impedir que se modifiquen mientras el supervisor de sucesos esté activo. Los bloqueos de tabla se mantienen en todas las tablas mientras el supervisor de sucesos está activo. Si en cualquiera de estas tablas de destino es necesario un acceso

CREATE EVENT MONITOR (violaciones de umbral)

exclusivo (por ejemplo, cuando se va a ejecutar un programa de utilidad), primero desactive el supervisor de sucesos para liberar los bloqueos de tabla antes de intentar dicho acceso.

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Se pueden especificar DBPARTITIONNUM o NODE en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Las comas se pueden utilizar para separar varias opciones en la cláusula *opciones-tabla-destino*

Ejemplo

Definir un supervisor de sucesos de violación de umbral denominado DB2THRESHOLDVIOLATIONS

```
CREATE EVENT MONITOR DB2THRESHOLDVIOLATIONS
  FOR THRESHOLD VIOLATIONS
  WRITE TO TABLE
  THRESHOLDVIOLATIONS (TABLE THRESHOLDVIOLATIONS_DB2THRESHOLDVIOLATIONS
                        IN USERSPACE1
                        PCTDEACTIVATE 100),
  CONTROL (TABLE CONTROL_DB2THRESHOLDVIOLATIONS
           IN USERSPACE1
           PCTDEACTIVATE 100)
  AUTOSTART;
```

CREATE EVENT MONITOR (unidad de trabajo)

La sentencia CREATE EVENT MONITOR (unidad de trabajo) crea un supervisor de sucesos que anotará los sucesos cuando se complete una unidad de trabajo.

Invocación

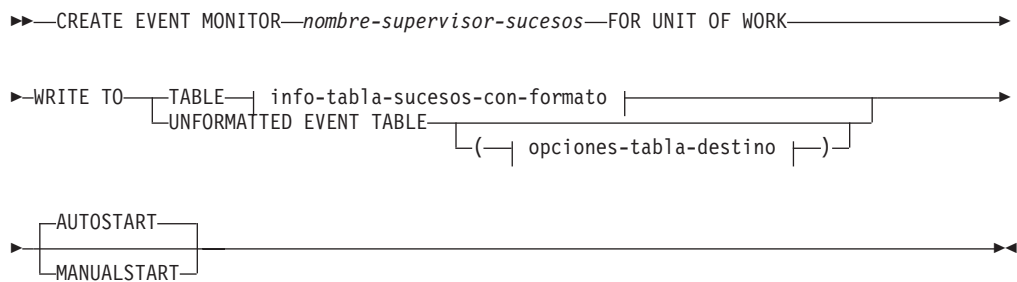
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

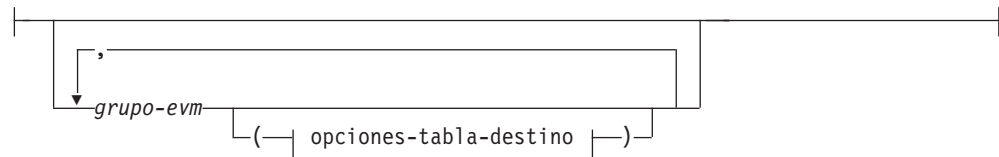
El ID de autorización de la sentencia debe tener una de las autorizaciones siguientes:

- Autorización DBADM
- Autorización SQLADM

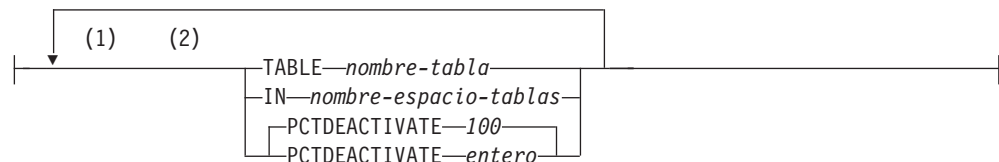
Sintaxis



info-tabla-sucesos-con-formato:



opciones-tabla-destino:



Notas:

- 1 Cada opción de tabla se puede especificar una vez como máximo (SQLSTATE 42613).
- 2 Las cláusulas pueden separarse mediante un espacio o una coma.

Descripción

nombre-supervisor-sucesos

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR

Especifica el tipo de suceso que se debe registrar.

UNIT OF WORK

Especifica que este supervisor de sucesos pasivo registrará un suceso siempre que se complete una unidad de trabajo (es decir, siempre que tenga lugar una operación de confirmación o retrotracción).

La creación del supervisor de sucesos de unidad de trabajo no indica que los datos de unidad de trabajo se recopilarán de manera inmediata. El suceso de unidad de trabajo real de interés se controla en el nivel de carga de trabajo.

WRITE TO

Especifica el destino para los datos.

TABLE

Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

info-tabla-sucesos-con-formato

Define las tablas de sucesos con formato de destino para un supervisor de sucesos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula *grupo-vm*, se registran todos los grupos del tipo de supervisor de sucesos.

grupo-vm

Identifica un grupo de datos lógicos para el que está definiéndose una tabla de destino. El valor depende del tipo de supervisor de sucesos, tal como se muestra en la tabla siguiente:

Tipo de supervisor de sucesos	Valor grupo-vm
Unidad de trabajo	<ul style="list-style-type: none"> • UOW • UOW_METRICS • UOW_PACKGE_LIST • UOW_EXECUTABLE_LIST • CONTROL

UNFORMATTED EVENT TABLE

Especifica que el destino del supervisor de sucesos es una tabla de sucesos sin formato. La tabla de sucesos sin formato se utiliza para almacenar los

CREATE EVENT MONITOR (unidad de trabajo)

datos de supervisor de sucesos de unidad de trabajo recopilados. Los datos se almacenan en su formato binario original dentro de una columna BLOB en línea. La columna BLOB puede contener varios registros binarios de tipos distintos. Los datos de la columna BLOB no tienen un formato legible y necesitan conversión, mediante la utilización de la herramienta **db2evmonfmt** basada en Java, la función de tabla `EVMON_FORMAT_UE_TO_XML` o el procedimiento `EVMON_FORMAT_UE_TO_TABLES`, en un formato que pueda utilizarse, como un documento XML o una tabla relacional.

opciones-tabla-destino

Identifica opciones para la tabla de destino. Si no se especifica un valor para **opciones-tabla-destino**, el proceso CREATE EVENT MONITOR continúa como se indica a continuación:

- Se utiliza un nombre de tabla derivado (tal y como se explica en la descripción de `TABLE nombre-tabla`).
- Un espacio de tablas por omisión se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.
- `PCTDEACTIVATE` se establece en 100.

TABLE nombre-tabla

Especifica el nombre de la tabla de destino. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el valor del registro especial `CURRENT SCHEMA`. Para una tabla de sucesos sin formato, si no se proporciona un nombre, el nombre no calificado es igual al `nombre-supervisor-sucesos`, es decir, a la tabla de sucesos sin formato se le asignará un nombre según el supervisor de sucesos. Para una tabla de sucesos con formato, si no se proporciona ningún nombre, el nombre no calificado se obtiene de `grupo-evm` y de `nombre-supervisor-sucesos` como se indica a continuación:

```
substring(grupo-evm CONCAT ' '
         CONCAT nombre-supervisor-sucesos,1,128)
```

IN nombre-espacio-tablas

Especifica el espacio de tablas en el que debe crearse la tabla. La sentencia CREATE EVENT MONITOR FOR UNIT OF WORK no crea espacios de tablas.

Si no se proporciona un nombre de espacio de tablas, el espacio de tablas se selecciona utilizando el mismo proceso que cuando se crea una tabla sin nombre de espacio de tablas utilizando CREATE TABLE.

Puesto que el tamaño de la página afecta a las longitudes de `INLINE LOB` utilizadas, considere la posibilidad de especificar un espacio de tablas con un tamaño de página lo más grande posible para mejorar el rendimiento de `INSERT` del supervisor de sucesos.

PCTDEACTIVATE entero

Si se está creando una tabla para el supervisor de sucesos en un almacenamiento automático (no temporal) o un espacio de tablas DMS, el parámetro `PCTDEACTIVATE` especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100, donde 100 significa que el supervisor de sucesos se desactiva cuando el espacio de tablas se llene por

CREATE EVENT MONITOR (unidad de trabajo)

completo. El valor por omisión que se presupone es 100 si no se especifica PCTDEACTIVATE. Esta opción se ignora para espacios de tablas SMS.

Importante: Si el espacio de tablas de destino tiene habilitado el ajuste de tamaño automático, establezca el parámetro PCTDEACTIVATE en 100. Si lo prefiere, puede omitir esta cláusula por completo para que se aplique el valor por omisión de 100. De lo contrario, puede que el supervisor de sucesos se desactive de forma inesperada si el espacio de tablas alcanza el umbral que especifica PCTDEACTIVTATE antes de que se vuelva a ajustar de forma automática el espacio de tablas.

AUTOSTART

Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Es el comportamiento por omisión del supervisor de sucesos de unidad de trabajo.

MANUALSTART

Especifica que el supervisor de sucesos debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia.

Notas

- La tabla se crea cuando la sentencia CREATE EVENT MONITOR FOR UNIT OF WORK se ejecuta, si no existe ya.
- Durante el proceso de CREATE EVENT MONITOR FOR UNIT OF WORK, si se encuentra una tabla que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR FOR UNIT OF WORK no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES. Si ya existe la tabla y no se ha definido para que la utilice otro supervisor de sucesos, no se crea ninguna tabla, se omiten los otros parámetros **opciones-tabla-destino** de tabla y continúa el proceso. Se enviará un aviso al programa de aplicación.
- Al descartar el supervisor de sucesos no se descartará ninguna tabla. Las tablas asociadas deben descartarse después de que se descarte el supervisor de sucesos.
- Los datos de suceso de bloqueo no se depurarán de forma automática desde las tablas de sucesos sin formato o desde las tablas comunes creadas por el supervisor de sucesos. Al utilizar el proceso EVMON_FORMAT_UE_TO_TABLES dispondrá de una opción para depurar datos desde las tablas UE. Para los supervisores de sucesos que graban tablas comunes, los datos de suceso se deben depurar de forma manual.
- Los datos de sucesos de tablas de sucesos sin formato se insertan en la tabla en una columna de datos BLOB en línea. Normalmente, los datos BLOB se almacenan en un espacio de tablas LOB independiente y pueden experimentar como resultado una actividad general adicional del rendimiento. Cuando los datos BLOB se colocan en línea en la página de datos de la tabla base, no experimentan dicha actividad general. El gestor de bases de datos DB2 colocará automáticamente en línea la parte de datos BLOB de una tabla de sucesos sin formato si el tamaño de los datos BLOB es inferior al tamaño de página del espacio de tablas menos el prefijo del registro. Por lo tanto, con el fin de mejorar la eficacia y el rendimiento de las aplicaciones, se recomienda crear el supervisor

CREATE EVENT MONITOR (unidad de trabajo)

de sucesos en un espacio de tablas lo más grande posible, en un espacio de tablas de hasta 32 KB inclusive, y la agrupación de almacenamientos intermedios asociada.

- Se creará un supervisor de sucesos de unidad de trabajo por base de datos únicamente, y no se crearán varios supervisores de sucesos de unidad de trabajo en la misma base de datos.
- En un entorno de base de datos particionada, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de sucesos sin formato de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base de datos seleccionables, creando un espacio de tablas que sólo exista en determinadas particiones de base de datos.
- En un entorno con múltiples miembros, los datos sólo se graban en las tablas de destino en las que se produzcan trabajos dentro de la unidad de trabajo.
- En un entorno de base de datos particionada, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.
- El supervisor de sucesos de unidad de trabajo no resulta afectado por el conmutador del supervisor de sucesos de unidad de trabajo. El conmutador del supervisor de sucesos de unidad de trabajo no cambia cuando se crea un supervisor de sucesos de unidad de trabajo, y los cambios en dicho conmutador no influyen en los contenidos del supervisor de sucesos de unidad de trabajo.
- La sentencia FLUSH EVENT MONITOR no es aplicable a este supervisor de sucesos y no se aplicará cuando se emita contra el mismo.
- La creación de un supervisor de sucesos de unidad de trabajo no provoca que se graben los sucesos en el supervisor de sucesos. El supervisor de sucesos de unidad de trabajo debe activarse con SET EVENT MONITOR STATE, y los datos de la unidad de trabajo deben recopilarse alterando la carga de trabajo correspondiente de modo que se especifique COLLECT UNIT OF WORK DATA o bien estableciendo el parámetro de configuración de base de datos **mon_uow_data** en un valor distinto de NONE.
- Al utilizar tablas de sucesos sin formato, cree el supervisor de sucesos de unidad de trabajo en un espacio de tablas con, como mínimo, 8 KB de espacio de página para asegurarse de que los datos de los sucesos están contenidos dentro de la columna BLOB en línea de la tabla de sucesos sin formato. Si la columna BLOB no está en línea, puede que el rendimiento de la grabación y lectura de los sucesos en la tabla de sucesos sin formato no sea eficiente.

Ejemplos

- *Ejemplo 1:* en este ejemplo se crea un supervisor de sucesos de unidad de trabajo UOWEVMON que recopilará datos para los sucesos de unidad de trabajo que tengan lugar en la base de datos de creación y grabará tablas de datos utilizando los nombres de tabla por omisión:

```
CREATE EVENT MONITOR UOWEVMON
FOR UNIT OF WORK
WRITE TO TABLE
```

Este supervisor de sucesos graba su salida en las tablas siguientes:

```
UOW_UOWEVMON
UOW_METRICS_UOWEVMON
UOW_PACKAGE_LIST_UOWEVMON
```

CREATE EVENT MONITOR (unidad de trabajo)

```
UOW_EXECUTABLE_LIST_UOWEVMON
UOW_CONTROL_UOWEVMON
```

Nota: El hecho de si las tablas de la lista de paquetes y la información de la lista ejecutable se llenan con datos depende de si se ha especificado que se deben recopilar los datos. Puede controlar la recopilación de estos datos utilizando los parámetros de configuración `mon_uow_pkglis` o `mon_uow_execlis`, o bien la cláusula `COLLECT UNIT OF WORK DATA` apropiada en las sentencias `CREATE` o `ALTER WORKLOAD`.

- *Ejemplo 2:* en este ejemplo se crea un supervisor de sucesos de unidad de trabajo `UOWEVMON` que recopilará los sucesos de unidad de trabajo que tengan lugar en la base de datos de creación y los almacenará en la tabla de sucesos sin formato `GREG.UOWEVENTS`.

```
CREATE EVENT MONITOR UOWEVMON
FOR UNIT OF WORK
WRITE TO UNFORMATTED EVENT TABLE (TABLE GREG.UOWEVENTS)
```

- *Ejemplo 3:* en este ejemplo se crea un supervisor de sucesos de unidad de trabajo `UOWEVMON` que recopilará los sucesos de unidad de trabajo que tengan lugar en la base de datos de creación y los almacenará en la tabla de sucesos sin formato `GREG.UOWEVENTS` en el espacio de tablas `APPSPACE`. El supervisor de sucesos se desactivará cuando el espacio de tablas esté un 85% lleno.

```
CREATE EVENT MONITOR UOWEVMON
FOR UNIT OF WORK
WRITE TO UNFORMATTED EVENT TABLE
(TABLE GREG.UOWEVENTS IN APPSPACE PCTDEACTIVATE 85)
```

CREATE FUNCTION

La sentencia CREATE FUNCTION se utiliza para registrar o definir una función definida por el usuario o una plantilla de función en el servidor actual.

Existen cinco tipos diferentes de funciones que se pueden crear utilizando esta sentencia. Cada una de ellos se describe por separado.

- Escalar externo. 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.
- 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.
- 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.
- Derivada o plantilla. Una función fuente se implanta invocando otra función (incorporada, externa, SQL o fuente) que ya está registrada en la base de datos. Es posible crear una función parcial, denominada *función de plantilla*, que define qué tipos de valores van a devolverse, pero que no contiene 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 puede registrarse solamente en un servidor de aplicaciones que esté designado como servidor federado.
- Escalar, tabla o fila de SQL. 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.

La sentencia CREATE FUNCTION se puede enviar en formato ofuscado. En una sentencia ofuscada, solamente son legibles el nombre de la función y sus parámetros. El resto de la sentencia se codifica de tal manera que no es legible, pero que el servidor de bases de datos puede descodificar. Las sentencias ofuscadas se pueden generar mediante una llamada a la función DBMS_DDL.WRAP.

CREATE FUNCTION (escalar externa)

La sentencia CREATE FUNCTION (escalar externa) se utiliza para registrar una función escalar externa definida por el usuario en el servidor actual. 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 se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, una de las autorizaciones siguientes:
 - Autorización IMPLICIT_SCHEMA en 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 hace referencia a un esquema existente
- Autorización DBADM

Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE FUNCTION no se tienen en cuenta.

Para crear una función no delimitada, los privilegios del ID de autorización de la sentencia debe incluir, como mínimo, una de las autorizaciones siguientes:

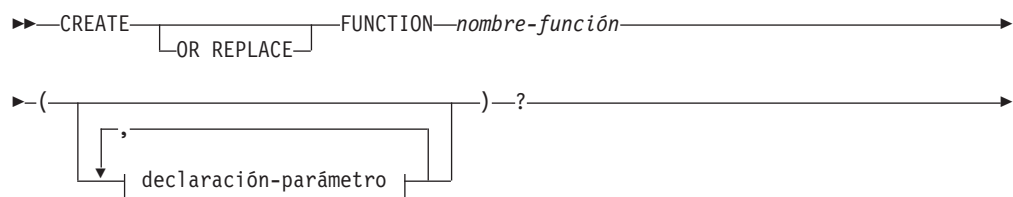
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

Para crear una función delimitada, no se necesitan autorizaciones ni privilegios adicionales.

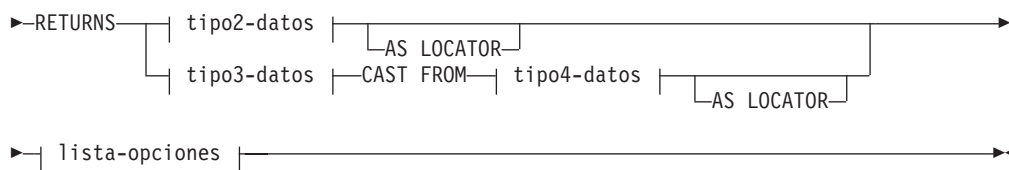
Para sustituir una función existente, el ID de autorización de la sentencia debe ser el propietario de la función existente (SQLSTATE 42501).

Si se especifica la opción SECURED, el ID de autorización de la sentencia debe incluir la autorización SECADM o CREATE_SECURE_OBJECT (SQLSTATE 42501).

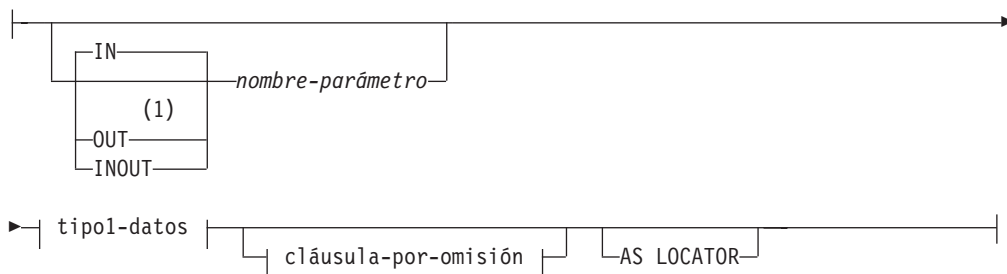
Sintaxis



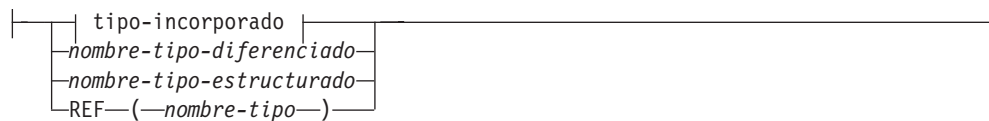
CREATE FUNCTION (escalar externa)



declaración-parámetro:

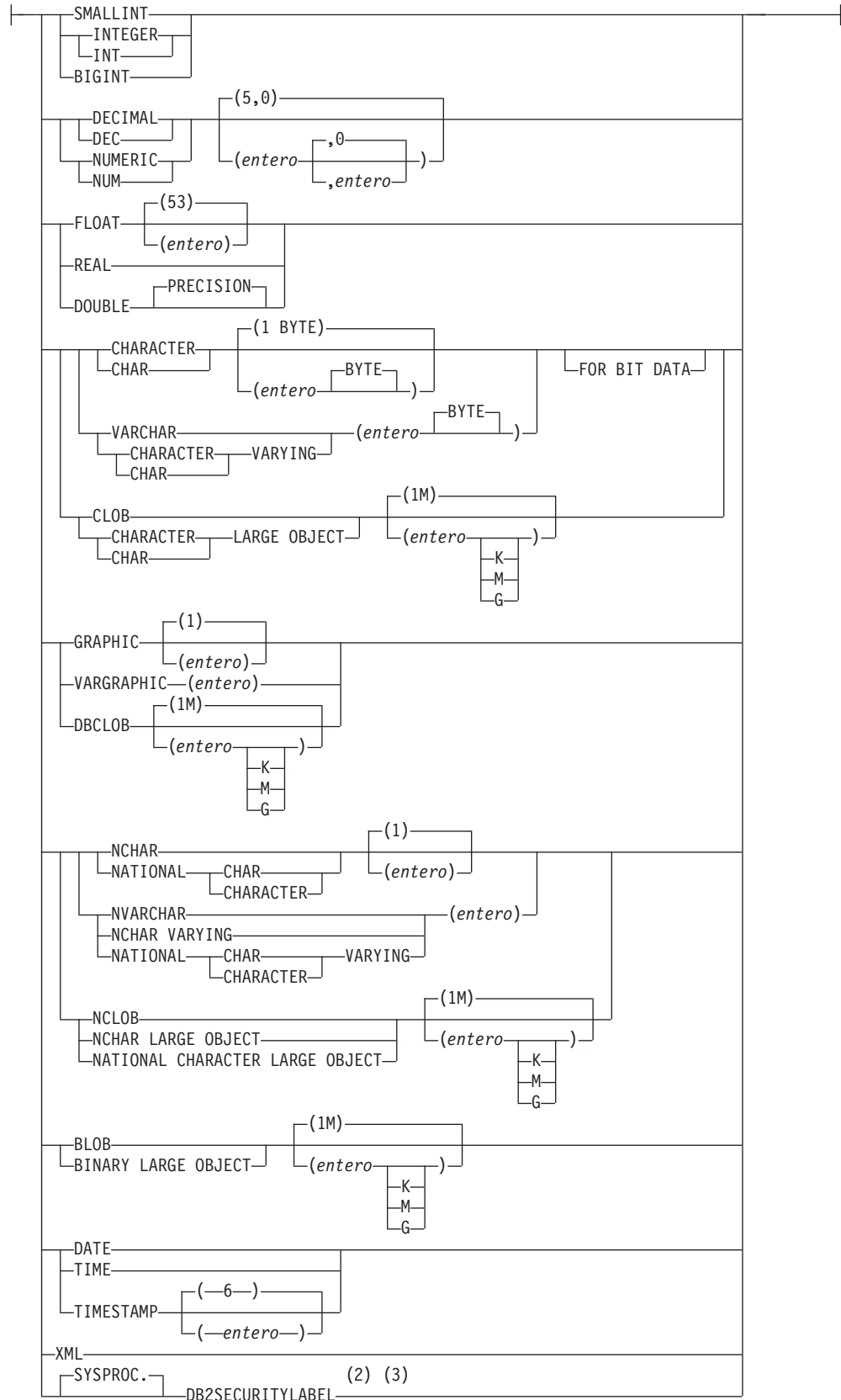


tipo1-datos, tipo2-datos, tipo3-datos, tipo4-datos:



tipo-incorporado:

CREATE FUNCTION (escalar externa)

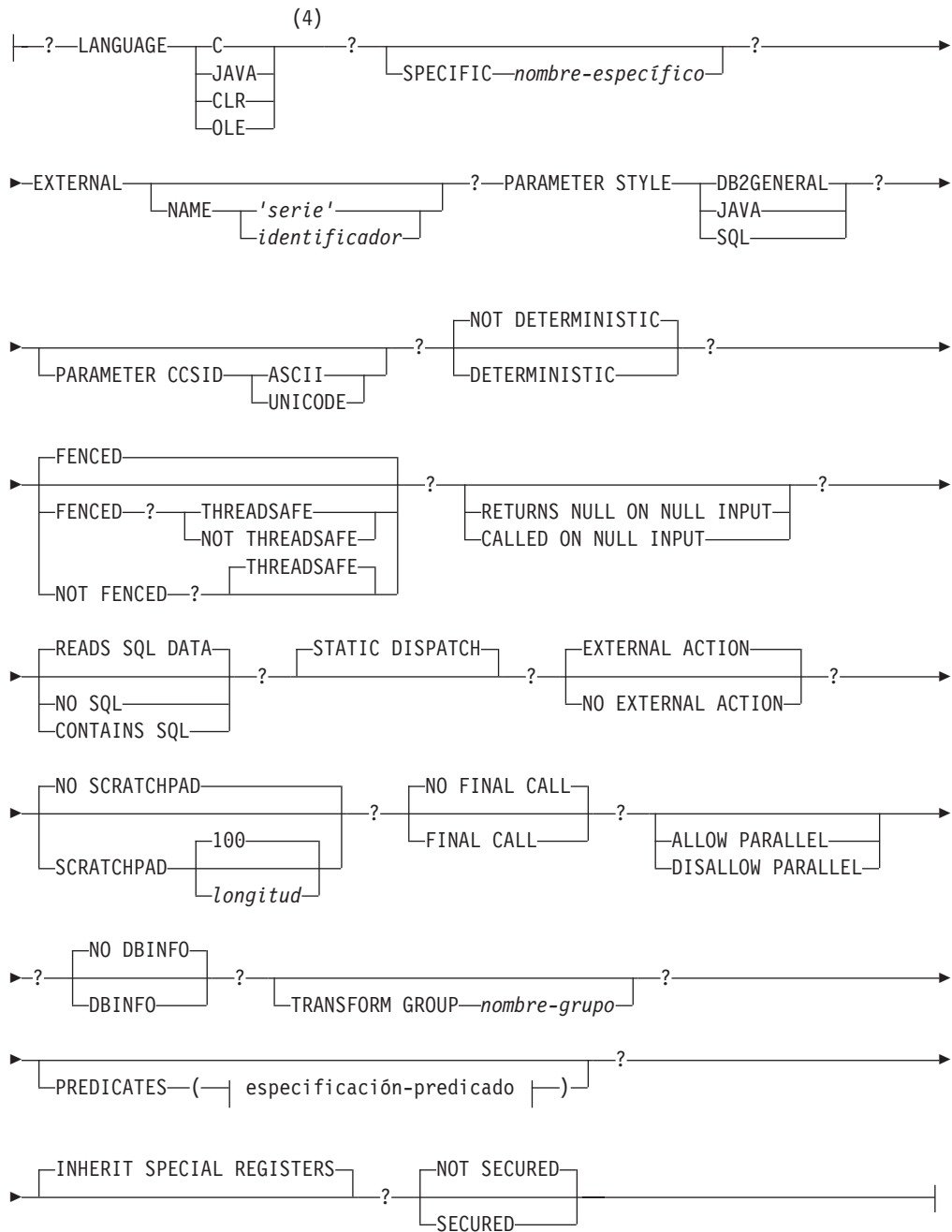


CREATE FUNCTION (escalar externa)

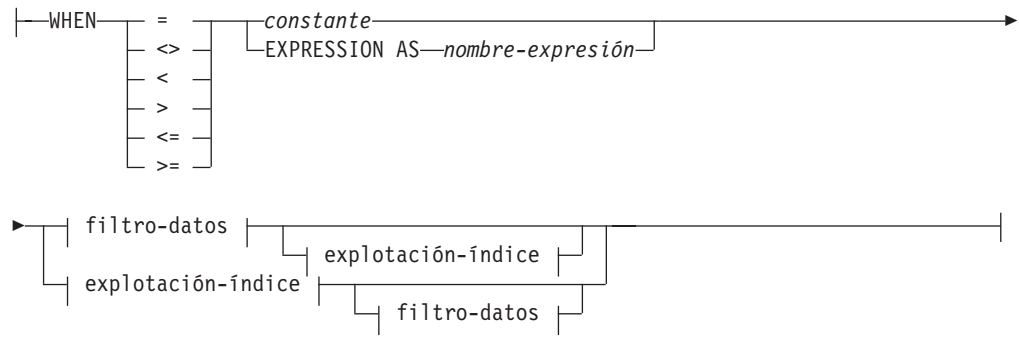
cláusula-por-omisión:



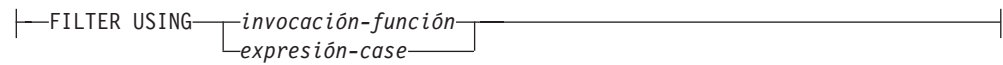
lista-opciones:



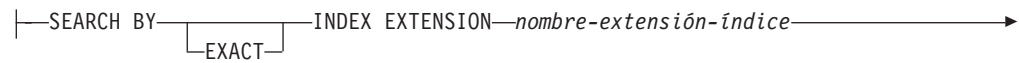
especificación-predicado:



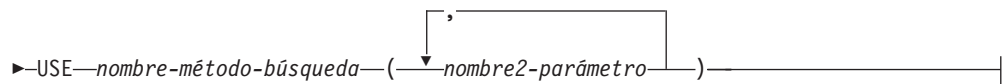
filtro-datos:



explotación-índice:



norma-explotación:



Notas:

- 1 OUT e INOUT sólo son válidos si la función tiene LANGUAGE C.
- 2 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 3 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.
- 4 También se da soporte a LANGUAGE SQL.

Descripción

OR REPLACE

Especifica que se debe sustituir la definición de la función si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre la función no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para la función en el servidor actual. Para sustituir una función ya existente, el nombre específico y el nombre de función de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de función de la antigua definición, o la signatura de la nueva definición debe coincidir con la signatura de la antigua definición. De lo contrario, se creará una nueva función.

Si se hace referencia a la función en la definición de un permiso de fila o una máscara de columna, la función no se puede sustituir (SQLSTATE 42893).

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. El formato no calificado de un *nombre-función* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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 de dos partes, el *nombre-esquema* no puede empezar por 'SYS'. De lo contrario, se genera un error (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función*. Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación. Cualquier incumplimiento de esta norma 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 agregada, se corre el riesgo de provocar errores en las sentencias del SQL dinámico o al volver a enlazar las

CREATE FUNCTION (escalar externa)

aplicaciones del SQL estático; es posible que la aplicación dé un resultado incorrecto, o incluso peor, que parezca que se ejecuta satisfactoriamente y genere un resultado diferente.

(declaración-parámetro,...)

Identifica el número de parámetros de entrada de la función y especifica la modalidad, el nombre, el tipo de datos y el valor por omisión opcional de cada parámetro. En la lista debe especificarse una única entrada por cada parámetro que la función espera recibir. Puede especificarse un máximo de 90 parámetros (SQLSTATE 54023).

Puede registrar una función que no tenga ningún parámetro; todavía será necesario codificar los paréntesis, sin indicar ningún tipo de datos. 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 consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que DECIMAL(11,2) y DECIMAL (4,3). Se considera que la especificación de un tipo diferenciado con tipo no firme es del mismo tipo de datos que el tipo de fuente del tipo diferenciado. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. 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 devuelve un error (SQLSTATE 42723).

IN | OUT | INOUT

Especifica la modalidad del parámetro. Si la función devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian. El valor por omisión es IN.

IN Identifica el parámetro como un parámetro de entrada para la función. Los cambios que se realicen en el parámetro dentro de la función no estarán disponibles para el contexto de invocación cuando se devuelva el control.

OUT

Identifica el parámetro como un parámetro de salida para la función.

La función debe definirse con LANGUAGE C (SQLSTATE 42613).

A la función sólo puede hacerse referencia en la parte derecha de una sentencia de asignación que se encuentre en una sentencia de SQL compuesto (compilado), y la referencia de función no puede formar parte de una expresión (SQLSTATE 42887).

INOUT

Identifica el parámetro como parámetro de entrada y de salida para la función.

La función debe definirse con LANGUAGE C (SQLSTATE 42613).

A la función sólo puede hacerse referencia en la parte derecha de una sentencia de asignación que se encuentre en una sentencia de SQL compuesto (compilado), y la referencia de función no puede formar parte de una expresión (SQLSTATE 42887).

nombre-parámetro

Especifica un nombre opcional para el parámetro. Los nombres de parámetro son necesarios para hacer referencia a los parámetros de una

CREATE FUNCTION (escalar externa)

función en la cláusula de *explotación-índice* de una especificación de predicado. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo-datos1

Especifica el tipo de datos del parámetro. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado, un tipo estructurado o un tipo de referencia. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Algunos tipos de datos no se soportan en todos los idiomas. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal, consulte "Tipos de datos que se correlacionan con tipos de datos de SQL en aplicaciones SQL incorporadas".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- DECFLOAT no es válido con LANGUAGE C, COBOL, CLR, JAVA y OLE (SQLSTATE 42815).
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de una función es una versión serializada del valor XML que se pasa como parámetro en la llamada de función, los parámetros del tipo XML deben declararse mediante la sintaxis XML AS CLOB(*n*).
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- No se pueden especificar tipos array (SQLSTATE 42815).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

Para un tipo de referencia, el parámetro se puede especificar como REF(*nombre-tipo*) si el parámetro no tiene ámbito.

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase *cláusula-por-omisión* en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

CREATE FUNCTION (escalar externa)

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión en las situaciones siguientes:

- Para parámetros INOUT o OUT (SQLSTATE 42601)
- Para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB)
- Para un parámetro de una definición de función donde también se ha especificado una cláusula PREDICATES (SQLSTATE 42613)

AS LOCATOR

Especifica que se pasa un localizador para el valor del parámetro a la función en lugar del valor real. Especifique AS LOCATOR sólo para parámetros con un tipo de datos LOB o un tipo diferenciado basado en un tipo de datos LOB (SQLSTATE 42601). El hecho de pasar localizadores en lugar de valores puede hacer que se pasen menos bytes a la función, especialmente cuando el valor del parámetro es muy grande.

La cláusula AS LOCATOR no tiene ningún efecto para determinar si los tipos de datos se pueden promocionar, ni afecta a la signatura de función, que se utiliza en la resolución de función.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS

Esta cláusula obligatoria identifica el resultado de la función.

tipo2-datos

Especifica el tipo de datos de la salida.

En este caso, se aplican exactamente las mismas consideraciones que para los parámetros de funciones externas descritos anteriormente en *tipo1-datos* para los 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 debe pasar un localizador de LOB de la UDF en lugar del valor real.

tipo3-datos **CAST FROM** *tipo4-datos*

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 genera un error (SQLSTATE 42880).

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 con parámetros especificados para *tipo-datos3*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo,

CREATE FUNCTION (escalar externa)

puede utilizarse VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Los tipos diferenciados, los tipos array y los tipos estructurados no son válidos como tipo especificado en *tipo4-datos* (SQLSTATE 42815).

La operación de conversión también está sujeta a comprobaciones durante la ejecución que podrían dar como resultado 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 debe devolver un localizador de LOB de una UDF en lugar del valor actual.

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 descartar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de 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: SQLaammddhhmmsxxx.

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 da por supuesto "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 tipo serie con un máximo de 254 bytes. 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. Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función de dentro de la biblioteca). No obstante, cuando la función se utiliza en una sentencia de SQL, la biblioteca y la función que está

CREATE FUNCTION (escalar externa)

dentro de la biblioteca deben existir y debe poder accederse a éstas desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

La *serie* se puede especificar de la forma siguiente:

```
»—'—id-biblioteca—id-vía-acceso-absoluta—!id-función—'—«
```

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

id-biblioteca

Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

- En los sistemas UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.
- En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

id-vía-acceso-absoluta

Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, en los sistemas UNIX, /u/jchui/mylib/myfunc haría que el gestor de bases de datos busque la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id-función

Identifica el nombre del punto de entrada de la función que debe invocarse. El signo de admiración (!) sirve como delimitador entre el ID de biblioteca y el ID de función.

Por ejemplo, en un sistema UNIX, 'mymod!func8' indicaría al gestor de bases de datos que debe buscar la biblioteca \$inst_home_dir/sqllib/function/mymod y que debe utilizar el punto de entrada func8 que está dentro de esa biblioteca.

En los sistemas operativos Windows, 'mymod!func8' indicaría al gestor de bases de datos que debe cargar el archivo mymod.dll y que debe llamar a la función func8() en la biblioteca de enlace dinámico (DLL).

Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

El cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de base de datos.

- Para LANGUAGE JAVA:

CREATE FUNCTION (escalar externa)

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 deberá existir cuando se ejecute la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

La *serie* se puede especificar de la forma siguiente:

→ ' id-jar : id-clase . id-método ' →

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

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 'myPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../myPacks/UserFuncs/' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '...\myPacks\UserFuncs\'.

id-método

Identifica el nombre del método del objeto Java que se invocará.

- Para LANGUAGE CLR:

La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se ejecuta la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos; de lo contrario, se devolverá un error (SQLSTATE 42724).

Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.

La *serie* se puede especificar de la forma siguiente:

►► '—*ensamblaje*—:—*id-clase*—!—*id-método*—' ◀◀

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

ensamblaje

Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de instalación de DB2 (por ejemplo, c:\sqlib\function). Si el archivo reside en un subdirectorio del directorio function de la instalación, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instalación es c:\sqlib y el archivo de ensamblaje es c:\sqlib\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

id-clase

Especifica el nombre de la clase del ensamblaje proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método

Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

La *serie* especificada es el identificador de programa OLE (idprog) o identificador de clase (clsid) 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. No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al ejecutar la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

La *serie* se puede especificar de la forma siguiente:

►► '—*idprog*—!—*id-método*—' ◀◀
 └—*idcls*—┘

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

idprog

Identifica el identificador de programa del objeto OLE.

CREATE FUNCTION (escalar externa)

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:

```
{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnnn}
```

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 del método del objeto OLE que se debe invocar.

NAME *identificador*

Este *identificador* especificado es un identificador de 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 del 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 para la que 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 de una clase Java.
- CLR** Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En estos momentos, sólo se soporta LANGUAGE CLR en el caso de las funciones definidas por el usuario que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).
- 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 la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para las funciones definidas por el usuario que están almacenadas en DB2 para los sistemas operativos Windows. THREADSAFE no puede especificarse para las UDF que se han definido con LANGUAGE OLE (SQLSTATE 42613).

PARAMETER STYLE

Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2GENERAL

Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, que están definidas como método en una clase Java. 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 el envío de parámetros que se ajusta a la especificación para el lenguaje Java y las Rutinas SQLJ. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA, no se han especificado como parámetros tipos de datos estructurados y no se han especificado tipos de datos CLOB, BLOB o DBCLOB como tipos de retorno (SQLSTATE 429B8). Las funciones PARAMETER STYLE JAVA no dan soporte a las cláusulas FINAL CALL, SCRATCHPAD o DBINFO.

SQL

Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es una base de datos Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo gráfico, tipo XML ni tipos definidos por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, las funciones se pueden crear con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

CREATE FUNCTION (escalar externa)

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

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. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen 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 se considera que la función es "segura" o no para ejecutarse en un proceso o espacio de dirección del entorno operativo del gestor de bases de datos.

Si una función se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los 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.

PRECAUCIÓN:

El uso de NOT FENCED en funciones que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 disponen de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no pueden garantizar la integridad completa cuando se utilizan funciones NOT FENCED definidas por el usuario.

Para una función con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar una función definida por el usuario como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

No se pueden crear funciones LANGUAGE CLR definidas por el usuario al especificar la cláusula NOT FENCED (SQLSTATE 42601).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que la función es segura para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si la función se ha definido con un LANGUAGE distinto de OLE:

- Si la función se ha definido como THREADSAFE, el gestor de bases de datos puede invocar la función en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, una función no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se

CREATE FUNCTION (escalar externa)

incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Las funciones FENCED y NOT FENCED pueden ser THREADSAFE.

- Si la función se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará al mismo tiempo la función en el mismo proceso que otra rutina.

Para las funciones FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (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 alguno de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación. Si no se especifica esta cláusula, el valor por omisión es RETURNS NULL ON NULL INPUT, salvo cuando se especifica PARAMETER STYLE JAVA, en cuyo caso el valor por omisión es CALLED ON NULL INPUT.

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 argumentos son 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.

NO SQL, CONTAINS SQL, READS SQL DATA

Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL

Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA

Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

CREATE FUNCTION (escalar externa)

STATIC DISPATCH

Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION

Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada tarea paralela, en lugar de una nota para la función. Especifique la cláusula DISALLOW PARALLEL en el caso de las funciones que no funcionen bien con paralelismo.

NO EXTERNAL ACTION

Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

NO SCRATCHPAD o SCRATCHPAD *longitud*

Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Se recomienda especialmente que las funciones definidas por el usuario sean reentrantes, para que una memoria de trabajo proporcione un medio para que la función "guarde el estado" entre una llamada a 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: X'00'.
- 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 que se ha mostrado anteriormente. Si la función se ejecuta en varias particiones de base de datos, 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 forma similar, si la consulta se ejecuta con el paralelismo de intrapartición habilitado, pueden asignarse más de tres memorias de trabajo.

- 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.

FINAL CALL o NO 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.
 - 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 tarea paralela: Este caso se produce cuando la función la ejecutan tareas paralelas.
 - Fin de transacción o interrupción: Este caso se produce cuando no tiene lugar la finalización normal de la sentencia. Por ejemplo, cuando por alguna razón la lógica de una aplicación ignora el cierre del cursor. Durante este tipo de llamada de finalización, no puede emitirse ninguna sentencia de SQL, a excepción del cursor CLOSE (SQLSTATE 38505). Este tipo de llamada final se indica con un valor especial en el argumento de "tipo de llamada".

CREATE FUNCTION (escalar externa)

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 ningún argumento de "tipo de llamada" a la función externa, y no se realiza ninguna llamada final.

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 las funciones escalares deben poder paralelizarse, pero puede 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 de base de datos? 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 función externa debe estar en un directorio que sea accesible en todas las particiones de 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.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

CREATE FUNCTION (escalar externa)

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

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 - si se dan las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - si, y 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 actualiza o inserta; de otro modo, 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.
- 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.

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 de SQL que realicen una transformación adecuada entre el tipo estructurado y sus atributos de tipo incorporados.

PREDICATES

Define el filtrado o la utilización de la extensión de índice que se lleva a cabo

CREATE FUNCTION (escalar externa)

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). Si se especifica la cláusula PREDICATES y la base de datos no es Unicode, no se debe especificar PARAMETER CCSID UNICODE (SQLSTATE 42613).

WHEN *operador-comparación*

Especifica 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 de resultados.

invocación-función

Especifica una función de filtro que se puede utilizar para realizar un filtrado adicional de la tabla de resultados. 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 realizará 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; de lo contrario, se descartará.

Esta función también debe cumplir estas condiciones:

- No debe estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 42845)
- No debe tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- No debe incluir una subconsulta (SQLSTATE 428E4)
- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E4)

Si un argumento invoca otra función u otro método, estas normas también se aplican para la función o el método anidados. No obstante,

CREATE FUNCTION (escalar externa)

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.

El usuario que define la función debe disponer de privilegio EXECUTE para la función de filtro especificada.

La cláusula *invocación-función* no debe superar los 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

expresión-case

Especifica una expresión CASE para realizar un mayor filtrado de la tabla de resultados. 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 normas especificadas en FILTER USING *invocación-función*. Cualquier función o método al que se hace referencia en la *expresión-case* también debe seguir las cuatro normas indicadas en la *invocación-función*.

No se pueden utilizar subconsultas ni las expresiones XMLQUERY o XMLEXISTS en ningún lugar de la *expresión-case* (SQLSTATE 428E4).

La expresión case debe devolver un valor entero (SQLSTATE 428E4). Un valor de retorno de 1 en la expresión-resultado significa que se conserva la fila; de lo contrario, se elimina.

La cláusula *invocación-case* no debe superar los 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

explotación-índice

Define un conjunto de normas 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.

norma-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.

CREATE FUNCTION (escalar externa)

WHEN KEY (*nombre1-parámetro*)

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 fuente 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 debe darse dentro de la misma jerarquía de tipos en el caso de tipos estructurados.

NOT SECURED o SECURED

Especifica si la función se considera segura para el control de acceso a filas y columnas. El valor por omisión es NOT SECURED.

NOT SECURED

Indica que la función no se considera segura. Cuando se invoca la función, los argumentos de la función no deben hacer referencia a una columna para la que haya habilitada una máscara de columna y el control de acceso de nivel de columna esté activado para su tabla (SQLSTATE 428HA). Esta norma se aplica a las funciones definidas por el usuario no seguras que se invocan en algún punto de la sentencia.

SECURED

Indica que la función se considera segura. La función debe ser segura cuando se hace referencia a ella en un permiso de fila o una máscara de columna (SQLSTATE 428H8).

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 con parámetros, 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 de fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.

CREATE FUNCTION (escalar externa)

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas de promoción que afectarán a sus valores de entrada (consulte el apartado “Promoción de tipos de datos”). 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 normas 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.
- Una función y un método no pueden estar en una relación de alteración temporal (SQLSTATE 42745). Para obtener más información sobre la alteración temporal, consulte “CREATE TYPE (estructurada)”.
- 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).
- La creación de una función 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- En un entorno de base de datos particionada, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Si la función permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- Los parámetros de XML sólo están soportados en las funciones externas de LANGUAGE JAVA si se especifica la cláusula PARAMETER STYLE DB2GENERAL.
- **Restricciones de acceso a las tablas**

Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.
- **Establecimiento del valor por omisión:** los parámetros de una función que están definidos con un valor por omisión se establecen en su valor por omisión

CREATE FUNCTION (escalar externa)

cuando se invoca la función, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca la función.

- **Privilegios:** el definidor de una función siempre recibe el privilegio WITH GRANT OPTION de EXECUTE sobre la función, así como el derecho de descartarla función.
Cuando la función se utiliza en una sentencia de SQL, el usuario que define la función debe disponer de privilegio EXECUTE para cualquiera de los paquetes que la función utiliza.
- **Funciones EXTERNAL ACTION:** Si se invoca una función EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca la función variará en función del plan de acceso utilizado.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL
 - NOT VARIANT puede utilizarse en lugar de DETERMINISTIC, y VARIANT puede especificarse en lugar de NOT DETERMINISTIC
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT, y NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT

La sintaxis siguiente se acepta como comportamiento por omisión:

- ASUTIME NO LIMIT
 - NO COLLID
 - PROGRAM TYPE SUB
 - STAY RESIDENT NO
 - CCSID UNICODE en una base de datos Unicode
 - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
- **Creación de una función segura:** Por lo general, los usuarios con autorización SECADM no tienen privilegios para crear objetos de base de datos, como activadores y funciones. Suelen examinar los datos a los que accede una función, comprobar que son seguros y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya tenga los privilegios necesarios para crear una función segura definida por el usuario. Tras crear la función, revocarán la autorización CREATE_SECURE_OBJECT del propietario de la función.
El atributo SECURED se considera una aserción que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. El gestor de bases de datos presupone que ese procedimiento de auditoría de control está en vigor para todas las sentencias ALTER FUNCTION posteriores o cambios en los paquetes externos.
 - **Invocación de otras funciones definidas por el usuario en una función segura:** Si una función segura definida por el usuario invoca otras funciones definidas por el usuario, el gestor de bases de datos no valida si esas funciones definidas por el usuario anidadas tienen el atributo SECURED. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.

CREATE FUNCTION (escalar externa)

- *Sustitución de una función existente de manera que el atributo seguro cambia (de SECURED a NOT SECURED y viceversa):* Los paquetes y las sentencias de SQL colocadas dinámicamente en antememoria que dependen de la función pueden quedar invalidadas porque el atributo seguro afecta a la selección de la vía de acceso de las sentencias que implican tablas para las que se ha activado el control de acceso de nivel de fila o columna.

Ejemplos

- *Ejemplo 1:* Pellow registra la función CENTRE en su esquema PELLOW. Permitiremos que las palabras clave que se tomarán por omisión se encarguen de realizar tal acción y que el sistema proporcione un nombre específico de la función:

```
CREATE FUNCTION CENTRE (INT,FLOAT)
  RETURNS FLOAT
  EXTERNAL NAME 'mod!middle'
  LANGUAGE C
  PARAMETER STYLE SQL
  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 PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  SPECIFIC FOCUS92
  EXTERNAL NAME 'effects!focalpt'
  LANGUAGE C PARAMETER STYLE SQL
  DETERMINISTIC FENCED NOT NULL CALL NO SQL NO EXTERNAL ACTION
  SCRATCHPAD NO FINAL CALL
  DISALLOW PARALLEL
```

- *Ejemplo 3:* El ejemplo siguiente es el programa de función definida por el usuario en lenguaje C que se escribe para implementar la norma $output = 2 * input - 4$ que devuelve NULL si, y sólo si, la entrada es nula. Podría haberse escrito incluso de forma más sencilla (es decir, sin comprobación de nullos) si la sentencia CREATE FUNCTION hubiera utilizado NOT NULL CALL. La sentencia CREATE FUNCTION:

```
CREATE FUNCTION ntest1 (SMALLINT)
  RETURNS SMALLINT
  EXTERNAL NAME 'ntest1!nudft1'
  LANGUAGE C PARAMETER STYLE SQL
  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 norma:
if (input is null)
set output = null;
else
set output = 2 * input - 4;
```

CREATE FUNCTION (escalar externa)

```
*/
void SQL_API_FN nudft1
(short *input,      /* puntero al argumento de entrada */
short *output,     /* puntero al resultado */
short *input_ind, /* puntero a variable de indicador de entrada */
short *output_ind, /* puntero a variable de indicador de salida */
char sqlstate[6], /* sqlstate, permite término nulo */
char fname[28],  /* nombre función calificado al completo, termino nulo */
char finst[19],  /* nombre específico función, término nulo */
char msgtext[71]) /* almacenamiento 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;
}
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:* El siguiente ejemplo registra una UDF Java que devuelve la posición de la primera vocal de una serie. La UDF está escrita en Java, se debe ejecutar como delimitada 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 SQL
DETERMINISTIC
NOT FENCED
NO SQL
NO EXTERNAL ACTION
EXTERNAL NAME 'db2sefn!SDESpatialRelations'
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 withinExplRule(g2)
WHEN KEY(g2) USE withinExplRule(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.

CREATE FUNCTION (escalar externa)

- **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 este ejemplo, 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, la tupla 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 SQL
DETERMINISTIC
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 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. Puesto que sólo es significativo el tipo de datos de la expresión de la parte derecha (frente a la utilización de una constante específica), es más adecuado seleccionar la cláusula EXPRESSION en el DDL CREATE FUNCTION para especificar un carácter comodín como valor de comparación.

CREATE FUNCTION (escalar externa)

Como alternativa, también es válida la sentencia siguiente como predicado definido por el usuario:

```
SELECT T1.C1
FROM T1, T2
WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
```

Actualmente existe una restricción por la que sólo el lado derecho se trata como una expresión; el término del 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 un nombre para la expresión, que se pasa como argumento a la función productora de rangos.

CREATE FUNCTION (tabla externa)

La sentencia CREATE FUNCTION (Tabla externa) se utiliza para registrar una función de tabla externa definida por el usuario en el servidor actual.

En la cláusula FROM de una sentencia SELECT, puede utilizarse una *función de tabla* y esta devuelve una tabla a la sentencia SELECT de fila en fila.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, una de las autorizaciones siguientes:
 - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
 - El privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función
- Autorización DBADM

Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE FUNCTION no se tienen en cuenta.

Para crear una función no delimitada, los privilegios del ID de autorización de la sentencia debe incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

Para crear una función delimitada, no se necesitan autorizaciones ni privilegios adicionales.

Para sustituir una función existente, el ID de autorización de la sentencia debe ser el propietario de la función existente (SQLSTATE 42501).

Si se especifica la opción SECURED, el ID de autorización de la sentencia debe incluir la autorización SECADM o CREATE_SECURE_OBJECT (SQLSTATE 42501).

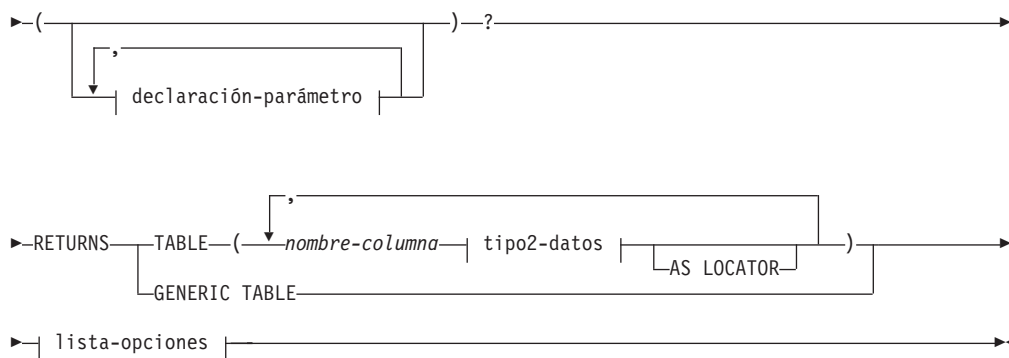
Sintaxis

```

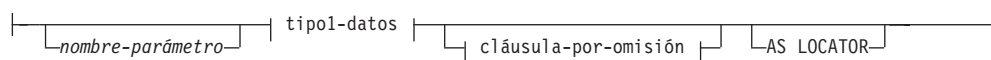
▶▶ CREATE [OR REPLACE] FUNCTION nombre-función

```

CREATE FUNCTION (tabla externa)



declaración-parámetro:

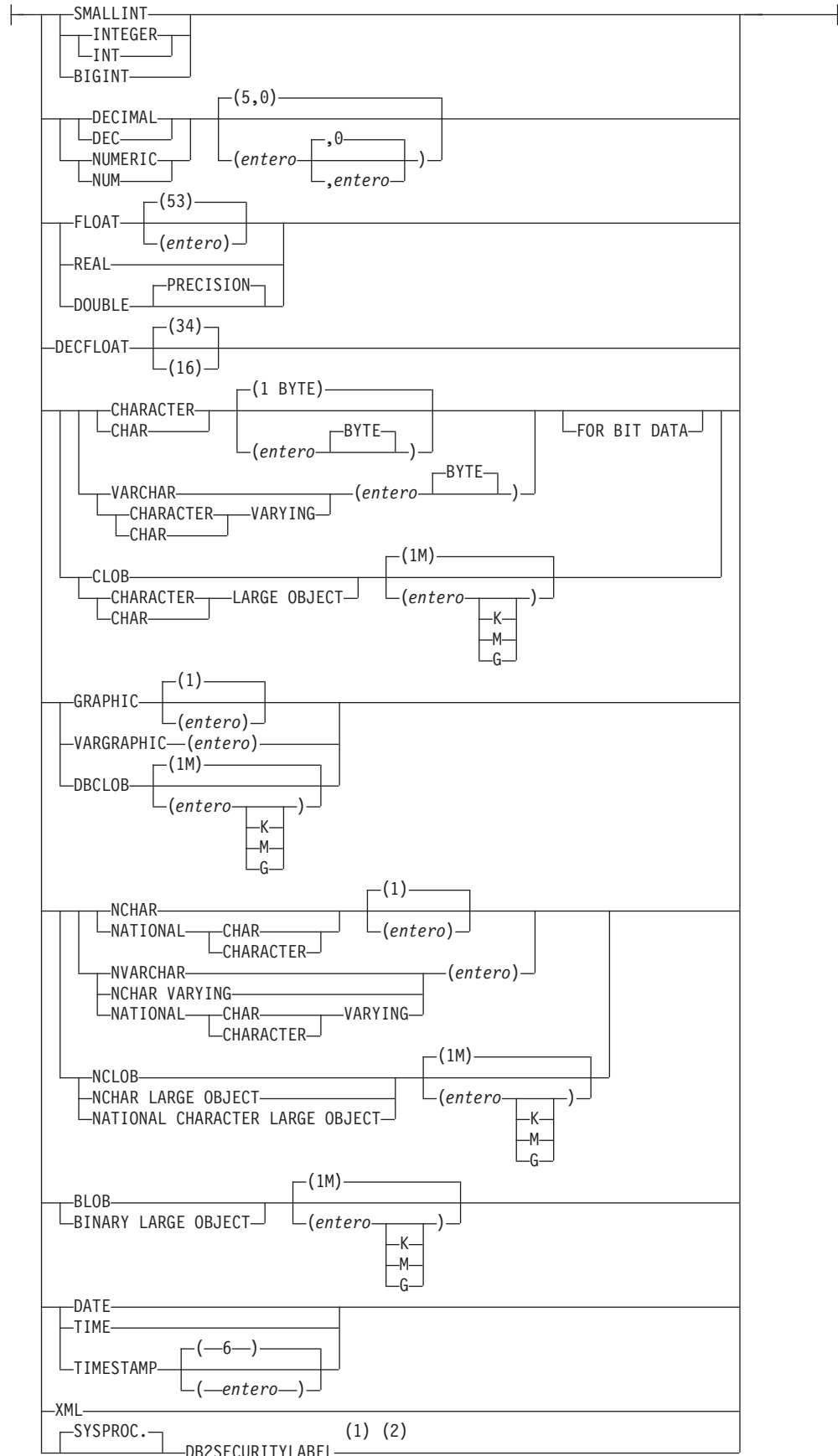


tipo1-datos, tipo2-datos:



tipo-incorporado:

CREATE FUNCTION (tabla externa)

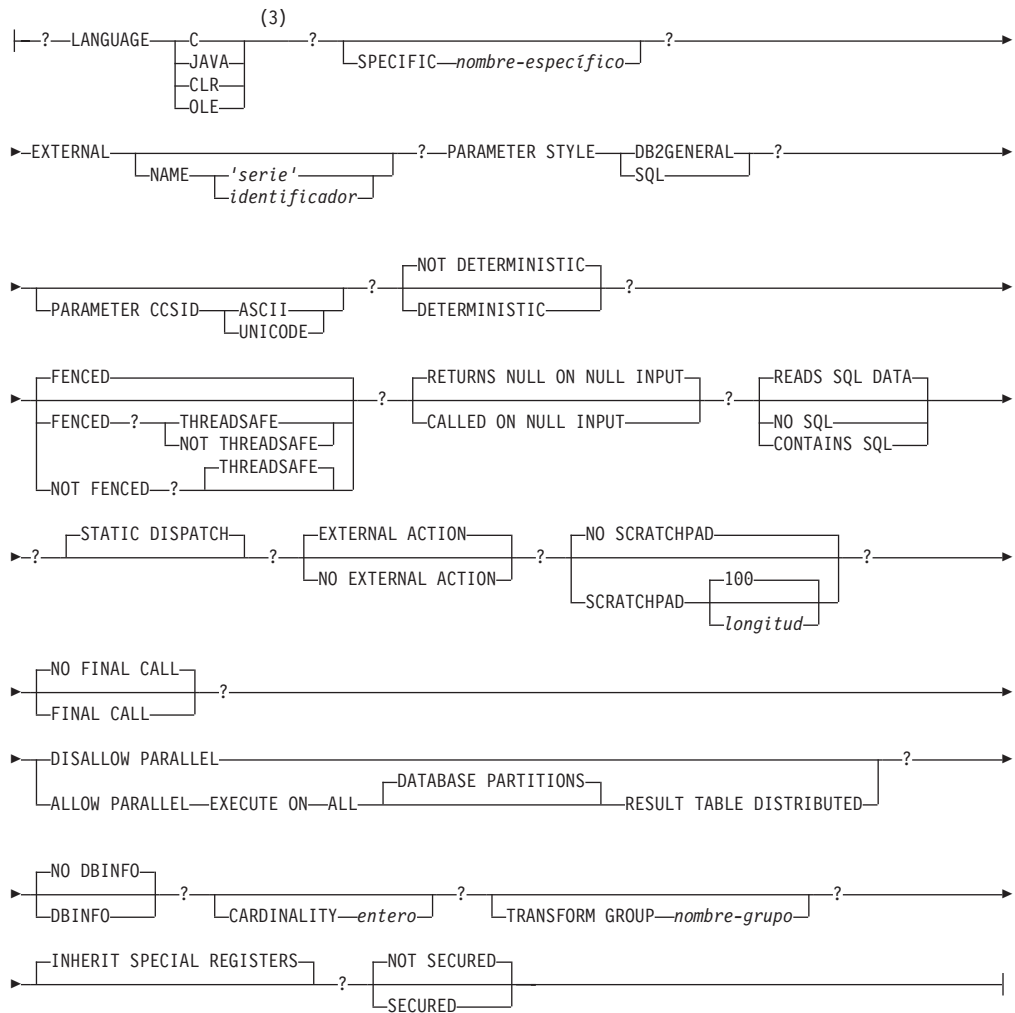


CREATE FUNCTION (tabla externa)

cláusula-por-omisión:



lista-opciones:



Notas:

- 1 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 2 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.
- 3 Para obtener información acerca de la creación de funciones de tabla externa LANGUAGE OLE DB, consulte "CREATE FUNCTION (Tabla externa OLE

DB)". Para obtener información sobre cómo crear funciones de tabla LANGUAGE SQL, consulte "CREATE FUNCTION (escalar de SQL, tabla o fila)".

Descripción

OR REPLACE

Especifica que se debe sustituir la definición de la función si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre la función no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para la función en el servidor actual. Para sustituir una función ya existente, el nombre específico y el nombre de función de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de función de la antigua definición, o la signatura de la nueva definición debe coincidir con la signatura de la antigua definición. De lo contrario, se creará una nueva función.

Si se hace referencia a la función en la definición de un permiso de fila o una máscara de columna, la función no se puede sustituir (SQLSTATE 42893).

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. El formato no calificado de un *nombre-función* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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 de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice 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.

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.

(declaración-parámetro,...)

Identifica el número de parámetros de entrada de la función y especifica el tipo de datos y el valor por omisión opcional de cada parámetro. En la lista debe

CREATE FUNCTION (tabla externa)

especificarse una entrada por cada parámetro que la función espera recibir. No se permiten más de 90 parámetros (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. 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 consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que DECIMAL(11,2) y DECIMAL(4,3). Se considera que la especificación de un tipo diferenciado con tipo no firme es del mismo tipo de datos que el tipo de fuente del tipo diferenciado. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. 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 devuelve un error (SQLSTATE 42723).

nombre-parámetro

Especifica un nombre opcional para el parámetro de entrada. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo-datos1

Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado, un tipo estructurado o un tipo de referencia. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Algunos tipos de datos no se soportan en todos los idiomas. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal, consulte "Tipos de datos que se correlacionan con tipos de datos de SQL en aplicaciones SQL incorporadas".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de una función es una versión serializada del valor XML que se pasa como parámetro en la llamada de función, los parámetros del tipo XML deben declararse mediante la sintaxis XML AS CLOB(*n*).
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- No se pueden especificar tipos array (SQLSTATE 42815).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

CREATE FUNCTION (tabla externa)

Para un tipo de referencia, el parámetro se puede especificar como REF(*nombre-tipo*) si el parámetro no tiene ámbito.

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase *cláusula-por-omisión* en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB).

AS LOCATOR

Especifica que se pasa un localizador para el valor del parámetro a la función en lugar del valor real. Especifique AS LOCATOR sólo para parámetros con un tipo de datos LOB o un tipo diferenciado basado en un tipo de datos LOB (SQLSTATE 42601). El hecho de pasar localizadores en lugar de valores puede hacer que se pasen menos bytes a la función, especialmente cuando el valor del parámetro es muy grande.

La cláusula AS LOCATOR no tiene ningún efecto para determinar si los tipos de datos se pueden promocionar, ni afecta a la signatura de función, que se utiliza en la resolución de función.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS

Especifica la salida de la función.

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. El estilo de lista es parecido al estilo de una sentencia CREATE TABLE 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.

tipo2-datos

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).

CREATE FUNCTION (tabla externa)

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 de resultados.

Los tipos válidos que se pueden utilizar con esta cláusula se describen en el tema de la sentencia "CREATE FUNCTION (escalar externa)".

GENERIC TABLE

Especifica que la salida de la función es una tabla genérica. Esta cláusula sólo está permitida si se especifica la cláusula LANGUAGE JAVA y la cláusula PARAMETER STYLE DB2GENERAL (SQLSTATE 42613).

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 descartar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de 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-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: SQLaammddhhmmsxxx.

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 da por supuesto "NAME *nombre-función*".

NAME '*serie*'

Esta cláusula identifica el código escrito por el usuario que implanta la función que se está definiendo.

La opción '*serie*' es una constante de tipo serie con un máximo de 254 bytes. 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. Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función de dentro de la biblioteca). No obstante, cuando se emplea

CREATE FUNCTION (tabla externa)

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.

La *serie* se puede especificar de la forma siguiente:

►► ' id-biblioteca !id-función ' ►►
└─id-vía-acceso-absoluta┘

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

id-biblioteca

Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

- En los sistemas UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.
- En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

id-vía-acceso-absoluta

Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, en los sistemas UNIX, /u/jchui/mylib/myfunc haría que el gestor de bases de datos busque la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id-función

Identifica el nombre del punto de entrada de la función que debe invocarse. El signo de admiración (!) sirve como delimitador entre el ID de biblioteca y el ID de función.

Por ejemplo, en un sistema UNIX, 'mymod!func8' indicaría al gestor de bases de datos que debe buscar la biblioteca \$inst_home_dir/sqllib/function/mymod y que debe utilizar el punto de entrada func8 que está dentro de esa biblioteca.

En los sistemas operativos Windows, 'mymod!func8' indicaría al gestor de bases de datos que debe cargar el archivo mymod.dll y que debe llamar a la función func8() en la biblioteca de enlace dinámico (DLL).

Si la serie no se ha formado correctamente, se devuelve 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 base de datos.

- Para LANGUAGE JAVA:

CREATE FUNCTION (tabla externa)

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 deberá existir cuando se ejecute 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.

La *serie* se puede especificar de la forma siguiente:

►► ' id-jar : id-clase . id-método ' ◀◀

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

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 'myPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../myPacks/UserFuncs/' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '...\myPacks\UserFuncs\.'

id-método

Identifica el nombre del método del objeto Java que se invocará.

- Para LANGUAGE CLR:

La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se ejecuta la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos; de lo contrario, se devolverá un error (SQLSTATE 42724). Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.

La *serie* se puede especificar de la forma siguiente:

►► ' ensamblaje : id-clase ! id-método ' ◀◀

CREATE FUNCTION (tabla externa)

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

ensamblaje

Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de instalación de DB2 (por ejemplo, c:\sqlib\function). Si el archivo reside en un subdirectorio del directorio function de la instalación, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instalación es c:\sqlib y el archivo de ensamblaje es c:\sqlib\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

id-clase

Especifica el nombre de la clase del ensamblaje proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método

Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

La *serie* especificada es el identificador de programa OLE (idprog) o identificador de clase (clsid) 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. No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al ejecutar la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

La *serie* se puede especificar de la forma siguiente:

► '—*idprog*—!*id-método*—' ◀
└──┬──┘
 idcls

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

idprog

Identifica el identificador de programa del objeto OLE.

CREATE FUNCTION (tabla externa)

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:

```
{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnnn}
```

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 del método del objeto OLE que se debe 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 de 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 del 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 para la que 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 de una clase Java.

CLR Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En estos momentos, sólo se soporta LANGUAGE CLR en el caso de las funciones definidas por el usuario que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

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 la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para las funciones definidas por el usuario que están almacenadas en DB2 para Windows de 32 bits.

CREATE FUNCTION (tabla externa)

Para obtener información acerca de la creación de funciones de tabla externa LANGUAGE OLE DB, consulte "CREATE FUNCTION (Tabla externa 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.

DB2GENERAL

Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, que están definidas como método en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

SQL

Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo gráfico ni ningún tipo definido por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode, se pueden crear funciones de tabla con PARAMETER CCSID UNICODE, pero se aplican las normas siguientes:

- Se debe especificar un orden de clasificación alternativo en la configuración de la base de datos antes de crear la función de tabla (SQLSTATE 56031). Las funciones de tabla PARAMETER CCSID UNICODE se clasifican con el orden de clasificación alternativo especificado en la configuración de la base de datos.
- No se pueden utilizar conjuntamente las tablas o las funciones de tablas creadas con CCSID ASCII y las tablas o las funciones de tablas creadas con CCSID UNICODE en una sola sentencia de SQL (SQLSTATE 53090). Esto se aplica a las tablas y a las funciones de tabla a las que se hace referencia directamente en la sentencia, así como a las tablas y las

CREATE FUNCTION (tabla externa)

funciones de tabla a las que se hace referencia indirectamente (por ejemplo, mediante restricciones de integridad referencial, activadores, tablas de consulta materializada y tablas de los cuerpos de las vistas).

- No se puede hacer referencia a las funciones de tabla creadas con `PARAMETER CCSID UNICODE` en funciones de SQL ni en métodos de SQL (SQLSTATE 560C0).
- Una sentencia de SQL que hace referencia a una función de tabla creada con `PARAMETER CCSID UNICODE` no puede invocar una función de SQL ni un método de SQL (SQLSTATE 53090).
- Los tipos gráficos, el tipo XML y los tipos definidos por el usuario no se pueden utilizar como parámetros en las funciones de tabla `PARAMETER CCSID UNICODE` (SQLSTATE 560C1).
- Las sentencias que hacen referencia a una función de tabla `PARAMETER CCSID UNICODE` sólo se pueden invocar desde un cliente de DB2 Versión 8.1 o posterior (SQLSTATE 42997).
- Las sentencias de SQL siempre se interpretan en la página de códigos de la base de datos. En particular, significa que cada carácter de los literales, literales hexadecimales e identificadores delimitados debe tener una representación en la página de códigos de la base de datos; de lo contrario, el carácter se sustituirá por el carácter de sustitución.

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, las funciones se pueden crear con `PARAMETER CCSID ASCII` o `PARAMETER CCSID UNICODE`. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

Esta cláusula no se puede especificar con `LANGUAGE OLE`, `LANGUAGE JAVA` ni `LANGUAGE CLR` (SQLSTATE 42613).

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. Con la especificación de `NOT DETERMINISTIC`, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. Un ejemplo de una función de tabla que no es determinante es aquel que hace referencia a registros especiales, variables globales, funciones no determinantes o secuencias de un modo que afecta al tipo de resultado de la función de tabla.

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 ha registrado como `FENCED`, el gestor de bases de datos protege sus recursos internos (por ejemplo, los 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`.

PRECAUCIÓN:

El uso de NOT FENCED en funciones que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 toman algunas precauciones para muchas de las anomalías inadvertidas más habituales que podrían producirse, pero no pueden asegurar la integridad completa si se emplean funciones definidas por el usuario NOT FENCED.

Para una función con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar una función definida por el usuario como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

No se pueden crear funciones LANGUAGE CLR definidas por el usuario al especificar la cláusula NOT FENCED (SQLSTATE 42601).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que la función es segura para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si la función se ha definido con un LANGUAGE distinto de OLE:

- Si la función se ha definido como THREADSAFE, el gestor de bases de datos puede invocar la función en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, una función no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Las funciones FENCED y NOT FENCED pueden ser THREADSAFE.
- Si la función se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará al mismo tiempo la función en el mismo proceso que otra rutina.

Para las funciones FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (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 corresponde a la UDF comprobar si los valores de argumentos son nulos.

CREATE FUNCTION (tabla 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, CONTAINS SQL, READS SQL DATA

Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001). Si se especifican todas las cláusulas ALLOW PARALLEL, EXECUTE ON ALL DATABASE PARTITIONS y RESULT TABLE DISTRIBUTED, sólo estará permitida la opción NO SQL.

CONTAINS SQL

Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA

Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

STATIC DISPATCH

Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION

Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada tarea paralela, en lugar de una nota para la función. Especifique la cláusula DISALLOW PARALLEL en el caso de las funciones que no funcionen bien con paralelismo.

NO EXTERNAL ACTION

Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

NO SCRATCHPAD o SCRATCHPAD *longitud*

Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Se recomienda especialmente que las funciones definidas por el usuario sean reentrantes, para que una memoria de trabajo proporcione un medio para que la función "guarde el estado" entre una llamada a la siguiente.)

CREATE FUNCTION (tabla externa)

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, establece el tamaño en bytes de la memoria de trabajo, que 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: `X'00'`.
- 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.

```
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.

FINAL CALL o NO FINAL CALL

Esta cláusula opcional especifica si debe realizarse una llamada final (y una primera llamada aparte) a una función externa. También controla cuándo debe volver a inicializarse 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.

CREATE FUNCTION (tabla externa)

Si la llamada final está realizándose porque se ha producido una interrupción o un fin de transacción, puede que la UDF no emita ninguna sentencia de SQL, a excepción del cursor CLOSE (SQLSTATE 38505). Para estas situaciones especiales de la llamada final, se pasa un valor especial en el argumento del "tipo de llamada".

DISALLOW PARALLEL o ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

Especifica si, para una única referencia a la función, debe paralelizarse la función.

DISALLOW PARALLEL

Especifica que en cada invocación de la función, DB2 invoca la función en una única partición de base de datos.

ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

Especifica que en cada invocación de la función, DB2 invoca la función en todas las particiones de base de datos. Se devuelve todos los conjuntos de resultados obtenidos en cada partición de base de datos. La función no puede ejecutar sentencias de SQL (debe especificarse también la cláusula NO SQL).

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica por DB2 a la función como argumento en tiempo de invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613).

Si se especifica DBINFO, se pasará una estructura que contiene la información siguiente:

- Nombre de la 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 a la base de datos
- ID de autorización de aplicación: el ID de autorización de ejecución de la aplicación, sin tener en cuenta las funciones anidadas entre esta función y la aplicación
- Página de códigos: 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 o release de la base de datos: la versión, release y nivel de modificación del servidor de la base de datos que invoca la función
- Plataforma: el tipo de plataforma del servidor
- Números de columna del resultado del método de tabla: matriz de números de la columna del resultado que utiliza la sentencia que hace referencia a la función; esta información permite que la función sólo devuelva los valores de columna necesarios y no todos los valores de ésta
- Número de partición de base de datos: el número de partición de base de datos en el que se invoca la función de tabla externa; en un entorno de una partición de base de datos, este valor es 0

CARDINALITY entero

Esta cláusula opcional proporciona una estimación del número esperado de filas que debe devolver la función con fines de optimización. Los valores

CREATE FUNCTION (tabla externa)

válidos para *entero* están comprendidos dentro del rango 0 a 9.223.372.036.854.775.807, ambos incluidos.

Si no se especifica la cláusula *CARDINALITY* para una función de tabla, DB2 asumirá un valor finito como valor por 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 - es decir, devuelve una fila cada vez que se llama para ello y nunca devuelve una condición de "fin de tabla" - las consultas que requieran una condición de fin de tabla para funcionar correctamente serán infinitas y se deberán interrumpir. Las consultas que contienen una cláusula *GROUP BY* o *ORDER BY* son ejemplos de esta clase de consultas. No se recomienda escribir estas 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).

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

NOT SECURED o SECURED

Especifica si la función se considera segura para el control de acceso a filas y columnas. El valor por omisión es *NOT SECURED*.

NOT SECURED

Indica que la función no se considera segura. Cuando se invoca la función, los argumentos de la función no deben hacer referencia a una columna para la que haya habilitada una máscara de columna y el control de acceso de nivel de columna esté activado para su tabla (SQLSTATE 428HA). Esta norma se aplica a las funciones definidas por el usuario no seguras que se invocan en algún punto de la sentencia.

SECURED

Indica que la función se considera segura. La función debe ser segura cuando se hace referencia a ella en un permiso de fila o una máscara de columna (SQLSTATE 428H8, SQLCODE -20470).

CREATE FUNCTION (tabla externa)

Normas

- En un entorno de base de datos particionada, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Si la función permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- **Restricciones de acceso a las tablas** Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.

Notas

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas para la promoción que afectarán a sus valores de entrada. Por ejemplo, una constante que puede utilizarse como un valor de entrada podría tener un tipo de datos incorporado distinto del que se espera y, todavía más importante, podría no promoverse al tipo de datos que se espera. De acuerdo con las normas 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 garantizar la portabilidad de las UDF entre distintas plataformas, se recomienda utilizar los tipos de datos siguientes:
 - DOUBLE o REAL en lugar de FLOAT
 - DECIMAL en lugar de NUMERIC
 - CLOB (o BLOB) en lugar de LONG VARCHAR
- La creación de una función 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- **Privilegios:** el definidor de una función siempre recibe el privilegio WITH GRANT OPTION de EXECUTE sobre la función, así como el derecho de descartarla función. Cuando la función se utiliza en una sentencia de SQL, el usuario que define la función debe disponer de privilegio EXECUTE para cualquiera de los paquetes que la función utiliza.
- **Establecimiento del valor por omisión:** los parámetros de una función que están definidos con un valor por omisión se establecen en su valor por omisión cuando se invoca la función, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca la función.

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL
 - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
 - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
 - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
 - DB2GENRL puede especificarse en lugar de DB2GENERAL

La sintaxis siguiente se acepta como comportamiento por omisión:

- ASUTIME NO LIMIT
 - NO COLLID
 - PROGRAM TYPE SUB
 - STAY RESIDENT NO
 - CCSID UNICODE en una base de datos Unicode
 - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
- **Creación de una función segura:** Por lo general, los usuarios con autorización SECADM no tienen privilegios para crear objetos de base de datos, como activadores y funciones. Suelen examinar los datos a los que accede una función, comprobar que son seguros y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya tenga los privilegios necesarios para crear una función segura definida por el usuario. Tras crear la función, revocarán la autorización CREATE_SECURE_OBJECT del propietario de la función.

El atributo SECURED se considera una aseveración que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. El gestor de bases de datos presupone que ese procedimiento de auditoría de control está en vigor para todas las sentencias ALTER FUNCTION posteriores o cambios en los paquetes externos.
 - **Invocación de otras funciones definidas por el usuario en una función segura:** Si una función segura definida por el usuario invoca otras funciones definidas por el usuario, el gestor de bases de datos no valida si esas funciones definidas por el usuario anidadas tienen el atributo SECURED. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.
 - **Sustitución de una función existente de manera que el atributo seguro cambia (de SECURED a NOT SECURED y viceversa):** Los paquetes y las sentencias de SQL colocadas dinámicamente en antememoria que dependen de la función pueden quedar invalidadas porque el atributo seguro afecta a la selección de la vía de acceso de las sentencias que implican tablas para las que se ha activado el control de acceso de nivel de fila o columna.
 - **Funciones EXTERNAL ACTION:** Si se invoca una función EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca la función variará en función del plan de acceso utilizado.

CREATE FUNCTION (tabla externa)

Ejemplos

- *Ejemplo 1:* El ejemplo siguiente registra una función de tabla escrita para devolver una fila que consta de una única columna de identificador de 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 de DB2.

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
  LANGUAGE C
  PARAMETER STYLE SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

- *Ejemplo 2:* El siguiente ejemplo registra una función de tabla OLE que se utiliza para recuperar la información de cabecera de mensaje y el texto de mensaje parcial de los mensajes en Microsoft Exchange.

```
CREATE FUNCTION MAIL()
  RETURNS TABLE (TIMERECEIVED DATE,
                 SUBJECT VARCHAR(15),
                 SIZE INTEGER,
                 TEXT VARCHAR(30))
  EXTERNAL NAME 'tfmail.header!list'
  LANGUAGE OLE
  PARAMETER STYLE SQL
  NOT DETERMINISTIC
  FENCED
  CALLED ON NULL INPUT
  SCRATCHPAD
  FINAL CALL
  NO SQL
  EXTERNAL ACTION
  DISALLOW PARALLEL
```


CREATE FUNCTION (tabla externa OLE DB)

La sentencia CREATE FUNCTION (Tabla externa OLE DB) se utiliza para registrar una función de tabla externa OLE DB definida por el usuario 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 incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, una de las autorizaciones siguientes:
 - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
 - El privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función
- Autorización DBADM

Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE FUNCTION no se tienen en cuenta.

Si se especifica la opción SECURED, el ID de autorización de la sentencia debe incluir la autorización SECADM o CREATE_SECURE_OBJECT (SQLSTATE 42501).

Sintaxis

►► CREATE FUNCTION *nombre-función* (*declaración-parámetro*) *?* ►►

►► RETURNS TABLE (*nombre-columna* | *tipo2-datos*) *lista-opciones* ►►

declaración-parámetro:

| *nombre-parámetro* | *tipo1-datos* | *cláusula-por-omisión* |

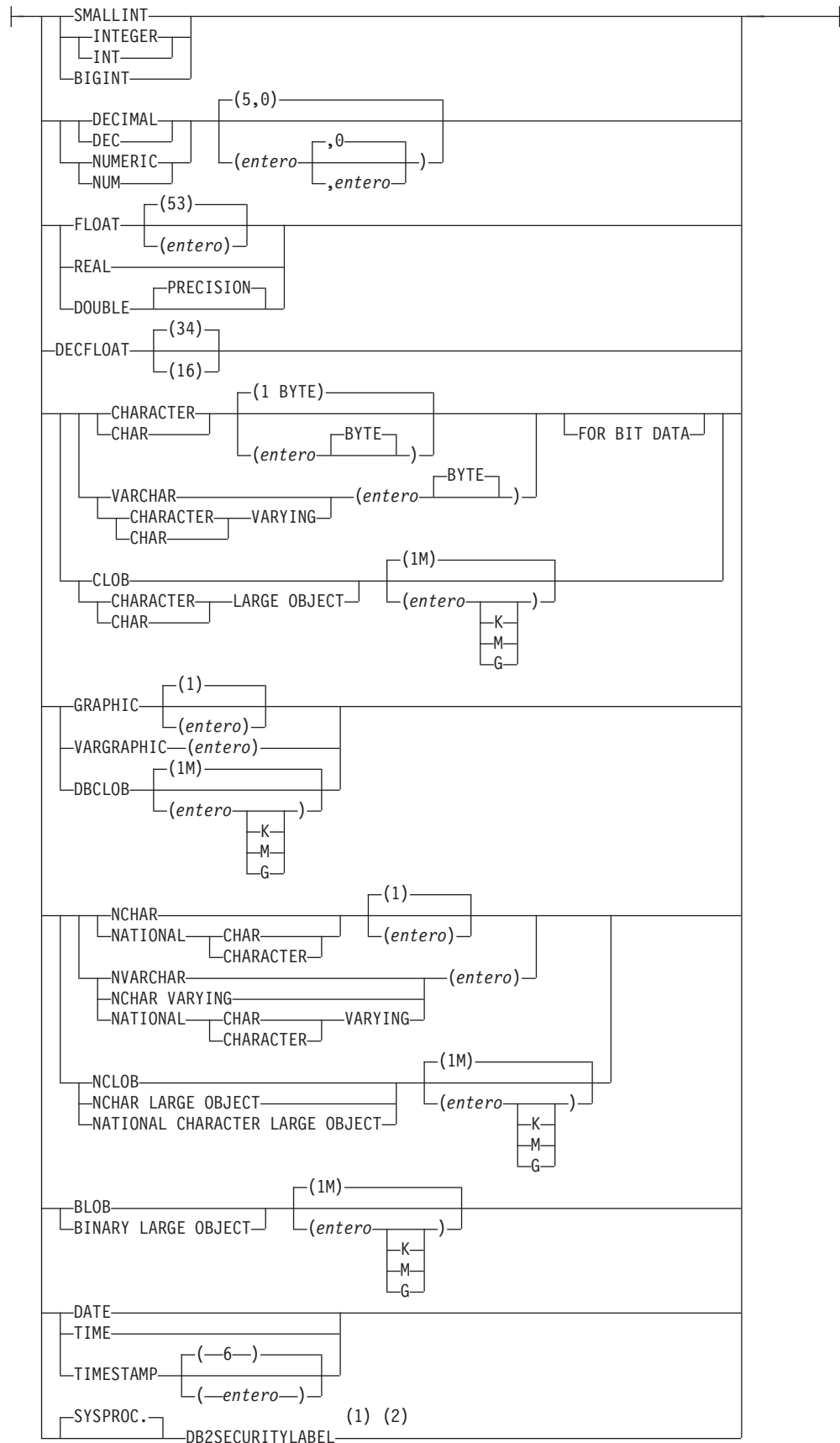
CREATE FUNCTION (tabla externa OLE DB)

tipo1-datos, tipo2-datos:



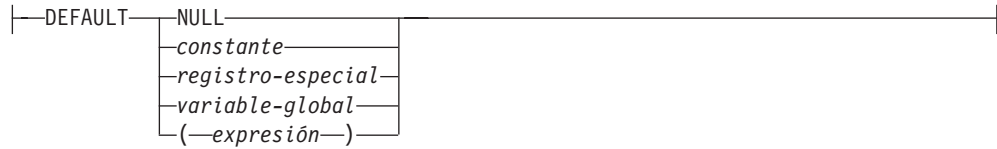
tipo-incorporado:

CREATE FUNCTION (tabla externa OLE DB)

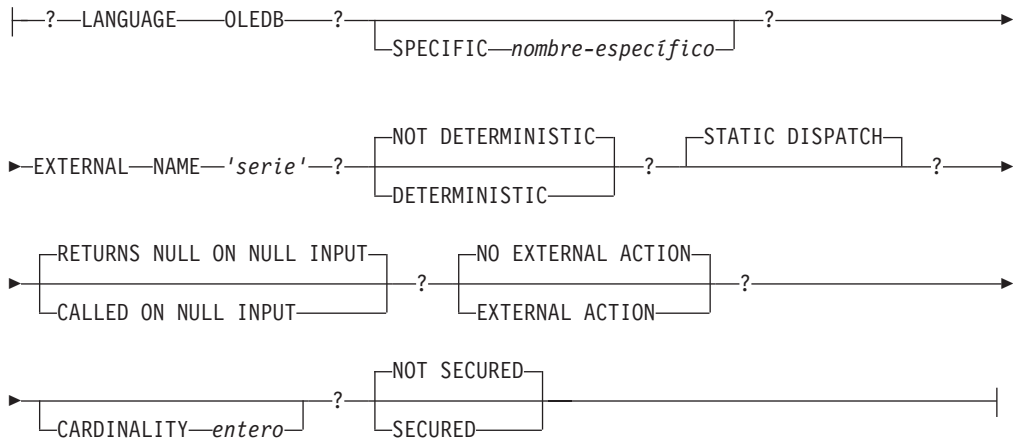


CREATE FUNCTION (tabla externa OLE DB)

cláusula-por-omisión:



lista-opciones:



Notas:

- 1 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 2 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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. El formato no calificado de un *nombre-función* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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.

CREATE FUNCTION (tabla externa OLE DB)

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice 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.

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.

(declaración-parámetro,...)

Identifica el número de parámetros de entrada de la función y especifica el tipo de datos y el valor por omisión opcional de cada 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 pasa el texto del mandato a un proveedor OLE DB.

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. 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 consideran en esta comparación de tipos. Por consiguiente, se considera que CHAR(8) y CHAR(35) son del mismo tipo. Se considera que la especificación de un tipo diferenciado con tipo no firme es del mismo tipo de datos que el tipo de fuente del tipo diferenciado. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro

Especifica un nombre opcional para el parámetro de entrada.

tipo1-datos

Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser cualquier tipo de datos de serie gráfica o de caracteres o un tipo diferenciado basado en un tipo de datos de serie gráfica o de caracteres. No se da soporte a parámetros del tipo XML (SQLSTATE 42815).

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase

CREATE FUNCTION (tabla externa OLE DB)

cláusula-por-omisión en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB).

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.

tipo2-datos

Especifica el tipo de datos de la columna. XML no es válido (SQLSTATE 42815).

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 descartar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de 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-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, 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-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 NAME '*serie*'

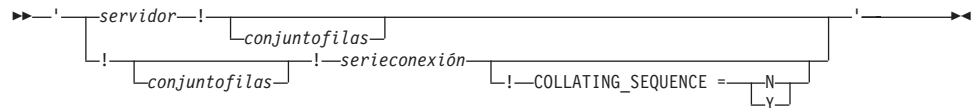
Esta cláusula identifica la tabla externa y el proveedor OLE DB.

La opción '*serie*' es una constante de tipo serie con un máximo de 254 bytes.

CREATE FUNCTION (tabla externa OLE DB)

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.

La *serie* se puede especificar de la forma siguiente:



servidor

Identifica el nombre local de una fuente de datos como lo define "CREATE SERVER".

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 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 las descripciones de las propiedades de inicialización de OLE DB (conjunto de propiedades DBPROPSET_DBINIT) o palabras clave específicas del proveedor.

COLLATING_SEQUENCE

Especifica si la fuente de datos utiliza el mismo orden de clasificación que DB2 Database para Linux, UNIX y Windows. Para obtener detalles, consulte "CREATE SERVER". Los valores válidos son los siguientes:

- Y = El mismo orden de clasificación
- N = Diferente orden de clasificación

Si no se especifica COLLATING_SEQUENCE, se da por supuesto que la fuente de datos tiene un orden de clasificación distinto de DB2 Database para Linux, UNIX y Windows.

Si se proporciona el *servidor*, *serie-conexión* 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.

CREATE FUNCTION (tabla externa 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. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

STATIC DISPATCH

Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

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, en tiempo de ejecución, alguno 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; es decir, 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 corresponde a la UDF comprobar si los valores de argumentos son 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.

NO EXTERNAL ACTION o EXTERNAL ACTION

Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es NO EXTERNAL ACTION.

NO EXTERNAL ACTION

Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

EXTERNAL ACTION

Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

CARDINALITY *entero*

Esta cláusula opcional proporciona una estimación del número esperado de filas que debe 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 valor por omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

CREATE FUNCTION (tabla externa OLE DB)

Aviso: Si una función tiene, de hecho, una cardinalidad infinita - es decir, devuelve una fila cada vez que se llama para ello y nunca devuelve una condición de "fin de tabla" - las consultas que requieran una condición de fin de tabla para funcionar correctamente serán infinitas y se deberán interrumpir. Las consultas que contienen una cláusula GROUP BY o ORDER BY son ejemplos de esta clase de consultas. No se recomienda escribir estas UDF.

NOT SECURED o SECURED

Especifica si la función se considera segura para el control de acceso a filas y columnas. El valor por omisión es NOT SECURED.

NOT SECURED

Indica que la función no se considera segura. Cuando se invoca la función, los argumentos de la función no deben hacer referencia a una columna para la que haya habilitada una máscara de columna y el control de acceso de nivel de columna esté activado para su tabla (SQLSTATE 428HA). Esta norma se aplica a las funciones definidas por el usuario no seguras que se invocan en algún punto de la sentencia.

SECURED

Indica que la función se considera segura. La función debe ser segura cuando se hace referencia a ella en un permiso de fila o una máscara de columna (SQLSTATE 428H8).

Notas

- FENCED, FINAL CALL, SCRATCHPAD, PARAMETER STYLE SQL, DISALLOW PARALLEL, NO DBINFO, NOT THREADSAFE y NO SQL están implícitas en la sentencia y pueden especificarse.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas para la promoción que afectarán a sus valores de entrada. Por ejemplo, una constante que puede utilizarse como un valor de entrada podría tener un tipo de datos incorporado distinto del que se espera y, todavía más importante, podría no promoverse al tipo de datos que se espera. De acuerdo con las normas 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 garantizar la portabilidad de las UDF entre distintas plataformas, se recomienda utilizar los tipos de datos siguientes:
 - DOUBLE o REAL en lugar de FLOAT
 - DECIMAL en lugar de NUMERIC
 - CLOB (o BLOB) en lugar de LONG VARCHAR
- La creación de una función 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- *Privilegios*: el definidor de una función siempre recibe el privilegio WITH GRANT OPTION de EXECUTE sobre la función, así como el derecho de descartarla función.
- *Establecimiento del valor por omisión*: los parámetros de una función que están definidos con un valor por omisión se establecen en su valor por omisión cuando se invoca la función, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca la función.

CREATE FUNCTION (tabla externa OLE DB)

- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
 - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
 - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
- *Creación de una función segura:* Por lo general, los usuarios con autorización SECADM no tienen privilegios para crear objetos de base de datos, como activadores o funciones. Suelen examinar los datos a los que accede una función, comprobar que son seguros y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya tenga los privilegios necesarios para crear una función segura definida por el usuario. Tras crear la función, revocarán la autorización CREATE_SECURE_OBJECT del propietario de la función.

El atributo SECURED se considera una aserción que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. El gestor de bases de datos presupone que ese procedimiento de auditoría de control está en vigor para todas las sentencias ALTER FUNCTION posteriores o cambios en los paquetes externos.
- *Invocación de otras funciones definidas por el usuario en una función segura:* Si una función segura definida por el usuario invoca otras funciones definidas por el usuario, el gestor de bases de datos no valida si esas funciones definidas por el usuario anidadas tienen el atributo SECURED. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.
- *Funciones EXTERNAL ACTION:* Si se invoca una función EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca la función variará en función del plan de acceso utilizado.

Ejemplos

- *Ejemplo 1:* Registrar una función de tabla OLE DB, que recupera información de clasificación de una base de datos de 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))

LANGUAGE OLEDB
EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
              Data Source=c:\sql1ib\samples\oledb\nwind.mdb
!COLLATING_SEQUENCE=Y';
```

- *Ejemplo 2:* Registrar una función de tabla OLE DB, que recupera información de cliente de una base de datos de Oracle. La serie de conexión se proporciona a través de una definición de servidor. El nombre de tabla está completamente

CREATE FUNCTION (tabla externa OLE DB)

calificado en el nombre externo. El usuario local john se correlaciona con el usuario remoto dave. Los demás usuarios utilizarán el ID de usuario de invitado en la serie de conexión.

```
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:* Registrar una función de tabla OLE DB, que recupera información sobre tiendas de una base de datos de 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 texto de sentencia de SQL para recuperar las tres 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;';
```

```
SELECT *
  FROM TABLE (favorites
              (' select top 3 sales.stor_id as store_id, ' CONCAT
              ' stores.stor_name as name, ' CONCAT
              ' sum(sales.qty) as sales ' CONCAT
              ' from sales, stores ' CONCAT
              ' where sales.stor_id = stores.stor_id ' CONCAT
              ' group by sales.stor_id, stores.stor_name ' CONCAT
              ' order by sum(sales.qty) desc ')) as f;
```

CREATE FUNCTION (derivado o plantilla)

La sentencia CREATE FUNCTION (derivada o plantilla) se utiliza para registrar una función o plantilla de función con un servidor.

Esta sentencia puede registrar los objetos siguientes:

- Una función definida por el usuario, basada en otra función escalar o agregada existente, en el servidor actual.
- Una plantilla de función en un servidor de aplicaciones que esté 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.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

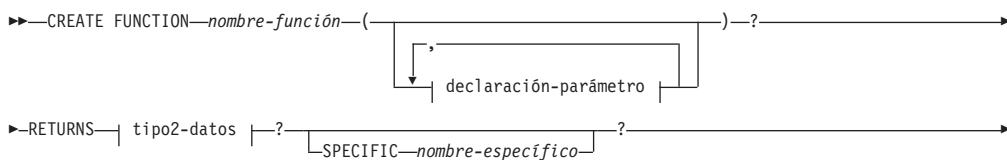
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
- El privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función
- Autorización DBADM

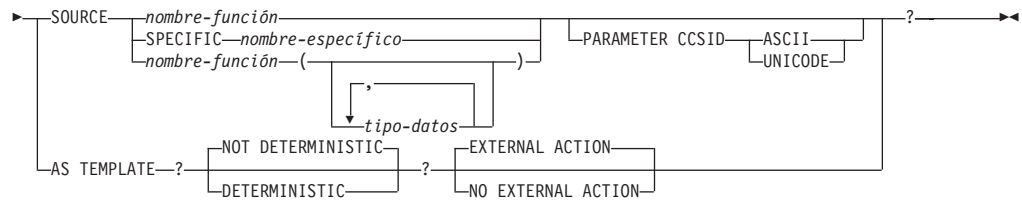
Los privilegios del ID de autorización de la sentencia también deben incluir el privilegio EXECUTE sobre la función fuente si el ID de autorización de la sentencia no tiene autorización DATAACCESS y se especifica la cláusula SOURCE.

Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE FUNCTION no se tienen en cuenta.

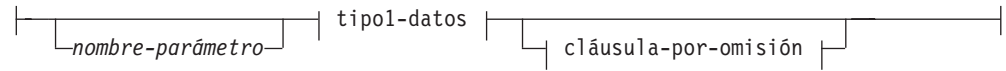
Sintaxis



CREATE FUNCTION (derivado o plantilla)



declaración-parámetro:

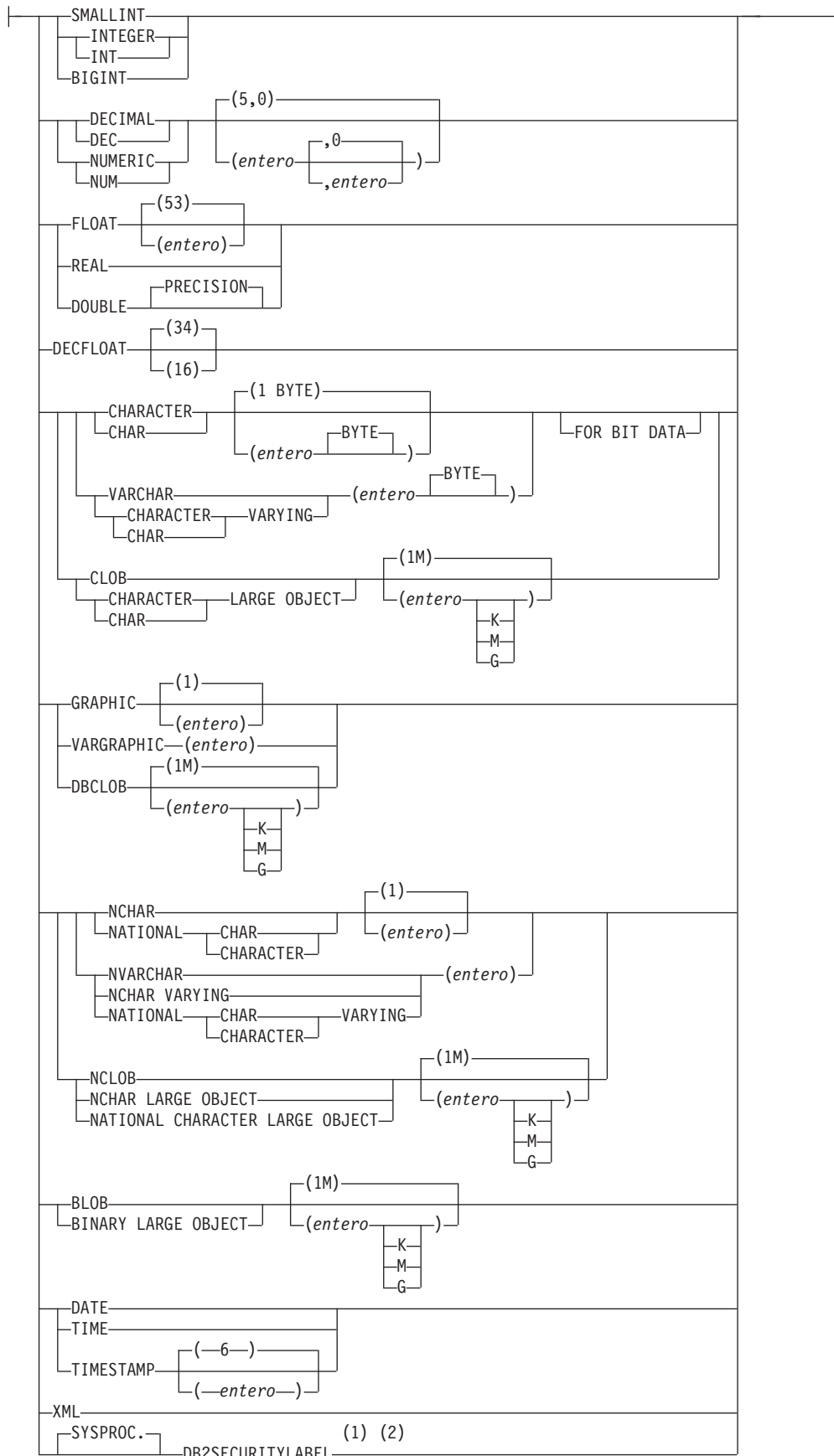


tipo1-datos, tipo2-datos:



tipo-incorporado:

CREATE FUNCTION (derivado o plantilla)



cláusula-por-omisión:



Notas:

- 1 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 2 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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. El formato no calificado de un *nombre-función* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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 de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice 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.

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.

CREATE FUNCTION (derivado o plantilla)

(declaración-parámetro,...)

Identifica el número de parámetros de entrada de la función o plantilla de función y especifica el tipo de datos y el valor por omisión opcional 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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. 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. Esta restricción también es aplicable a una función y una plantilla de función con el mismo nombre en el mismo esquema. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que DECIMAL(11,2) y DECIMAL (4,3). Se considera que la especificación de un tipo diferenciado con tipo no firme es del mismo tipo de datos que el tipo de fuente del tipo diferenciado. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. 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 devuelve un error (SQLSTATE 42723).

nombre-parámetro

Especifica un nombre opcional para el parámetro de entrada. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo-datos1

Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado o un tipo estructurado.

Se puede utilizar cualquier tipo de datos SQL válido si éste se puede convertir en el tipo del parámetro correspondiente de la función identificada en la cláusula SOURCE (para obtener información consulte "Conversiones entre tipos de datos"). Sin embargo, esta comprobación no garantiza que no se vaya a producir un error cuando se invoque la función.

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- No se pueden especificar tipos array (SQLSTATE 42879).
- No se puede especificar un tipo de referencia especificado como REF(*nombre-tipo*) (SQLSTATE 42879).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

CREATE FUNCTION (derivado o plantilla)

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

Puesto que la función es derivada, no es necesario (pero se permite) especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Pueden utilizarse paréntesis vacíos en su lugar; por ejemplo, CHAR(). Un *tipo de datos con parámetros* es cualquiera de los tipos de datos que se pueden definir con una longitud, escala o precisión específicas. Los tipos de datos con parámetros son los tipos de datos de serie, los tipos de datos decimales y el tipo de datos TIMESTAMP.

También se pueden utilizar paréntesis vacíos con una plantilla de función en lugar de especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Se recomienda utilizar paréntesis vacíos para los tipos de datos con parámetros. Si utiliza paréntesis vacíos, la longitud, la precisión o la escala son iguales que las de la función remota, que se determina cuando se correlaciona la plantilla de función con una función remota mediante la creación de una correlación de función. Si omite los paréntesis, se utiliza la longitud por omisión para el tipo de datos (vea "CREATE TABLE").

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase *cláusula-por-omisión* en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB).

RETURNS

Esta cláusula obligatoria identifica el resultado de la función o plantilla de función.

tipo2-datos

Especifica el tipo de datos de la salida.

Con una función escalar derivada, se acepta cualquier tipo de datos SQL válido, como un tipo diferenciado, siempre que se pueda convertir desde el tipo del resultado de la función fuente. Un tipo array no puede especificarse como tipo de datos de un parámetro (SQLSTATE 42879).

El parámetro de un tipo con parámetros no es necesario especificarlo para los parámetros de una función derivada. En su lugar, pueden utilizarse paréntesis vacíos; por ejemplo, VARCHAR().

CREATE FUNCTION (derivado o plantilla)

Para obtener información acerca de las consideraciones y normas adicionales que se aplican a la especificación del tipo de datos en la cláusula RETURNS cuando la función tiene su fuente en otra función, consulte el apartado “Normas” de esta sentencia.

Con una plantilla de función, los paréntesis vacíos no están permitidos (SQLSTATE 42611). Se debe especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Se recomienda especificar la misma longitud, precisión o escala que las de la función remota.

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 descartar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de 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 devuelve 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, debe ser el mismo calificador explícito o implícito del *nombre-función* o se devuelve 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: SQLaammddhhmmsxxx.

SOURCE

Especifica que la nueva función se va a definir como función derivada. Una *función derivada* se implementa mediante otra función (la *función fuente*). La función debe ser una función escalar o agregada que existe en el servidor actual, y debe ser de uno de los tipos siguientes:

- Una función que se definió con una sentencia CREATE FUNCTION
- Una función de conversión generada por una sentencia CREATE TYPE
- Una función incorporada

Si la función fuente no es una función incorporada, la función concreta se puede identificar por su nombre, signatura de función o nombre específico.

Si la función fuente es una función incorporada, la cláusula SOURCE debe incluir una signatura de función para la función incorporada. La función fuente no debe ser ninguna de las funciones incorporadas siguientes (si se indica una sintaxis concreta, no se puede especificar únicamente el formato indicado):

- CARDINALITY
- CHAR cuando se especifica más de un argumento y el primer argumento es un tipo de datos de fecha y hora
- CHARACTER_LENGTH
- COALESCE
- CONTAINS
- CURSOR_ROWCOUNT
- DATAPARTITIONNUM

CREATE FUNCTION (derivado o plantilla)

- DBPARTITIONNUM
- Deref
- EXTRACT
- GRAPHIC cuando se especifica más de un argumento y el primer argumento es un tipo de datos de fecha y hora
- GREATEST
- HASHEDVALUE
- INSERT cuando se especifican más de cuatro argumentos
- INSTR cuando se especifican más de cuatro argumentos
- LCASE cuando se especifican más de tres argumentos
- LEAST
- LEFT cuando se especifican más de dos argumentos
- LENGTH cuando se especifica más de un argumento
- LOCATE cuando se especifican más de tres argumentos
- LOCATE_IN_STRING cuando se especifican más de cuatro argumentos
- LOWER cuando se especifican más de tres argumentos
- MAX
- MAX_CARDINALITY
- MIN
- NODENUMBER
- NULLIF
- NVL
- OVERLAY
- PARAMETER
- POSITION
- RAISE_ERROR
- REC2XML
- RID
- RID_BIT
- RIGHT cuando se especifican más de dos argumentos
- SCORE
- STRIP
- SUBSTRING
- TRIM
- TRIM_ARRAY
- TYPE_ID
- TYPE_NAME
- TYPE_SCHEMA
- UCASE cuando se especifican más de tres argumentos
- UPPER cuando se especifican más de tres argumentos
- VALUE
- VARCHAR cuando se especifica más de un argumento y el primer argumento es un tipo de datos de fecha y hora
- VARGRAPHIC cuando se especifica más de un argumento y el primer argumento es un tipo de datos de fecha y hora

CREATE FUNCTION (derivado o plantilla)

- XMLATTRIBUTES
- XMLCOMMENT
- XMLCONCAT
- XMLDOCUMENT
- XMLELEMENT
- XMLFOREST
- XMLNAMESPACES
- XMLPARSE
- XMLPI
- XMLQUERY
- XMLROW
- XMLSERIALIZE
- XMLTEXT
- XMLVALIDATE
- XMLXSROBJECTID
- XSLTRANSFORM

nombre-función

Identifica la función en particular que va a utilizarse como fuente y sólo es válida si existe exactamente una función específica en el esquema con este *nombre-función* para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, para localizar la función se utiliza la vía de acceso de SQL actual (el valor del registro especial CURRENT PATH). El primer esquema de la vía de acceso de SQL que tiene una función con este nombre y para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función con este nombre en el esquema nombrado o si el esquema no está calificado y no hay ninguna función con este nombre en la vía de acceso a SQL, se devuelve un error (SQLSTATE 42704). Si hay más de una instancia específica autorizada de la función en el esquema nombrado o localizado, se devuelve un error (SQLSTATE 42725). Si existe una función con este nombre y el ID de autorización de la sentencia no tiene el privilegio EXECUTE en esta función, se devuelve un error (SQLSTATE 42501).

SPECIFIC *nombre-específico*

Identifica una función determinada definida por el usuario que debe 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, se utiliza la vía de acceso SQL actual para localizar la función. Se selecciona el primer esquema de la vía de acceso a SQL que tiene una función con este nombre específico para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función para este *nombre-específico* en el esquema nombrado o si el nombre no está calificado y no hay ninguna función con este *nombre-específico* en la vía de acceso SQL, se devuelve un error (SQLSTATE 42704). Si existe una función con este *nombre-específico* y el ID

CREATE FUNCTION (derivado o plantilla)

de autorización de la sentencia no dispone de privilegio EXECUTE para esta función, se devuelve un error (SQLSTATE 42501).

nombre-función (tipo-datos,...)

Proporciona la signatura de la función, que identifica de manera exclusiva la función fuente. Esta es la única variante de sintaxis válida para una función fuente que es una función incorporada.

Las normas para la resolución de funciones se aplicarán con el fin de seleccionar una función entre las funciones que tienen el mismo nombre de función, según 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 a 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 con parámetros 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).

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (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. Puede ser útil para asegurarse de que se va a utilizar la función que desea. Tenga en cuenta que los sinónimos de los tipos de datos también se considerarán una coincidencia (por ejemplo, DEC y NUMERIC coincidirán).

No es necesario que un tipo FLOAT(n) coincida con el valor definido para n, porque $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ninguna función con la signatura especificada en el esquema nombrado o implícito, se devuelve un error (SQLSTATE 42883).

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

CREATE FUNCTION (derivado o plantilla)

Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

La cláusula `PARAMETER CCSID` debe especificar el mismo esquema de codificación que la función fuente (`SQLSTATE 53090`).

AS TEMPLATE

Indica que esta sentencia se utilizará para crear una plantilla de función y no una función con código ejecutable.

NOT DETERMINISTIC o DETERMINISTIC

Especifica si la función devuelve los mismos resultados para argumentos de entrada idénticos. El valor por omisión es `NOT DETERMINISTIC`.

NOT DETERMINISTIC

Especifica que es posible que la función no devuelva el mismo resultado cada vez que la función se invoque con los mismos argumentos de entrada. La función depende de algunos valores de estado que afectan a los resultados. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL. Un ejemplo de una función que no es determinante es la que genera números aleatorios.

Una función que no es determinante puede recibir resultados incorrectos si se ejecuta por tareas paralelas.

DETERMINISTIC

Especifica que la función siempre devuelve el mismo resultado cada vez que se invoca con los mismos argumentos de entrada. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL. Un ejemplo de función determinante es la que calcula la raíz cuadrada del argumento de entrada.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es `EXTERNAL ACTION`.

EXTERNAL ACTION

Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona. `EXTERNAL ACTION` se debe especificar implícita o explícitamente si el cuerpo de la rutina de SQL invoca una función que está definida con `EXTERNAL ACTION` (`SQLSTATE 428C2`).

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada tarea paralela, en lugar de una nota para la función.

NO EXTERNAL ACTION

Especifica que la función no realiza ninguna acción que cambie el

CREATE FUNCTION (derivado o plantilla)

estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

Normas

- Para que resulte más sencillo, en esta sección denominaremos CF a la función que se está creando, y denominaremos FS a la función identificada en la cláusula SOURCE, con independencia de cuál de las tres sintaxis permitidas se ha utilizado para identificar a 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. Se deben extremar las precauciones al utilizar este recurso ya que podría ser muy difícil depurar una aplicación si una función invocada indirectamente devuelve 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 / CONTAINS SQL / READS SQL DATA
 - SCRATCHPAD / NO SCRATCHPAD
 - FINAL CALL / NO FINAL CALL
 - RETURNS TABLE (...)
 - CARDINALITY ...
 - ALLOW PARALLEL / DISALLOW PARALLEL
 - DBINFO / NO DBINFO
 - THREADSAFE / NOT THREADSAFE
 - INHERIT SPECIAL REGISTERS

Si se incumplen estas normas se produce un error (SQLSTATE 42613).

- El número de parámetros de entrada en CF debe ser igual a los de SF; de lo contrario, se devuelve un error (SQLSTATE 42624).
- La CF no tendrá que especificar necesariamente la longitud, precisión ni escala para un tipo de datos con parámetros 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 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 en las normas restantes para los parámetros de entrada, y se devuelve un valor.

CREATE FUNCTION (derivado o plantilla)

- La especificación de los parámetros de entrada de la CF se comprueban respecto a 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. Si algún parámetro no es del mismo tipo o no se puede convertir, se devuelve un error (SQLSTATE 42879).

Tenga en cuenta que esta norma 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 devuelve un error (SQLSTATE 42866).

Tenga en cuenta que esta norma 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 con parámetros, 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 de 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 normas de promoción que afectarán a sus valores de entrada (consulte el apartado “Promoción de tipos de datos”). 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 normas 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 todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre 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.

CREATE FUNCTION (derivado o plantilla)

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 procese una consulta que especifique la plantilla, DB2 invocará la función de fuente de datos con la que ésta se correlaciona, siempre que exista un plan de acceso para la invocación de esta función.

- **Privilegios:** el definidor de una función siempre recibe el privilegio EXECUTE de la función, así como el derecho de descartar la función. El usuario que define la función también recibe WITH GRANT OPTION si se aplica cualquiera de las condiciones siguientes:
 - La función fuente es una función incorporada.
 - El usuario que define la función dispone de EXECUTE WITH GRANT OPTION para la función fuente.
 - La función es una plantilla.
- **Funciones EXTERNAL ACTION:** Si se invoca una función EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca la función variará en función del plan de acceso utilizado.
- **Establecimiento del valor por omisión:** los parámetros de una función que están definidos con un valor por omisión se establecen en su valor por omisión cuando se invoca la función, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca la función.
- **CREATE FUNCTION MAPPING con funciones de tabla o fila:** las correlaciones de creación de función con funciones remotas que devuelven una tabla o una fila no están soportadas en las bases de datos federadas.
- **Herencia de los atributos SECURED o NOT SECURED de la función fuente:** la función derivada definida por el usuario hereda el atributo SECURED o NOT SECURED de la función fuente en la que sólo se tiene en cuenta la función definida por el usuario que ocupa la posición más alta. Si la función que ocupa la posición más alta definida por el usuario es segura, las funciones anidadas definidas por el usuario se consideran seguras. El gestor de bases de datos no valida si esas funciones anidadas definidas por el usuario son seguras. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y que se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.

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)
```

- *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 (derivado o plantilla)

```
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 código siguiente para registrar la plantilla en el servidor federado.

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)
RETURNS DOUBLE
AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

- *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 usa el código siguiente para registrar esta plantilla en el servidor federado.

```
CREATE FUNCTION BONUS ()
RETURNS DECIMAL (8,2)
AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

CREATE FUNCTION (tabla, fila o escalar de SQL)

La sentencia CREATE FUNCTION (tabla, fila o escalar de SQL) se utiliza para definir una función de tabla, fila o escalar de SQL definida por el usuario.

Una *función escalar* devuelve un solo valor cada vez que se invoca y en general es válida cuando una expresión SQL es válida. Se puede utilizar una *función de tabla* en una cláusula FROM y devuelve una tabla. Se puede utilizar una *función de fila* como función de transformación y devuelve una fila.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la función hace referencia a un esquema existente
- Autorización DBADM

y, como mínimo, una de las autorizaciones siguientes en cada tabla, vista o apodo que se identifica en cualquier selección completa:

- Privilegio CONTROL para esa tabla, vista o apodo
- Privilegio SELECT para esa tabla, vista o apodo
- Autorización DATAACCESS

Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE FUNCTION.

Los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo se aplican al invocarse la función. El ID de autorización de la conexión se puede correlacionar con un ID de autorización remoto diferente.

Los privilegios que mantiene el ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL especificadas en el cuerpo de la función.

Para sustituir una función existente, el ID de autorización de la sentencia debe ser el propietario de la función existente (SQLSTATE 42501).

Si se especifica la opción SECURED, el ID de autorización de la sentencia debe incluir la autorización SECADM o CREATE_SECURE_OBJECT (SQLSTATE 42501).

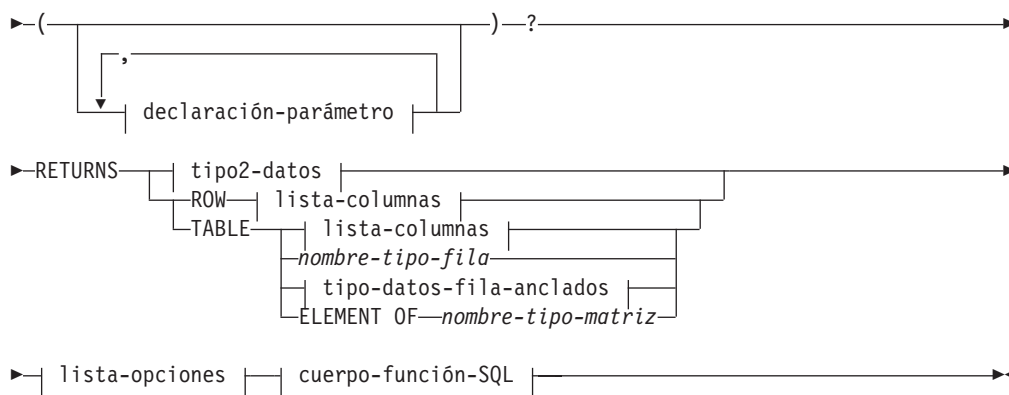
Sintaxis

```

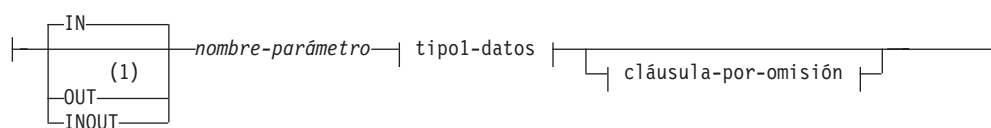
▶▶ CREATE [OR REPLACE] FUNCTION nombre-función

```

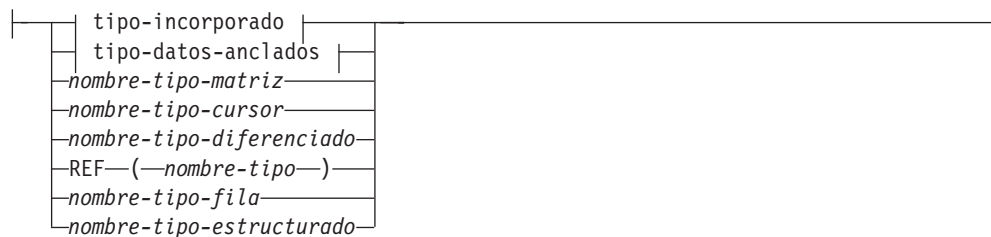
CREATE FUNCTION (tabla, fila o escalar de SQL)



declaración-parámetro:

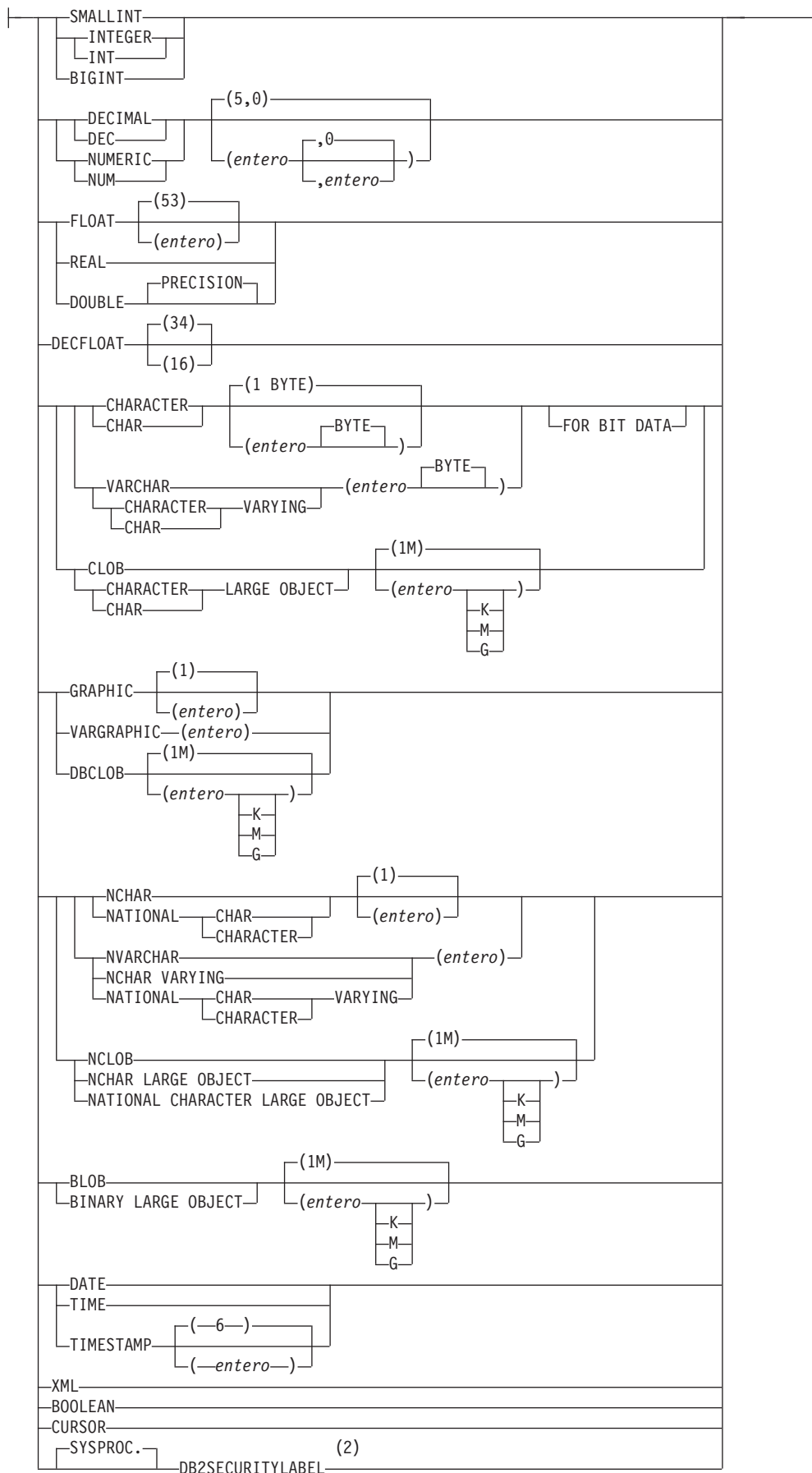


tipo1-datos, tipo2-datos:



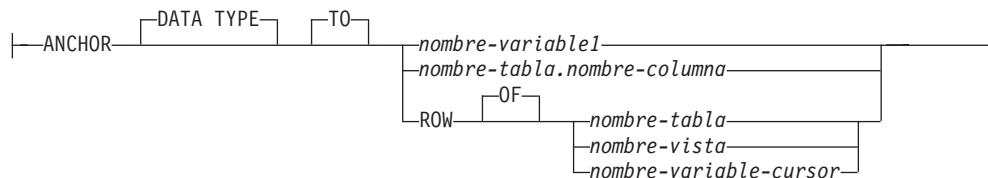
tipo-incorporado:

CREATE FUNCTION (tabla, fila o escalar de SQL)

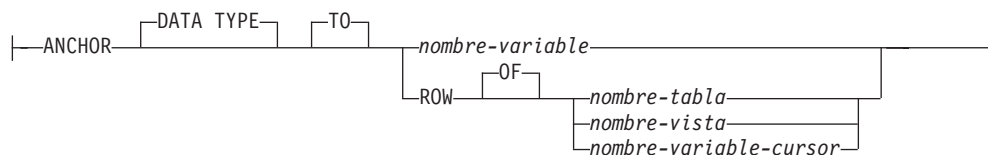


CREATE FUNCTION (tabla, fila o escalar de SQL)

tipo-datos-anclados:



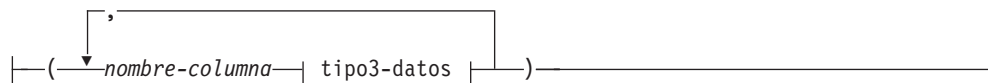
tipo-datos-fila-anclados:



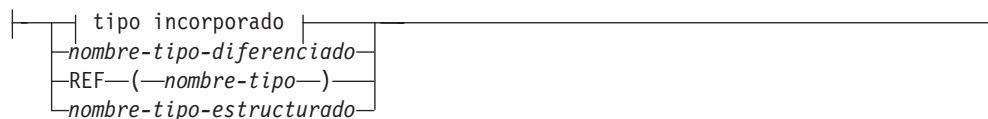
cláusula-por-omisión:



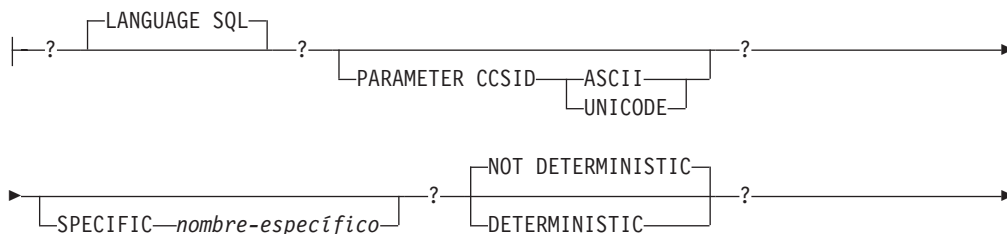
lista-columnas:



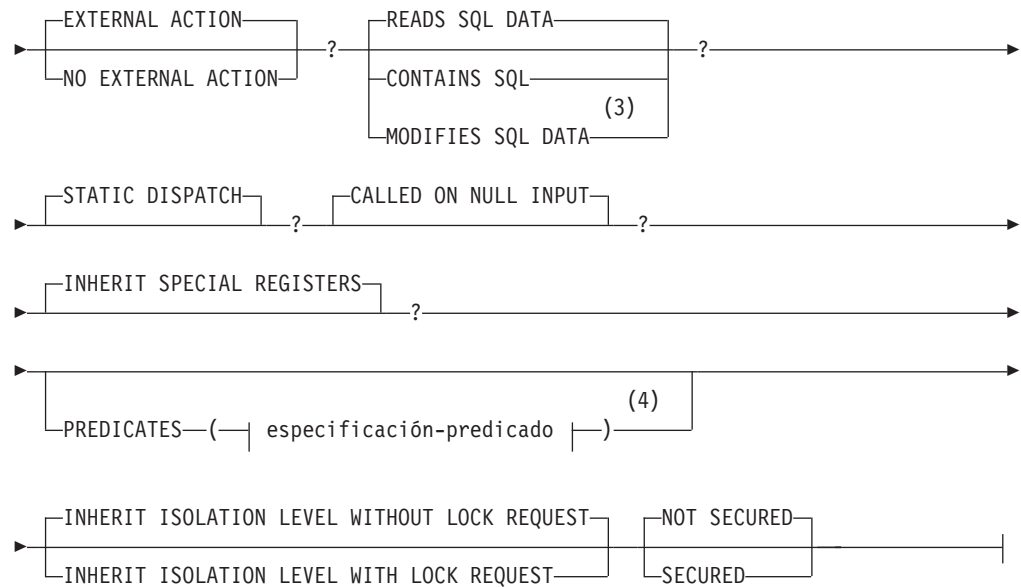
tipo3-datos:



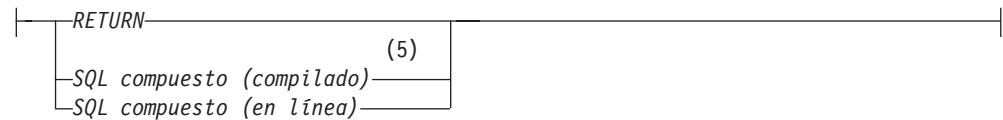
lista-opciones:



CREATE FUNCTION (tabla, fila o escalar de SQL)



cuerpo-función-SQL:



Notas:

- 1 OUT e INOUT sólo son válidos si RETURNS especifica un resultado escalar y cuerpo-función-SQL es una sentencia de SQL compuesto (compilado).
- 2 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 3 Solo es válido para la definición de función escalar compilada y una definición de función de tabla en línea. Una función escalar compilada definida como MODIFIES SQL DATA solo puede utilizarse como el único elemento del lado derecho de una sentencia de asignación que se encuentra dentro de una sentencia de SQL compuesto (compilado).
- 4 Sólo es válido si RETURNS especifica un resultado escalar (*tipo-datos2*)
- 5 A la especificación de una sentencia de SQL compuesto (compilado) se aplica lo siguiente: a) Debe utilizarse si los tipos de datos de los parámetros o tipos de datos devueltos incluyen un tipo de fila, tipo de matriz o tipo de cursor; b) Debe utilizarse si la cláusula RETURNS TABLE especifica cualquier sintaxis que no sea una lista-columnas; c) No recibe soporte si se especifica RETURNS ROW; d) No recibe soporte cuando se define una función de tabla en un entorno de base de datos particionada.

Descripción

OR REPLACE

Especifica que se debe sustituir la definición de la función si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los

CREATE FUNCTION (tabla, fila o escalar de SQL)

privilegios que se han otorgado sobre la función no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para la función en el servidor actual. Para sustituir una función ya existente, el nombre específico y el nombre de función de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de función de la antigua definición, o la signatura de la nueva definición debe coincidir con la signatura de la antigua definición. De lo contrario, se creará una nueva función.

Si se hace referencia a la función en la definición de un permiso de fila o una máscara de columna, la función no se puede sustituir (SQLSTATE 42893).

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. El formato no calificado de un *nombre-función* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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 de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice 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.

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.

(declaración-parámetro,...)

Identifica el número de parámetros de entrada de la función y especifica la modalidad, el nombre, el tipo de datos y el valor por omisión opcional 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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. 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 consideran en esta comparación de tipos. Por consiguiente, se considera que CHAR(8) y CHAR(35) son del mismo

CREATE FUNCTION (tabla, fila o escalar de SQL)

tipo, igual que DECIMAL(11,2) y DECIMAL (4,3), así como DECFLOAT(16) y DECFLOAT(34). Se considera que la especificación de un tipo diferenciado con tipo no firme es del mismo tipo de datos que el tipo de fuente del tipo diferenciado. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. 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 devuelve un error (SQLSTATE 42723).

Si el tipo de datos de un parámetro es tipo de datos booleano, tipo de matriz, tipo de cursor o tipo de fila, el cuerpo de la función de SQL solo puede hacer referencia al parámetro dentro de la sentencia de SQL compuesto (compilado) (SQLSTATE 428H2).

IN | OUT | INOUT

Especifica la modalidad del parámetro. Si la función devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian. El valor por omisión es IN.

IN Identifica el parámetro como un parámetro de entrada para la función. Los cambios que se realicen en el parámetro dentro de la función no estarán disponibles para el contexto de invocación cuando se devuelva el control.

OUT

Identifica el parámetro como un parámetro de salida para la función.

La función debe ser una función escalar que se ha definido con una sentencia de SQL compuesto (compilado) (SQLSTATE 42613).

A la función sólo puede hacerse referencia en la parte derecha de una sentencia de asignación que se encuentre en una sentencia de SQL compuesto (compilado), y la referencia de función no puede formar parte de una expresión (SQLSTATE 42887).

INOUT

Identifica el parámetro como parámetro de entrada y de salida para la función.

La función debe ser una función escalar que se ha definido con una sentencia de SQL compuesto (compilado) (SQLSTATE 42613).

A la función sólo puede hacerse referencia en la parte derecha de una sentencia de asignación que se encuentre en una sentencia de SQL compuesto (compilado), y la referencia de función no puede formar parte de una expresión (SQLSTATE 42887).

nombre-parámetro

Especifica un nombre para el parámetro. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo1-datos

Especifica el tipo de datos del parámetro.

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado, excepto BOOLEAN y CURSOR, que no pueden especificarse para una tabla, consulte el apartado "CREATE TABLE".

CREATE FUNCTION (tabla, fila o escalar de SQL)

BOOLEAN

Para un booleano.

CURSOR

Para una referencia a un cursor subyacente.

tipo-datos-anclados

Identifica otro objeto que se utiliza para definir el tipo de datos de parámetro. El tipo de datos del objeto de anclaje puede ser cualquiera de los tipos de datos explícitamente permitidos como *tipo-datos1*. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican a la especificación del tipo de datos directamente o, en el caso de una fila, a la creación de un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable1

Identifica una variable global. El tipo de datos de la variable global se utiliza como tipo de datos para el *nombre-parámetro*.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el *nombre-parámetro*.

ROW OF *nombre-tabla* o *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. El tipo de datos del *nombre-parámetro* es un tipo de fila sin nombre.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los elementos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula CONSTANT especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable de cursor no es de tipo firme que utiliza un tipo de fila con nombre, el tipo de datos de *nombre-parámetro* es un tipo de fila sin nombre.

nombre-tipo-matriz

Especifica el nombre de un tipo de matriz definido por el usuario. Si se especifica el *nombre-tipo-matriz* sin un nombre de esquema, el tipo de matriz se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-cursor

Especifica el nombre de un tipo de cursor. Si se especifica el

CREATE FUNCTION (tabla, fila o escalar de SQL)

nombre-tipo-cursor sin un nombre de esquema, el tipo de cursor se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. La longitud, precisión y la escala del parámetro son, respectivamente, la longitud, la precisión y la escala del tipo de fuente del tipo diferenciado. Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

REF (*nombre-tipo*)

Especifica un tipo de referencia sin ámbito. El *nombre-tipo* especificado debe identificar un tipo estructurado definido por el usuario (SQLSTATE 428DP). 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 eliminación de referencia 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 eliminación de referencia sólo si primero se convierte para que tenga un ámbito. Si se especifica un nombre de tipo sin un nombre de esquema, el *nombre-tipo* se resuelve buscando los esquemas en la vía de acceso de SQL.

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. Los campos de cada parámetro son los campos del tipo de fila. Si se especifica el *nombre-tipo-fila* sin un nombre de esquema, el tipo de fila se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-estructurado

Especifica el nombre de un tipo estructurado definido por el usuario. Si se especifica el *nombre-tipo-estructurado* sin un nombre de esquema, el tipo estructurado se resuelve buscando en los esquemas de la vía de acceso de SQL.

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase *cláusula-por-omisión* en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión en las situaciones siguientes:

- Para parámetros INOUT o OUT (SQLSTATE 42601)

CREATE FUNCTION (tabla, fila o escalar de SQL)

- Para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB)
- Para un parámetro de una definición de función donde también se ha especificado una cláusula RETURNS ROW o una cláusula PREDICATES (SQLSTATE 42613)

RETURNS

Esta cláusula obligatoria identifica el tipo de salida de la función.

Si el tipo de datos de la salida de la función es un tipo de datos booleano, tipo de matriz, tipo de cursor o tipo de fila, el cuerpo de la función de SQL debe ser una sentencia de SQL compuesto (compilado) (SQLSTATE 428H2).

tipo2-datos

Especifica el tipo de datos de la salida.

En esta sentencia, son válidas exactamente las mismas consideraciones que para los parámetros de funciones de SQL descritos anteriormente en *tipo1-datos* para los parámetros de función.

ROW

Especifica que la salida de la función es una única fila. Si la función devuelve más de una fila, se devolverá un error (SQLSTATE 21505).

Esta forma de función de fila sólo puede utilizarse como función de transformación para un tipo estructurado (que tenga un tipo estructurado como parámetro y que sólo devuelva tipos de datos incorporados).

lista-columnas

La lista de nombres de columnas y tipos de datos devueltos para una función ROW. La *lista-columnas* debe incluir, como mínimo, dos columnas (SQLSTATE 428F0).

nombre-columna

Especifica el nombre de esta columna. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la lista.

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.

Se aplican las mismas consideraciones que para los parámetros de funciones de SQL descritos anteriormente en *tipo1-datos* para los parámetros de función. Sin embargo, el *tipo-datos3* no da soporte a: *tipo-datos-anclados*, *nombre-tipo-matriz*, *nombre-tipo-cursor* y *nombre-tipo-fila*.

TABLE

Especifica que el resultado de la función es una tabla.

lista-columnas

La lista de nombres de columnas y tipos de datos devueltos para una función TABLE.

nombre-columna

Especifica el nombre de esta columna. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la lista.

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.

CREATE FUNCTION (tabla, fila o escalar de SQL)

Se aplican las mismas consideraciones que para los parámetros de funciones de SQL descritos anteriormente en *tipo1-datos* para los parámetros de función. Sin embargo, el *tipo-datos3* no da soporte a: *tipo-datos-anclados*, *nombre-tipo-matriz*, *nombre-tipo-cursor* y *nombre-tipo-fila*.

nombre-tipo-fila

Especifica un tipo de fila cuyos campos se utilizan para derivar la lista de columnas. Los nombres de campo de los tipos de fila se utilizan como nombres de columna.

tipo-datos-fila-anclados

Identifica la información de fila de otro objeto que ha de utilizarse como columnas de la tabla devuelta.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable a la que se hace referencia debe ser un tipo de fila.

ROW OF *nombre-tabla* o *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. Los tipos de datos de las columnas de objeto de anclaje tienen las mismas limitaciones que se aplican a *tipo-datos3*.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los objetos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula **CONSTANT** especificando una sentencia-select en la que todas las columnas de resultados tienen nombre.

ELEMENT OF *nombre-tipo-matriz*

Especifica un tipo de matriz cuyo tipo de datos de elemento se utiliza para derivar la lista de columnas. Si *nombre-tipo-matriz* identifica un tipo de matriz con elementos que son un tipo de fila, los nombres de campo del tipo de fila se utilizan como los nombres de columna. Si el *nombre-tipo-matriz* identifica un tipo de matriz con elementos que no son tipos de fila, el resultado único del nombre de columna es **COLUMN_VALUE**.

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 descartar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. El formato no calificado de un *nombre-especifico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de SQL. El nombre

CREATE FUNCTION (tabla, fila o escalar de SQL)

(incluido el calificador implícito o explícito) no debe designar 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, debe ser el mismo que el calificador explícito o implícito de *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.

LANGUAGE SQL

Especifica que la función está escrita en SQL.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

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. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION

Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

NO EXTERNAL ACTION

Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

CREATE FUNCTION (tabla, fila o escalar de SQL)

CONTAINS SQL, READS SQL DATA o MODIFIES SQL DATA

Indica qué tipo de sentencias de SQL se pueden ejecutar.

CONTAINS SQL

Indica que la función puede ejecutar sentencias de SQL que no lean ni modifiquen datos SQL (SQLSTATE 42985).

READS SQL DATA

Indica que la función puede ejecutar sentencias de SQL que no modifiquen datos SQL (SQLSTATE 42985).

MODIFIES SQL DATA

Indica que la función puede ejecutar todas las sentencias de SQL soportadas en el cuerpo-función-SQL.

STATIC DISPATCH

Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

CALLED ON NULL INPUT

Esta cláusula indica que se invoca la función con independencia de si 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.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional indica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no de la definición del objeto).

Al proceso que llama a la función no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Algunos registros especiales, como los registros especiales de hora y fecha, reflejan una propiedad de la sentencia que se está ejecutando y, por consiguiente, nunca se heredan de quien llama.

PREDICATES

Para los predicados que hacen uso de esta función, esta cláusula indica quiénes 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). Si se especifica la cláusula PREDICATES y la base de datos no es Unicode, no se debe especificar PARAMETER CCSID UNICODE (SQLSTATE 42613). No se puede especificar PREDICATES si el cuerpo-función-SQL es una sentencia de SQL compuesto (compilado) (SQLSTATE 42613).

especificación-predicado

Para obtener información detallada acerca de la especificación del predicado, consulte "CREATE FUNCTION (escalar externa)".

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST o INHERIT ISOLATION LEVEL WITH LOCK REQUEST

Especifica si una petición de bloqueo puede asociarse o no a la cláusula de

CREATE FUNCTION (tabla, fila o escalar de SQL)

aislamiento de la sentencia cuando la función hereda el nivel de aislamiento de la sentencia que invoca la función. El valor por omisión es INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST

Especifica que, como la función hereda el nivel de aislamiento de la sentencia que invoca, no se puede invocar en el contexto de una sentencia de SQL que incluya una cláusula de petición de bloqueo como parte de la cláusula de aislamiento especificada (SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST

Especifica que, como la función hereda el nivel de aislamiento de la sentencia que invoca, también hereda la cláusula de petición de bloqueo especificada.

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.

Para más información sobre la secuencia RETURN, consulte la sección Sentencia RETURN.

Para *SQL compuesto (compilado)*, consulte: Sentencia de SQL compuesto (compilado).

Para *SQL compuesto (en línea)*, consulte: Sentencia de SQL compuesto (en línea).

NOT SECURED o SECURED

Especifica si la función se considera segura para el control de acceso a filas y columnas. El valor por omisión es NOT SECURED.

NOT SECURED

Indica que la función no se considera segura. Cuando se invoca la función, los argumentos de la función no deben hacer referencia a una columna para la que haya habilitada una máscara de columna y el control de acceso de nivel de columna esté activado para su tabla (SQLSTATE 428HA). Esta norma se aplica a las funciones definidas por el usuario no seguras que se invocan en algún punto de la sentencia.

SECURED

Indica que la función se considera segura. La función debe ser segura cuando se hace referencia a ella en un permiso de fila o una máscara de columna (SQLSTATE 428H8).

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.
- **Uso de tipos de fila y de cursor:** una función que utiliza un tipo de cursor o un tipo de fila para un parámetro o que devuelve un tipo de cursor o un tipo de fila sólo puede invocarse desde una sentencia de SQL compuesto (compilado) (SQLSTATE 428H2).
- **Restricciones de acceso a las tablas:** si se define una función como READS SQL DATA, ninguna sentencia de la función podrá acceder a una tabla que la sentencia que invocó la función esté modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha

CREATE FUNCTION (tabla, fila o escalar de SQL)

definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.

Si una función definida con MODIFIES SQL DATA contiene sentencias CALL anidadas, no se permite el acceso de lectura a las tablas que la función está modificando (por la definición de función o la sentencia que ha invocado la función) (SQLSTATE 57053).

- **Uso en un entorno de base de datos particionada:** En un entorno de base de datos particionada, solo puede hacerse referencia a una función escalar definida mediante la utilización de una sentencia de SQL compuesto (compilado) en la parte derecha de una sentencia de asignación, y la referencia de función no puede formar parte de una expresión. Una sentencia de asignación de este tipo no puede aparecer en una sentencia de SQL compuesto (en línea).

Notas

- La resolución de las llamadas de función dentro del cuerpo de la función se realiza de acuerdo con la vía de acceso de SQL 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.
- Si un objeto al que se hace referencia en el cuerpo de la función de SQL no existe o se ha marcado como no válido o el definidor no tiene temporalmente los privilegios para acceder al objeto y, si el parámetro de configuración de la base de datos **auto_reval** no se ha establecido en DISABLED, la función de SQL seguirá creándose satisfactoriamente. La función SQL se marcará como no válida y se volverá a validar la próxima vez que se invoque.
- Todas las sentencias que crean funciones o métodos aplican las normas siguientes:
 - 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 estar en una relación de alteración temporalmente. Esto significa que si la función fuera un método con su primer parámetro como sujeto, no debe alterar a ni ser alterado temporalmente por otro método. Para obtener más información acerca de los métodos de alteración temporal, consulte la sentencia “CREATE TYPE (estructurado)”.
 - Puesto que la alteración temporal no se aplica a las funciones, pueden existir dos funciones de tal forma que, si fueran métodos, una alteraría temporalmente a la otra.

Por lo que respecta a la comparación de tipos de parámetros en las normas 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.
- **Privilegios:** el definidor de una función siempre recibe el privilegio EXECUTE de la función, así como el derecho de descartar la función. El usuario que define la función también recibe WITH GRANT OPTION para la función si dicho

CREATE FUNCTION (tabla, fila o escalar de SQL)

usuario dispone de WITH GRANT OPTION para todos los privilegios necesarios para definir la función o si el usuario que define la función dispone de autorización SYSADM o DBADM.

El usuario que define una función sólo adquiere privilegios si, en el momento de crearse la función, existen los privilegios de los que estos privilegios se obtienen. El usuario que define el método debe disponer de estos privilegios directamente o bien porque PUBLIC tiene los privilegios. Los privilegios que tienen los grupos de los que es miembro el usuario que define la función no se consideran. Cuando se utiliza la función, el ID de autorización del usuario conectado debe disponer de los privilegios válidos para la tabla o vista a la que hace referencia el apodo en la fuente de datos.

- **Establecimiento del valor por omisión:** los parámetros de una función que están definidos con un valor por omisión se establecen en su valor por omisión cuando se invoca la función, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca la función.
- **Funciones EXTERNAL ACTION:** Si se invoca una función EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca la función variará en función del plan de acceso utilizado.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse:
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUTLa sintaxis siguiente se acepta como comportamiento por omisión:
 - CCSID UNICODE en una base de datos Unicode
 - CCSID ASCII en una base de datos que no es Unicode
- **Creación de una función segura:** Por lo general, los usuarios con autorización SECADM no tienen privilegios para crear objetos de base de datos, como activadores o funciones. Suelen examinar los datos a los que accede una función, comprobar que son seguros y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya tenga los privilegios necesarios para crear una función segura definida por el usuario. Tras crear la función, revocarán la autorización CREATE_SECURE_OBJECT del propietario de la función.

El atributo SECURED se considera una aserción que declara que el usuario ha establecido un procedimiento de auditoría de control de todos los cambios en la función definida por el usuario. El gestor de bases de datos presupone que ese procedimiento de auditoría de control está en vigor para todas las sentencias ALTER FUNCTION posteriores o cambios en los paquetes externos.
- **Invocación de otras funciones definidas por el usuario en una función segura:** Si una función segura definida por el usuario invoca otras funciones definidas por el usuario, el gestor de bases de datos no valida si esas funciones definidas por el usuario anidadas tienen el atributo SECURED. Si esas funciones anidadas pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que garantizar que esas funciones pueden acceder a esos datos y se ha establecido un procedimiento de auditoría de control de cambios para todos los cambios en esas funciones.
- **Sustitución de una función existente de manera que el atributo seguro cambia (de SECURED a NOT SECURED y viceversa):** Los paquetes y las sentencias de SQL colocadas dinámicamente en antememoria que dependen de la función pueden quedar invalidadas porque el atributo seguro afecta a la selección de la

CREATE FUNCTION (tabla, fila o escalar de SQL)

vía de acceso de las sentencias que implican tablas para las que se ha activado el control de acceso de nivel de fila o columna.

Ejemplos

- *Ejemplo 1:* Defina 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:* Defina 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:* Defina 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
  RETURN
  SELECT EMPNO, LASTNAME, FIRSTNAME
  FROM EMPLOYEE
  WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

- *Ejemplo 4:* Defina la función de tabla del Ejemplo 3 con auditoría.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  MODIFIES SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  BEGIN ATOMIC
  INSERT INTO AUDIT
  VALUES (USER,
          'Table: EMPLOYEE Prd: DEPTNO = ' || CONCAT DEPTNO);
  RETURN
  SELECT EMPNO, LASTNAME, FIRSTNAME
  FROM EMPLOYEE
  WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
  END
```

- *Ejemplo 5:* Defina una función escalar que invierta una serie de caracteres.

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(4000))
  RETURNS VARCHAR(4000)
  DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
  BEGIN ATOMIC
```

CREATE FUNCTION (tabla, fila o escalar de SQL)

```
DECLARE REVSTR, RESTSTR VARCHAR(4000) DEFAULT '';
DECLARE LEN INT;
IF INSTR IS NULL THEN
RETURN NULL;
END IF;
SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
WHILE LEN > 0 DO
SET (REVSTR, RESTSTR, LEN)
= (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
SUBSTR(RESTSTR, 2, LEN - 1),
LEN - 1);
END WHILE;
RETURN REVSTR;
END
```

- *Ejemplo 6:* crear una función que incremente una variable pasada como parámetro INOUT y devuelva un error como código de retorno.

```
CREATE FUNCTION increment(INOUT result INTEGER, IN delta INTEGER)
RETURNS INTEGER
BEGIN
DECLARE code INTEGER DEFAULT 0;
DECLARE SQLCODE INTEGER;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION BEGIN
SET code = SQLCODE;
RETURN code;
END;
SET result = result + delta;
RETURN code;
END@
```

- *Ejemplo 7:* Crear una función de SQL compilado que tome un documento XML como entrada y devuelva el nombre de cliente.

```
CREATE FUNCTION get_customer_name_compiled(doc XML)
RETURNS VARCHAR(25)
BEGIN
RETURN XMLCAST(XMLQUERY
('$d/customerinfo/name' PASSING doc AS "d")AS VARCHAR(25));
END
```

- *Ejemplo 8:* Crear una función de SQL compilado que tome un número de teléfono y un número de región pasado como parámetros IN y devuelva el número completo en un parámetro OUT de XML.

```
CREATE FUNCTION construct_xml_phone
(IN phoneNo VARCHAR(20),
IN regionNo VARCHAR(8),
OUT full_phone_xml XML)
RETURNS VARCHAR(28)
LANGUAGE SQL
NO EXTERNAL ACTION
BEGIN
SET full_phone_xml = XMLELEMENT (NAME "phone", regionNo || phoneNo);
RETURN regionNo || phoneNo;
END
```

CREATE FUNCTION MAPPING

La sentencia CREATE FUNCTION MAPPING puede definir una correlación entre una plantilla de función o función de base de datos federada y una función de fuente de datos o bien inhabilitar una correlación por omisión entre una función de base de datos federada y una función de fuente de datos.

Cuando se define una correlación, la sentencia CREATE FUNCTION MAPPING puede asociar la plantilla o función de base de datos federada a una función en las fuentes siguientes:

- Una fuente de datos especificada
- Un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular

Si se pueden aplicar múltiples correlaciones de función a una función, se aplica la más reciente.

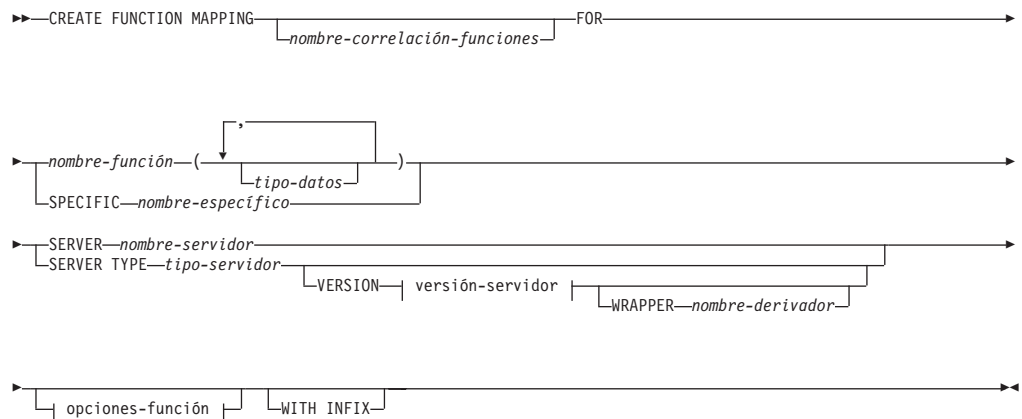
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

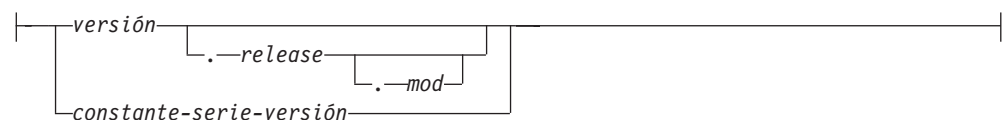
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

Sintaxis



versión-servidor:



CREATE FUNCTION MAPPING

opciones-función:

```
|—OPTIONS—(—nombre-opción-funciones—constante-serie—)|
```

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

Especifica el nombre calificado o no calificado de la función o de la plantilla de función de base de datos federada desde la cual se debe realizar la correlación.

tipo-datos

Para una función o una plantilla de función que tiene parámetros de entrada, *tipo-datos* especifica el tipo de datos de cada parámetro. El *tipo de datos* no puede ser un tipo XML ni un tipo definido por el usuario.

Se pueden utilizar paréntesis vacíos en lugar de especificar la longitud, la precisión o la escala para tipos de datos con parámetros. Se recomienda utilizar los paréntesis vacíos para tipos de datos con parámetros; por ejemplo, CHAR(). Un tipo de datos con parámetros es cualquiera de los tipos de datos que se puede definir con una longitud, una escala o una precisión específica. Los tipos de datos con parámetros son los tipos de datos de serie y los tipos de datos decimales. Si especifica la longitud, la precisión o la escala, deben ser las mismas que las de la plantilla de función. Si se omiten los paréntesis, se utilizará la longitud por omisión para el tipo de datos (consulte la descripción de la sentencia CREATE TABLE).

SPECIFIC *nombre-específico*

Identifica la función o la plantilla de función desde la cual se debe realizar la correlación. Especifique *nombre-específico* para crear un nombre de función conveniente.

SERVER *nombre-servidor*

Nombra la fuente de datos que contiene la función que se está correlacionando.

SERVER TYPE *tipo-servidor*

Identifica el tipo de fuente de datos que contiene la función que se está correlacionando.

VERSION

Identifica la versión de la fuente de datos indicada por *tipo-servidor*.

versión

Especifica el número de versión. El valor debe ser un entero.

release

Especifica el número del release de la versión indicada por *versión*. El valor debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. El valor 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 estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-derivador*

Especifica el nombre del derivador 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 deben habilitar.

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 debe generar en formato infijo. El sistema de base de datos federado convierte la notación de prefijo en la notación infijo que utiliza la fuente de datos remota.

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 se definen para la función o la plantilla federadas son compatibles con los tipos de datos correspondientes definidos para la función de la fuente de datos.
- Si una petición distribuida hace referencia a una función de 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 contrario, si la invocación de la función DB2 requiere más actividad general, se invocará a 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 ser no operativas inhabilitándolas (no se pueden descartar). Para inhabilitar una correlación de funciones por omisión, codifique la sentencia CREATE FUNCTION MAPPING de modo que especifique el nombre de la función de 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* utilizando el calificador explícito SYSIBM; por ejemplo, SYSIBM.LENGTH().

CREATE FUNCTION MAPPING

- Una sentencia CREATE FUNCTION MAPPING de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
 - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o 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 uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
 - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- **CREATE FUNCTION MAPPING con funciones de tabla o fila:** las correlaciones de creación de función con funciones remotas que devuelven una tabla o una fila no están soportadas en las bases de datos federadas.
- **Alternativas de sintaxis:** se soporta la sintaxis siguiente para mantener la compatibilidad con versiones anteriores de DB2:
 - ADD se puede especificar antes de *nombre-opción-función constante-serie*.

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

```
CREATE FUNCTION MAPPING
FOR SYSFUN.WEEK(INT)
SERVER TYPE ORACLE
OPTIONS (DISABLE 'Y')
```

- *Ejemplo 4:* Correlacione la función federada UCASE(CHAR) con una UDF que se utiliza en una fuente de datos Oracle denominada 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 GLOBAL TEMPORARY TABLE

La sentencia CREATE GLOBAL TEMPORARY TABLE crea una descripción de una tabla temporal en el servidor actual. Cada sesión que seleccione desde una tabla temporal creada sólo recuperará filas que haya insertado la misma sesión. Cuando la sesión finaliza, se suprimen las filas de la tabla asociadas con la sesión.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM o la autorización CREATETAB en combinación con otra autorización, como se describe a continuación.

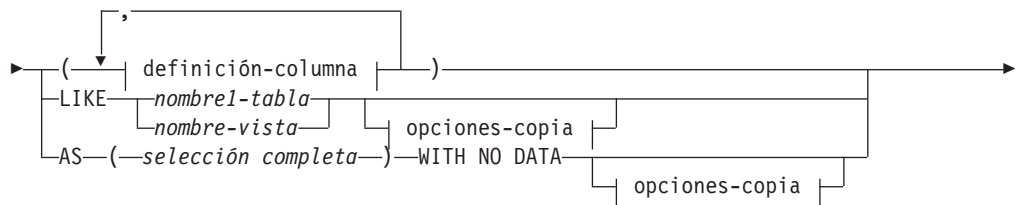
- Uno de los privilegios y autorizaciones siguientes:
 - Privilegio USE en el espacio de tablas
 - SYSADM
 - SYSCTRL
- Más uno de estos privilegios y autorizaciones:
 - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla no existe
 - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla hace referencia a un esquema existente

Cuando se define una tabla utilizando LIKE o una selección completa, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes para cada tabla o vista identificada:

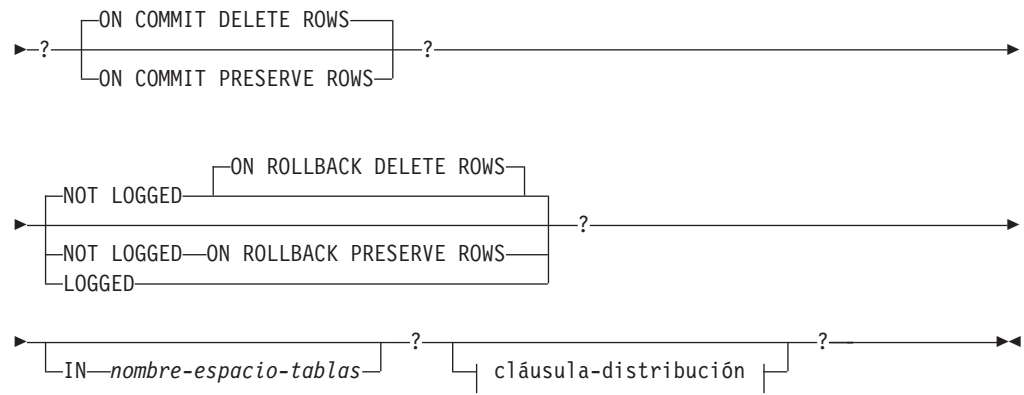
- Privilegio SELECT para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Autorización DATAACCESS

Sintaxis

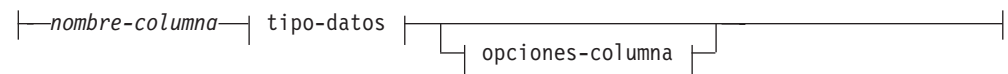
►► CREATE GLOBAL TEMPORARY TABLE *nombre-tabla* ►►



CREATE GLOBAL TEMPORARY TABLE



definición-columna:

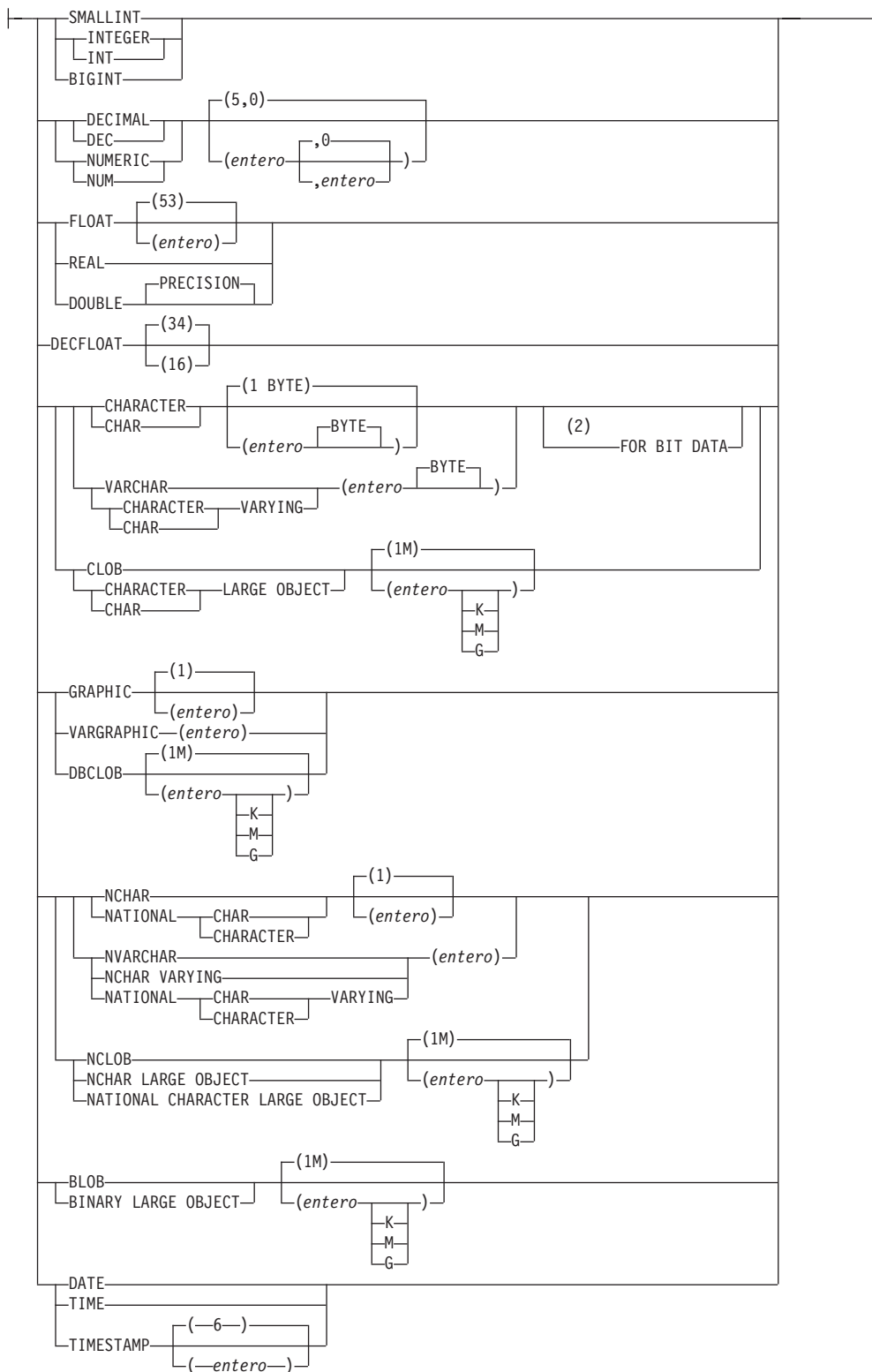


tipo-datos:

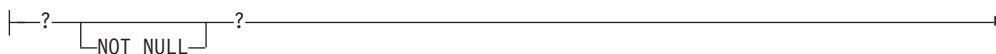


tipo-incorporado:

CREATE GLOBAL TEMPORARY TABLE



opciones-columna:



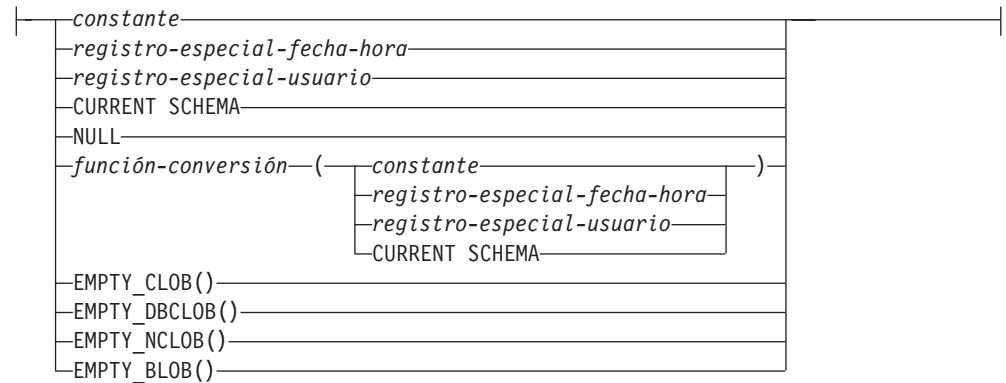
CREATE GLOBAL TEMPORARY TABLE



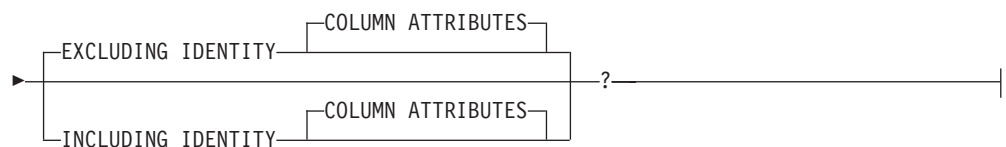
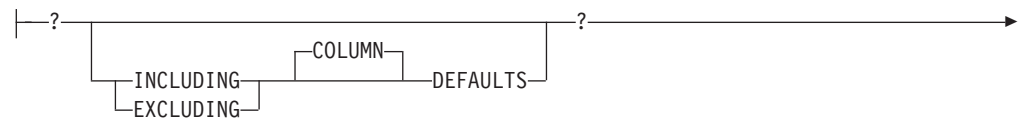
cláusula-por-omisión:



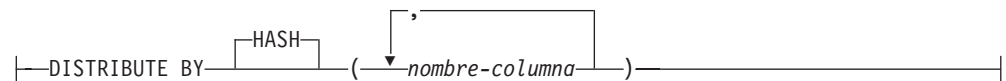
valores-omisión:



opciones-copia:



cláusula-distribución:



Notas:

- 1 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados.
- 2 La cláusula FOR BIT DATA se puede especificar en cualquier orden con las restricciones de columna siguientes.

CREATE GLOBAL TEMPORARY TABLE

Descripción

nombre-tabla

Indica el nombre de la tabla. El nombre, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo ni alias descrito en el catálogo. Si se especifica un nombre compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS' (SQLSTATE 42939).

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 puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no debe ser superior a 4.005.
- Un tamaño de página de 8K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 8.101.
- Un tamaño de página de 16K con un máximo de 1.012 columnas, donde el número total 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, donde el número total de bytes de las columnas no debe ser superior a 32.677.

Una tabla temporal creada no puede tener una columna de inicio de fila, una columna de fin de fila o una columna de ID de inicio de transacción.

Para ver más detalles, consulte el apartado “Tamaño de fila” en “CREATE TABLE”.

tipo-datos

Especifica el tipo de datos de la columna.

tipo-incorporado

Especifica un tipo de datos incorporado. Consulte “CREATE TABLE” para obtener una descripción de *tipo-incorporado*.

No se puede especificar un tipo de datos XML y SYSPROC.DB2SECURITYLABEL para una tabla temporal creada.

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 (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico).

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 de fuente del tipo diferenciado. El tipo diferenciado para una columna no puede tener ninguna restricción de tipo de datos, y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2).

opciones-columna

Define otras opciones relacionadas con las columnas de la tabla.

NOT NULL

Evita que la columna contenga valores nulos. Para obtener información acerca de la especificación de valores nulos, consulte NOT NULL en "CREATE TABLE".

cláusula-por-omisión

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 "ALTER TABLE".

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-por-omisión* (SQLSTATE 42842).

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 normas de asignación
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- ser una constante numérica o un valor especial de coma flotante decimal si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y, a continuación, se convierten a coma flotante decimal, si la columna de destino es DECFLOAT. Para las columnas DECFLOAT(16), las constantes decimales que tengan una precisión de más de 16 dígitos se redondearán utilizando las modalidades de redondeo especificadas por el registro especial CURRENT DECFLOAT ROUNDING MODE.

CREATE GLOBAL TEMPORARY TABLE

- no tener dígitos que no sean cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal (por ejemplo, 1.234 no puede ser el valor por omisión de una columna DECIMAL(5,2)).
- estar expresada con un máximo de 254 bytes, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificados y paréntesis, cuando la constante es el argumento de una *función-conversión*

registro-especial-fecha-hora

Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de 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).

registro-especial-usuario

Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER.

CURRENT SCHEMA

Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud superior o igual al atributo de longitud del registro especial CURRENT SCHEMA.

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 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 esta debe coincidir con el nombre del tipo de fuente del tipo diferenciado,

CREATE GLOBAL TEMPORARY TABLE

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 indicación 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 normas de una constante para el tipo de 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 serie de caracteres.

registro-especial-fecha-hora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario

Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de 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.

CURRENT SCHEMA

Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()

Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

IDENTITY y opciones-identidad

Para obtener información acerca de la especificación de columnas de identidad, consulte IDENTITY y *opciones-identidad* en "CREATE TABLE".

LIKE nombre1-tabla 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 *nombre-tabla-1*), la vista (*nombre-vista*) o el apodo (*apodo*) identificados. 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). No puede especificarse una tabla protegida (SQLSTATE 42962). No puede especificarse una tabla que tenga una columna definida como IMPLICITLY HIDDEN (SQLSTATE 560AE).

CREATE GLOBAL TEMPORARY TABLE

El uso de LIKE es una definición implícita de n columnas, donde n es el número de columnas de la tabla (incluidas las columnas implícitamente ocultas), la vista o el apodo identificados. La definición implícita depende de lo que se identifica después de LIKE.

- 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-tabla-1*. 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*. Los tipos de datos de las columnas de la vista deben ser tipos de datos válidos para las columnas de una tabla.
- Si se designa un 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, activadores, índices, claves de particionamiento de tablas ni claves de distribución. 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.

Cuando una tabla se identifica en la cláusula LIKE y dicha tabla contiene una columna ROW CHANGE TIMESTAMP, la columna correspondiente de la nueva tabla hereda sólo el tipo de datos de la columna ROW CHANGE TIMESTAMP. La nueva columna no se considera una columna generada.

Si el control de acceso de nivel de fila o columna (RCAC) se implanta para *nombre1-tabla*, la nueva tabla no hereda RCAC.

AS (*selección-completa*) WITH NO DATA

Especifica que las columnas de la tabla tienen el mismo nombre y descripción que las columnas que aparecerían en la tabla de resultados derivada de la selección completa si esta tuviera que ejecutarse. El uso de AS (*selección completa*) es una definición implícita de n columnas de la tabla temporal creada, donde n es el número de columnas resultantes de la selección completa.

La definición implícita incluye los atributos siguientes de las n columnas (si son aplicables al tipo de datos):

- Nombre de columna
- Tipo de datos, longitud, precisión y escala
- Capacidad de nulos

Los atributos siguientes no se incluyen (el valor por omisión y los atributos de identidad pueden incluirse utilizando las *opciones-copia*):

- Valor por omisión
- Atributos de identidad
- Atributo oculto
- ROW CHANGE TIMESTAMP

La definición implícita no incluye ningún otro atributo opcional de las tablas o vistas a las que se hace referencia en la selección completa.

Cada elemento de la lista de selección debe tener un nombre exclusivo (SQLSTATE 42711). La cláusula AS puede utilizarse en la cláusula select para proporcionar nombres exclusivos. La selección completa no debe hacer referencia a variables del lenguaje principal ni incluir marcadores de parámetro. Los tipos de datos de las columnas de resultados de la selección completa deben ser tipos de datos válidos para las columnas de una tabla.

Si se implanta el control de acceso de nivel de fila o columna para cualquier tabla que se especifique en *selección completa*, RCAC no se aplica en cascada a la nueva tabla.

opciones-copia

Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados fuente (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 de resultados fuente. 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-1*, y *nombre-tabla-1* identifica una tabla base, una tabla temporal creada o una tabla temporal declarada, INCLUDING COLUMN DEFAULTS será el valor por omisión.

EXCLUDING COLUMN DEFAULTS

Especifica que no deben copiarse los valores por omisión de columnas contenidos en la definición de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica una tabla base, una tabla temporal creada o una tabla temporal declarada, entonces INCLUDING COLUMN DEFAULTS es el valor por omisión.

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 de resultados fuente. Es posible copiar estos atributos si el elemento de la columna correspondiente en la tabla, la vista o la selección completa es el nombre de una columna de una tabla, o el nombre de una columna de una vista que, directa o indirectamente, se correlaciona con el nombre de columna de una tabla base o de una tabla temporal creada que tiene la propiedad 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)
- 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).

CREATE GLOBAL TEMPORARY TABLE

EXCLUDING IDENTITY COLUMN ATTRIBUTES

Especifica que no deben copiarse los atributos de columna de identidad contenidos en la definición de la tabla de resultados fuente.

ON COMMIT

Especifica la acción que se realiza sobre la tabla temporal creada cuando se ejecuta una operación COMMIT. El valor por omisión es DELETE ROWS.

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.

PRESERVE ROWS

Las filas de la tabla se conservan.

LOGGED o NOT LOGGED

Especifica si se anotan cronológicamente las operaciones para la tabla. El valor por omisión es NOT LOGGED ON ROLLBACK DELETE ROWS.

NOT LOGGED

Especifica que las operaciones de inserción, actualización o supresión efectuadas en la tabla no deben anotarse cronológicamente, pero que la creación o descarte de la tabla sí debe anotarse cronológicamente. Durante una operación ROLLBACK (o ROLLBACK TO SAVEPOINT):

- Si la tabla se había creado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla se descarta
- Si la tabla se había descartado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla vuelve a crearse, pero sin datos

ON ROLLBACK

Especifica la acción que va a tener lugar en la tabla temporal creada sin anotaciones cronológicas cuando se realice una operación ROLLBACK (o ROLLBACK TO SAVEPOINT). El valor por omisión es DELETE ROWS.

DELETE ROWS

Si se han cambiado los datos de la tabla, se suprimirán todas las filas.

PRESERVE ROWS

Las filas de la tabla se conservan.

LOGGED

Especifica que las operaciones de inserción, actualización o supresión efectuadas en la tabla, así como la creación o descarte de la tabla, deben anotarse cronológicamente.

IN *nombre-espacio-tablas*

Identifica el espacio de tablas en el que se creará una instancia de la tabla temporal creada. El espacio de tablas debe existir y estar definido como USER TEMPORARY (SQLSTATE 42838), para el cual el ID de autorización 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 que pertenezca 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 ningún espacio de tablas USER TEMPORARY elegible, se produce un error (SQLSTATE 42727).

La determinación del espacio de tablas puede cambiar cuando:

- Se descartan o crean espacios de tablas
- Se otorgan o revocan los 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. Para ver más detalles, consulte el apartado “Tamaño de fila” en “CREATE TABLE”.

cláusula-distribución

Especifica el particionamiento de base de datos o el modo en que se distribuyen los datos en diversas particiones de base de datos.

DISTRIBUTE BY HASH (*nombre-columna,...*)

Especifica la utilización de la función de hash por omisión en las columnas especificadas (denominada *clave de distribución*) como método de distribución en las particiones de base de datos. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No se puede utilizar como parte de una clave de distribución ninguna columna cuyo tipo de datos sea BLOB, CLOB, DBCLOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42962).

Si no se especifica esta cláusula y la tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de distribución se define como la primera columna cuyo tipo de datos es válido para una clave de distribución.

Si ninguna columna cumple los requisitos de una clave de distribución por omisión, la tabla se crea sin ninguna. Estas tablas solo están permitidas en espacios de tablas definidos en grupos de particiones de base de datos de partición única.

Para las tablas de los espacios de tablas que están definidos en grupos de particiones de base de datos de partición única, se puede utilizar cualquier conjunto de columnas con tipos de datos válidos para una clave de distribución a fin de definir la clave de distribución. Si no se especifica esta cláusula, no se crea ninguna clave de distribución.

Notas

- Debe existir un espacio de tablas temporal de usuario antes de crear una tabla temporal creada (SQLSTATE 42727).
- **Creación de instancias y terminación:** en la explicación siguiente, P representa una sesión y T es una tabla temporal creada de la sesión P:
 - Se crea una instancia vacía de T como resultado de la primera referencia a T que se ejecuta en P.
 - Cualquier sentencia de SQL en P puede hacer referencia a T, y cualquier referencia a T en P es una referencia a la misma instancia de T.
 - 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 T.

CREATE GLOBAL TEMPORARY TABLE

- Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye una modificación de T:
 - Si se especificó NOT LOGGED, se suprimen todas las filas desde T salvo que se hubiera especificado también ON ROLLBACK PRESERVE ROWS.
 - Si no se ha especificado NOT LOGGED, los cambios realizados en T se deshacen
- Si se especificó NOT LOGGED y una sentencia INSERT, UPDATE o DELETE falla durante su ejecución (en contraposición con un error de compilación), se suprimen todas las filas desde T.
- Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye la creación de T, la retrotracción incluye la operación DROP TABLE.T.
- Si una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye descartar la tabla temporal creada T, la retrotracción deshace el descarte de la tabla. Si se ha especificado NOT LOGGED, la tabla también se habrá vaciado.
- Cuando el proceso de aplicación que hizo referencia a T finaliza o se desconecta de la base de datos, se descarta la instancia privada de T y se destruyen las instancias de sus filas.
- Cuando finaliza la conexión con el servidor donde se hizo referencia a T, se descarta la instancia privada de T y se destruyen las instancias de sus filas.
- **Restricciones en el uso de tablas temporales creadas:** las tablas temporales creadas:
 - No se pueden especificar en las sentencias ALTER, LOCK o RENAME (SQLSTATE 42995).
 - No se pueden especificar en restricciones de referencia (SQLSTATE 42995).
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - DEFINITION ONLY puede especificarse en lugar de WITH NO DATA
 - La cláusula PARTITIONING KEY puede especificarse en lugar de la cláusula DISTRIBUTE BY

La sintaxis siguiente se acepta como comportamiento por omisión:

- CCSID ASCII
- CCSID UNICODE

Ejemplos

- *Ejemplo 1:* Crear una tabla temporal, CURRENTMAP. Asignar un nombre a dos columnas, CODE y MEANING, ninguna de las cuales puede contener nulos. CODE contiene datos numéricos y MEANING tiene datos de caracteres.

```
CREATE GLOBAL TEMPORARY TABLE CURRENTMAP
(CODE          INTEGER      NOT NULL,
 MEANING       VARCHAR (254) NOT NULL)
```

- *Ejemplo 2:* Crear una tabla temporal, TMPDEPT.

```
CREATE GLOBAL TEMPORARY TABLE TMPDEPT
(TMPDEPTNO    CHAR(3)      NOT NULL,
 TMPDEPTNAME  VARCHAR(36)  NOT NULL,
 TMPMGRNO     CHAR(6),
 TMPLOCATION   CHAR(16) )
```

CREATE HISTOGRAM TEMPLATE

La sentencia CREATE HISTOGRAM TEMPLATE define una plantilla que describe el tipo de histograma que puede utilizarse para alterar temporalmente uno o más histogramas por omisión de una clase de servicio o de una clase de trabajo.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Sintaxis

```
► CREATE HISTOGRAM TEMPLATE nombre-plantilla
► HIGH BIN VALUE constante-bigint
```

Descripción

nombre-plantilla

Da nombre a la plantilla del histograma. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar una plantilla de histograma existente en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

HIGH BIN VALUE *constante-bigint*

Especifica el valor superior desde el segundo binario hasta el último (el último binario tiene un valor superior no vinculado). Las unidades dependen del modo en que se utiliza el histograma. El valor máximo es 268 435 456.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)

CREATE HISTOGRAM TEMPLATE

- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplo

Cree una plantilla de histograma denominada LIFETIMETEMP en la clase de servicio PAYROLL de la superclase de servicio ADMIN que altere temporalmente la plantilla de histograma de vida útil de actividad por omisión especificando el nuevo valor de intervalo alto 90.000, que representa 90.000 milisegundos. Esta acción producirá un histograma con rangos de intervalos que se incrementarán exponencialmente y que finalizará con un intervalo cuyo rango irá de 90 000 al infinito.

```
CREATE HISTOGRAM TEMPLATE LIFETIMETEMP  
HIGH BIN VALUE 90000
```

```
CREATE SERVICE CLASS PAYROLL  
UNDER ADMIN ACTIVITY LIFETIME HISTOGRAM TEMPLATE LIFETIMETEMP
```


CREATE INDEX

La sentencia CREATE INDEX se utiliza para definir un índice en una tabla DB2. Un índice puede definirse en datos XML o en datos relacionales. La sentencia CREATE INDEX también se utiliza para crear una especificación de índice (metadatos que indican al optimizador que una tabla de fuente de datos tiene un índice).

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

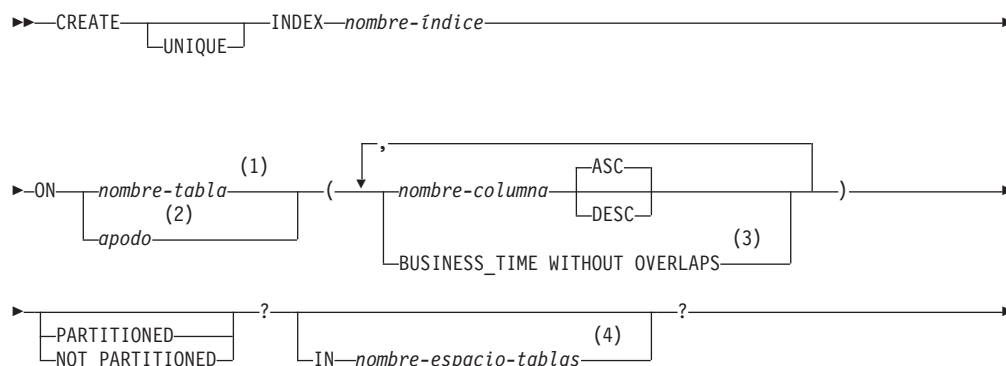
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

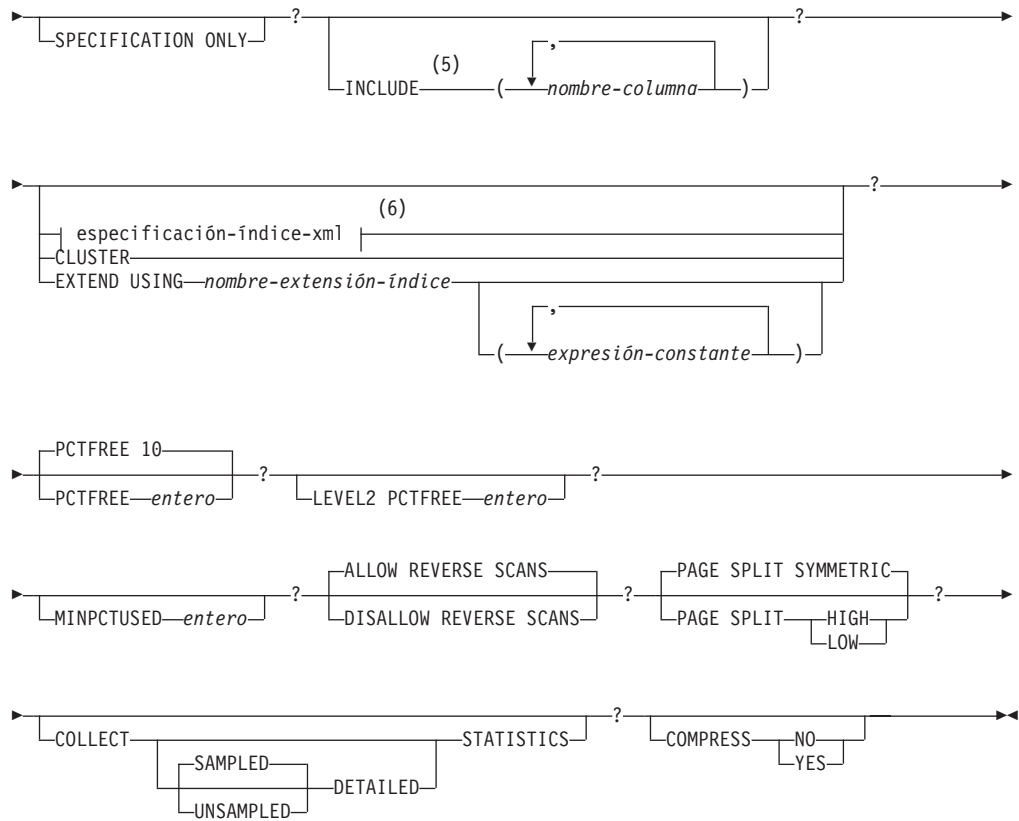
- Uno de los siguientes:
 - El privilegio CONTROL sobre la tabla o el apodo en el que está definido el índice
 - El privilegio INDEX en la tabla o el apodo en el que está definido el índice
- y uno de estos:
 - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito del índice no existe
 - Privilegio CREATEIN para el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Autorización DBADM

No se necesita ningún privilegio explícito para crear un índice en una tabla temporal declarada.

Sintaxis



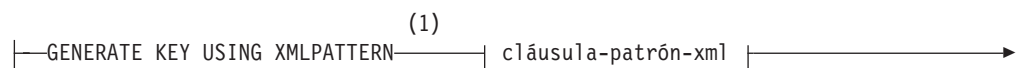
CREATE INDEX



Notas:

- 1 En un sistema federado, *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 crea una especificación de índice. En este caso, no se pueden especificar INCLUDE, *especificación-índice-xml*, CLUSTER, EXTEND USING, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, ALLOW REVERSE SCANS, PAGE SPLIT o COLLECT STATISTICS.
- 3 La cláusula BUSINESS_TIME WITHOUT OVERLAPS solamente se puede especificar si se especifica UNIQUE.
- 4 La cláusula IN *nombre-espacio-tablas* solamente se puede especificar para un índice no particionado en una tabla particionada.
- 5 La cláusula INCLUDE solamente se puede especificar si se especifica UNIQUE.
- 6 Si se especifica *especificación-índice-xml*, no se puede especificar *nombre-columna* DESC, INCLUDE o CLUSTER.

especificación-índice-xml:



▶ cláusula-tipo-xml

Notas:

- 1 Puede utilizarse la sintaxis alternativa GENERATE KEYS USING XMLPATTERN.

cláusula-patrón-xml:

' declaraci3n-espacio-nombres expresi3n-patr3n '

declaraci3n-espacio-nombres:

DECLARE NAMESPACE *prefijo-espacio-nombres=uri-espacio-nombres* ;
 DECLARE DEFAULT ELEMENT NAMESPACE *uri-espacio-nombres*

expresi3n-patr3n:

/ eje-avance */* prueba-nombre-xml prueba-tipo-xml
// eje-avance prueba-nombre-xml paso-funci3n

eje-avance:

child::
 @
 attribute::
 descendant::
 self::
 descendant-or-self::

prueba-nombre-xml:

nombre-calificado-xml
comod3n-xml

comod3n-xml:

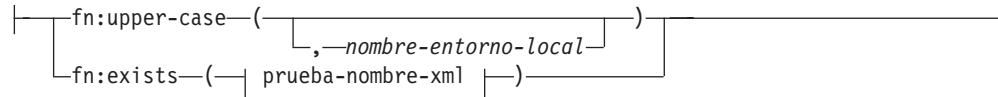
*
*prefijo-ns-xml:**
 *:nombre-nc-xml

CREATE INDEX

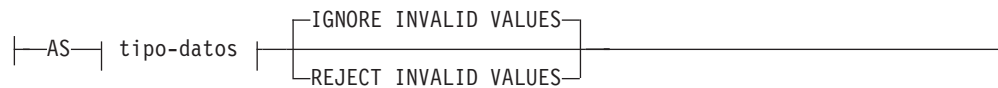
prueba-tipo-xml:



paso-función:



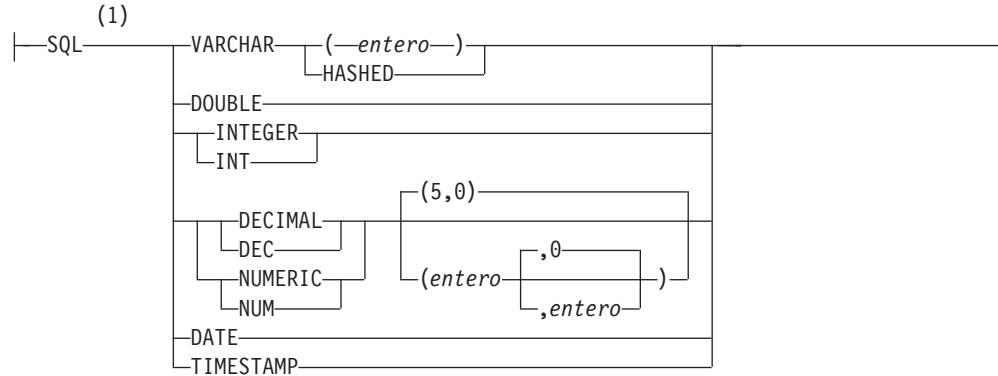
cláusula-tipo-xml:



tipo-datos:



tipo-datos-sql:



Notas:

- 1 Si especifica un nombre de función, como `fn:upper-case`, al final del patrón XML, los tipos de datos de índice soportados pueden ser un subconjunto de los tipos de datos de índice que se muestran aquí. Puede comprobar si hay tipos de datos de índice válidos en la descripción de cláusula-patrón-xml.

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.

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.

Si el índice se encuentra en una columna XML (se trata de un índice de datos XML), la unicidad se aplica a los valores para los que se ha especificado *expresión-patrón* para todas las filas de la tabla. La unicidad se aplica en cada valor después de convertir el valor al *tipo-datos-sql* especificado. Puesto que la conversión al *tipo-datos-sql* especificado puede dar lugar a una pérdida de precisión o de rango, o a que distintos valores se generen aleatoriamente en el mismo valor clave, varios valores que parecen ser exclusivos en el documento XML pueden dar lugar a errores de claves duplicadas. La unicidad de series de caracteres depende de la semántica de XQuery donde los blancos de cola son significativos. Por lo tanto, los valores que serían duplicados en SQL pero que difieren en blancos de cola se consideran valores exclusivos en un índice sobre datos XML.

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 distribución, las columnas de la clave de índice deben ser un superconjunto de la clave de distribución. Es decir, las columnas especificadas para una clave de índice exclusiva debe incluir todas las columnas de la clave de distribución (SQLSTATE 42997).

Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).

Si se especifica ON *apodo*, sólo se debe especificar UNIQUE si los datos para la clave de índice contienen valores exclusivos para cada fila de la tabla de fuente de datos. No se comprobará la exclusividad.

Para un índice sobre datos XML, UNIQUE sólo puede incluirse si el paso de contexto de la *expresión-patrón* especifica una única vía de acceso completa y no contiene un eje de tipo descendant o descendant-or-self, "/" /, un *comodín-xml*, *node()*, o *processing-instruction()* (SQLSTATE 429BS).

En un entorno de base de datos particionada, las siguientes normas se aplican a la tabla con una o más columnas XML:

- Una tabla redistribuida no puede tener un índice exclusivo sobre datos XML.
- Solo se da soporte a un índice exclusivo sobre datos XML en una tabla sin clave de distribución y que no es una base de datos de varias particiones.
- Si una tabla tiene un índice exclusivo sobre datos XML, la tabla no se puede modificar para añadir una clave de distribución.

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 una especificación de índice que se describa en el catálogo o un índice existente de una tabla temporal declarada (SQLSTATE 42704). El calificador no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

El calificador implícito o explícito de los índices de las tablas temporales globales declaradas debe ser SESSION (SQLSTATE 428EK).

ON *nombre-tabla* **o** *apodo*

El *nombre-tabla* identifica una tabla en la que va a crearse un índice. La tabla debe ser una tabla base (no una vista), una tabla temporal creada, una tabla

temporal declarada, una tabla de consulta materializada que exista en el servidor actual o una tabla temporal declarada. El nombre de una tabla temporal declarada debe calificarse con `SESSION`. El *nombre-tabla* no debe identificar una tabla de catálogo (`SQLSTATE 42832`). Si se especifica `UNIQUE` y *nombre-tabla* es una tabla con tipo, no debe ser una subtabla (`SQLSTATE 429B3`).

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 aparecer en la lista del catálogo.

nombre-columna

Para un índice, *nombre-columna* identifica una columna que debe 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 de fuente de datos.

Cada *nombre-columna* debe ser un nombre no calificado que identifique a una columna de la tabla. El número de columnas más dos veces el número de períodos identificados no puede superar 64 (`SQLSTATE 54008`). Si *nombre-tabla* es una tabla con tipo, el número de columnas no puede superar 63 (`SQLSTATE 54008`). Si *nombre-tabla* es una subtabla, como mínimo deberá especificarse un *nombre-columna* en la subtabla; es decir, no deberá heredarse de una supertabla (`SQLSTATE 428DS`). No puede repetirse ningún *nombre-columna* (`SQLSTATE 42711`).

La suma de las longitudes de las columnas especificadas no debe ser superior al límite de longitud de claves de índice para el tamaño de página. Para los límites de longitud de clave, consulte “Límites de SQL”. Si *nombre-tabla* es una tabla con tipo, la longitud de la clave de índice debe ser todavía 4 bytes menos. Tenga en cuenta que la actividad general del sistema puede reducir todavía más este límite de longitud, que varía en función del tipo de datos de la columna y de si la columna puede contener nulos. Para obtener más información sobre cómo afecta la actividad general a este límite, consulte la sección “Número total de bytes” en “CREATE TABLE”.

Tenga en cuenta que la actividad general del sistema puede reducir esta longitud, que varía en función del tipo de datos de la columna y en función de si puede contener nulos. Para obtener más información sobre cómo la actividad general afecta a este límite, consulte “Número total de bytes” en “CREATE TABLE”.

No e pueden utilizar columnas LOB o columnas de tipo diferenciado basadas en LOB como parte de un índice, incluso cuando el atributo de longitud de la columna es suficientemente pequeño para caber en el límite de longitud de la clave de índice para el tamaño de página (`SQLSTATE 54008`). 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 columna y el tipo de la columna debe ser un tipo estructurado o un tipo diferenciado no basado en LOB (`SQLSTATE 42997`).

Si un índice tiene sólo una columna y los datos de dicha columna son de tipo XML, y también se especifica la cláusula `GENERATE KEY USING XMLPATTERN`, el índice es un índice sobre datos XML. Una columna con datos de tipo XML sólo puede especificarse si también se especifica la cláusula `GENERATE KEY USING XMLPATTERN` (`SQLSTATE 42962`). Si se especifica la

cláusula `GENERATE KEY USING XMLPATTERN`, sólo se puede especificar una columna y el tipo de la columna debe ser XML.

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` o si el índice es un índice sobre datos XML (SQLSTATE 42601).

BUSINESS_TIME WITHOUT OVERLAPS

`BUSINESS_TIME WITHOUT OVERLAPS` sólo se pueden especificar para un índice definido como `UNIQUE` (SQLSTATE 428HW) a fin de indicar que, para el resto de las claves especificadas, los valores son exclusivos con respecto de cualquier período de tiempo. `BUSINESS_TIME WITHOUT OVERLAPS` sólo se puede especificar como último elemento de la lista. Cuando se especifica `BUSINESS_TIME WITHOUT OVERLAPS`, la columna final y la columna inicial del período `BUSINESS_TIME` se añaden automáticamente a la clave de índice en orden ascendente e impone que no haya solapamientos en el tiempo. Cuando se especifica `BUSINESS_TIME WITHOUT OVERLAPS`, las columnas del período `BUSINESS_TIME` no deben especificarse como columnas de clave, como columnas en la clave de particionamiento ni como columnas en la clave de distribución (SQLSTATE 428HW).

PARTITIONED

Indica que debe crearse un índice particionado. El *nombre-tabla* debe identificar una tabla definida con las particiones de datos (SQLSTATE 42601).

Si la tabla está particionada y no se especifica ni `PARTITIONED` ni `NOT PARTITIONED`, el índice se crea como particionado (con algunas excepciones). Se crea un índice no particionado en lugar de un índice particionado si se aplican alguna de las situaciones siguientes:

- Se especifique `UNIQUE` y la clave de índice no incluya todas las columnas de clave de particionamiento de tabla.
- Se crea un índice espacial.
- El índice se define sobre datos XML.

Un índice particionado con una definición que duplique la definición de un índice no particionado no se considera un índice duplicado. Para obtener más información, consulte el apartado “Normas” en la página 604 de este tema.

La palabra clave `PARTITIONED` no puede especificarse para los índices siguientes:

- Un índice en una tabla no particionada (SQLSTATE 42601)
- Un índice definido sobre datos XML (SQLSTATE 42613)
- Un índice único donde la clave de índice no incluya todas las columnas de clave de particionamiento de la tabla (SQLSTATE 42990)
- Un índice espacial (SQLSTATE 42997)

Un índice particionado no puede crearse en una tabla particionada que tenga tablas dependientes desenlazadas, por ejemplo, `MQT` (SQLSTATE 55019).

La colocación del espacio de tablas para una partición de índice del índice particionado viene determinada por las normas siguientes:

CREATE INDEX

- Si la tabla que se desea indexar se ha creado mediante la cláusula INDEX IN *opciones-espacio-tablas-partición* de la sentencia CREATE TABLE, la partición de índice se crea en el espacio de tablas especificado en esa cláusula INDEX IN.
- Si la sentencia CREATE TABLE para la tabla que se desea indexar no ha especificado la cláusula INDEX IN *opciones-espacio-tablas-partición*, el índice particionado de la partición de índice se crea en el mismo espacio de tablas que la partición de datos correspondiente que indexa.

La cláusula IN de la sentencia CREATE INDEX no está soportada para los índices particionados (SQLSTATE 42601). La cláusula INDEX IN *cláusulas-espacio-tablas* de la sentencia CREATE TABLE se pasa por alto para los índices particionados. Si se especifica BUSINESS_TIME WITHOUT OVERLAPS para la clave de índice, las columnas de clave de particionamiento no deben incluir la columna inicial o final del período BUSINESS_TIME (SQLSTATE 428HW).

NOT PARTITIONED

Indica que sólo debe crearse un índice no particionado que se expanda por todas las particiones de datos definidas para la tabla. El *nombre-tabla* debe identificar una tabla definida con las particiones de datos (SQLSTATE 42601).

Un índice no particionado con una definición que duplique la definición de un índice particionado no se considera un índice duplicado. Para obtener más información, consulte el apartado “Normas” en la página 604 de este tema.

La colocación del espacio de tablas para un índice no particionado viene determinada por las normas siguientes:

- Si se especifica la cláusula IN de la sentencia CREATE INDEX, el índice no particionado se coloca en el espacio de tablas especificado en esa cláusula IN.
- Si no se especifica la cláusula IN de la sentencia CREATE INDEX, las normas siguientes determinan la colocación del espacio de tablas del índice no particionado:
 - Si la tabla que se desea indexar se ha creado mediante la cláusula INDEX IN *cláusulas-espacio-tablas* de la sentencia CREATE TABLE, el índice no particionado se coloca en el espacio de tablas especificado en esa cláusula INDEX IN.
 - Si la tabla que se desea indexar se ha creado sin utilizar la cláusula INDEX IN *cláusulas-espacio-tablas* de la sentencia CREATE TABLE, el índice no particionado se crea en el espacio de tablas de la primera partición de datos visible o enlazada de la tabla. Ésta es la primera partición de la lista de particiones de datos almacenadas en la base de las especificaciones de rango. Además, el ID de autorización de la sentencia no será necesario para poseer el privilegio USE en el espacio de tablas por omisión.

IN *nombre-espacio-tablas*

Especifica el espacio de tablas en el que se crea el índice no particionado en una tabla particionada. Esta cláusula no puede especificarse para un índice particionado o un índice en una tabla no particionada (SQLSTATE 42601). Al especificar un espacio de tablas expresamente para el índice, se altera temporalmente la especificación realizada mediante una cláusula INDEX IN al crear la tabla.

El espacio de tablas especificado por *nombre-espacio-tablas* debe estar en el mismo grupo de particiones de base de datos que los espacios de tablas de datos para la tabla y gestionar el espacio del mismo modo que los demás

espacios de tablas de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE.

Si no se especifica la cláusula IN, el índice se crea en el espacio de tablas que se ha especificado con la cláusula INDEX IN en la sentencia CREATE TABLE. Si no se ha especificado ninguna cláusula INDEX IN, se utiliza el espacio de tablas de la primera partición de datos visibles o adjunta de la tabla. Ésta es la primera partición de la lista de particiones de datos almacenadas en la base de las especificaciones de rango. Si no se especifica la cláusula IN, el ID de autorización de la sentencia no será necesario para poseer el privilegio USE en el espacio de tablas por omisión.

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).

Si la especificación de índice se aplica a un índice que es exclusivo, DB2 no verifica si los valores de la columna de la tabla remota son exclusivos. Si los valores de la columna remota no son exclusivos, es posible que las consultas realizadas sobre el apodo que incluyan la columna del índice devuelvan datos incorrectos o errores.

Esta cláusula no se puede utilizar cuando se crea un índice en una tabla temporal creada o una tabla temporal declarada (SQLSTATE 42995).

INCLUDE

Esta palabra clave entra 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). Debe especificarse UNIQUE si se especifica INCLUDE (SQLSTATE 42613). 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.

Esta cláusula no se puede utilizar con tablas temporales creadas o tablas temporales declaradas (SQLSTATE 42995).

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. Las palabras clave ASC o DESC pueden especificarse a continuación de *nombre-columna*, pero no tienen ningún efecto en el orden.

INCLUDE no se puede especificar para índices definidos con EXTEND USING, si se ha especificado *apodo* o si el índice se ha definido en una columna XML (SQLSTATE 42601).

especificación-índice-xml

Especifica cómo se generan las claves de índice a partir de documentos XML que están almacenados en una columna XML. No se puede especificar *especificación-índice-xml* si hay más de una columna de índice o si la columna no tiene el tipo de datos XML.

Esta cláusula sólo se aplica a columnas XML (SQLSTATE 429BS).

GENERATE KEY USING XMLPATTERN *cláusula-patrón-xml*

Especifica las partes de un documento XML que deben indexarse. Los valores de patrón XML son los valores indexados generados por la *cláusula-patrón-xml*. El índice no soporta los nodos de tipo de datos de lista. Si un nodo es calificado por la *cláusula-patrón-xml* y existe un esquema XML que especifica que el nodo es de tipo de datos de lista, el nodo de tipo de datos de lista no se puede indexar (SQLSTATE 23526 para sentencias CREATE INDEX o SQLSTATE 23525 para sentencias INSERT y UPDATE).

cláusula-patrón-xml

Contiene una expresión de patrón que identifica los nodos que deben indexarse. Consta de una *declaración-espacio-nombres* opcional y de una *expresión-patrón* necesaria.

declaración-espacio-nombres

Si la expresión de patrón contiene nombres calificados, debe especificarse una *declaración-espacio-nombres* para definir prefijos de espacios de nombres. Puede definirse un espacio de nombres por omisión para nombres sin calificar.

DECLARE NAMESPACE *prefijo-espacio-nombres=uri-espacio-nombres*

Correlaciona el *prefijo-espacio-nombres*, que es un nombreNC, con el *uri-espacio-nombres*, que es un literal de cadena. La *declaración-espacio-nombres* puede contener varias correlaciones *prefijo-espacio-nombres-a-uri-espacio-nombres*. El *prefijo-espacio-nombres* debe ser exclusivo dentro de la lista de *declaración-espacio-nombres* (SQLSTATE 10503).

DECLARE DEFAULT ELEMENT NAMESPACE *uri-espacio-nombres*

Declara el URI de espacio de nombres por omisión para los tipos o los nombres de elemento sin calificar. Si no se ha declarado ningún espacio de nombres por omisión, los tipos y los nombres de elemento sin calificar no se encuentran en ningún espacio de nombres. Sólo se puede declarar un espacio de nombres por omisión (SQLSTATE 10502).

expresión-patrón

Especifica los nodos de un documento XML que están indexados. La *expresión-patrón* puede contener caracteres comodín (*). Es similar a una expresión de vía de acceso en XQuery, pero admite un subconjunto de lenguaje XQuery que DB2 también admite.

/ (barra inclinada)

Separa los pasos de la expresión de vía de acceso.

// (dos barras inclinadas)

Es la sintaxis abreviada para */descendant-or-self::node()/*. Si se especifica UNIQUE, no se puede utilizar *// (dos barras inclinadas)*.

eje-avance

child::

Especifica los hijos del nodo de contexto. Es el valor por omisión, si no se especifica ningún otro eje de avance.

@ Especifica los atributos del nodo de contexto. Ésta es la sintaxis abreviada del atributo::

attribute::

Especifica los atributos del nodo de contexto.

descendant::

Especifica los descendientes del nodo de contexto. No se puede utilizar *descendant::* si también se especifica UNIQUE.

self::

Especifica el nodo de contexto propiamente dicho.

descendant-or-self::

Especifica el nodo de contexto y los descendientes del nodo de contexto. No se puede utilizar *descendant-or-self::* si también se especifica UNIQUE.

prueba-nombre-xml

Especifica el nombre del nodo para el paso en la vía utilizando un nombre XML calificado (nombre-xml-c) o un comodín (comodín-xml).

nombre-nc-xml

Nombre XML según define XML 1.0. No puede incluir el carácter de dos puntos.

nombre-xml-c

Especifica un nombre XML calificado (conocido por QName) que puede tener dos formas posibles:

- *prefijo-ns-xml:nombre-nc-xml*, donde *prefijo-ns-xml* es un *nombre-nc-xml* que identifica un espacio de nombres del ámbito.
- *nombre-nc-xml*, que indica que el espacio de nombres por omisión debe aplicarse como *prefijo-ns-xml* implícito.

comodín-xml

Especifica un nombre-xml-c como un comodín que puede tener tres formas posibles:

- * (un único carácter de asterisco) indica cualquier nombre-xml-calificado
- *prefijo-ns-xml:** indica cualquier nombre-nc-xml dentro del espacio de nombres especificado
- **:nombre-nc-xml* indica un nombre XML específico dentro de cualquier espacio de nombres del ámbito

No se puede utilizar *comodín-xml* en el paso de contexto de una expresión de patrón si también se especifica UNIQUE.

prueba-tipo-xml

Utilice estas opciones para especificar con qué tipos de nodo coincide el patrón. Las opciones disponibles son:

node()

Coincide con cualquier nodo. No se puede utilizar *node()* si también se especifica UNIQUE.

text()

Coincide con cualquier nodo de texto.

comment ()

Coincide con cualquier nodo de comentario.

processing-instruction()

Coincide con cualquier nodo de instrucción de proceso. No se puede utilizar processing-instruction() si también se especifica UNIQUE.

paso-función

Utilice estas llamadas de función para especificar índices con propiedades especiales, como que no se tiene en cuenta las mayúsculas y minúsculas. Sólo se permite un paso de función por cada cláusula XMLPATTERN. Los pasos de función sólo se pueden aplicar en elementos o atributos. No se puede colocar ninguna opción *prueba-tipo-xml* justo antes del paso de función. La función no se puede utilizar en medio del XMLPATTERN y debe aparecer sólo en el paso final. Actualmente, sólo se da soporte a las funciones fn:upper-case y fn:exists.

Tenga en cuenta que, en lugar de especificar el prefijo fn: para el nombre de función, puede especificar otro espacio de nombres válido o puede completamente fn:.

fn:upper-case

Hace que los valores de índice se almacenen en formato de mayúsculas. El primer parámetro de fn:upper-case es obligatorio y debe ser una expresión de elemento de contexto (' . '); el segundo parámetro es opcional y es el entorno local. Si fn:upper-case aparece en el patrón, VARCHAR y VARCHAR HASHED son los únicos tipos de índice soportados.

fn:exists

Comprueba si existe un elemento o un elemento de atributo en el documento XML. Si el elemento existe, este predicado devuelve true. El parámetro de fn:exists es obligatorio y debe ser un elemento o un atributo. Si esta función se utiliza en la vía de acceso de índice, el tipo de índice debe definirse como VARCHAR(1).

cláusula-tipo-xml

AS tipo-datos

Especifica el tipo de datos a los que se convierten los valores indexados antes de almacenarlos. Los valores se convierten al tipo de datos de índice XML que corresponde al tipo de datos de índice SQL especificado.

Tabla 17. Tipos de datos de índice correspondientes

Tipo de datos de índice XML	Tipo de datos de índice SQL
xs:string	VARCHAR(<i>entero</i>), VARCHAR HASHED
xs:double	DOUBLE
xs:int	INTEGER
xs:decimal	DECIMAL
xs:date	DATE
xs:dateTime	TIMESTAMP

Para VARCHAR(*entero*) y VARCHAR HASHED, el valor se convierte a un valor xs:string utilizando la función XQuery fn:string. El atributo de longitud de VARCHAR(*entero*) se aplica como restricción al valor xs:string resultante. Un tipo de datos de índice SQL de VARCHAR HASHED aplica un algoritmo hash al valor xs:string resultante para generar un código hash que se inserta en el índice.

Para índices que utilizan los tipos de datos DOUBLE, DATE, INTEGER, DECIMAL y TIMESTAMP, el valor se convierte al tipo de datos de índice XML mediante la expresión de conversión de XQuery.

Si el índice es exclusivo, la unicidad del valor se aplica después de convertir el valor al tipo indexado.

tipo-datos

Se da soporte al tipo de datos siguiente:

tipo-datos-sql

Los tipos de datos SQL son:

VARCHAR(*entero*)

Si se especifica este formato de VARCHAR, DB2 utiliza *entero* como restricción. Si los nodos de documento que deben indexarse tienen valores superiores al *entero*, los documentos no se insertan en la tabla si el índice ya existe. Si el índice no existe, el índice no se crea. *entero* es un valor entre 1 y un máximo que depende del tamaño de la página. La Tabla 18 muestra el valor máximo para cada tamaño de página.

Tabla 18. Longitud máxima de los nodos del documento por tamaño de página

Tamaño de página	Longitud máxima del nodo de documento (bytes)
4 KB	817
8 KB	1841
16 KB	3889
32 KB	7985

La semántica XQuery se utiliza para comparaciones de series, donde los blancos de cola son significativos. Esto difiere de la semántica SQL, donde los blancos de cola no son significantes durante las comparaciones.

VARCHAR HASHED

Especifique VARCHAR HASHED para gestionar la indexación de series de caracteres de longitud arbitraria. La longitud de una serie indexada no tiene límite. DB2 genera un código hash de ocho bytes para toda la serie. Los índices que utilizan estas series de caracteres con códigos hash sólo pueden utilizarse para búsquedas de igualdad. La semántica XQuery se utiliza para comparaciones de igualdad de series, donde los blancos de cola son significativos. Esto difiere de la semántica SQL, donde los blancos de cola no son

significantes durante las comparaciones. El hash de la serie preserva la igualdad de la semántica de XQuery y no la semántica de SQL.

DOUBLE

Especifica que se utiliza el tipo de datos DOUBLE para indexar valores numéricos. Los tipos de decimal no definido y los enteros de 64 bits pueden perder precisión cuando se almacenan como un valor DOUBLE. Los valores para DOUBLE pueden incluir los valores numéricos especiales *NaN*, *INF*, *-INF*, *+0* y *-0*, aunque el tipo de datos de SQL DOUBLE no soporta estos valores.

INTEGER

Especifica que se utiliza el tipo de datos INTEGER para indexar valores XML. Tenga en cuenta que el tipo de datos de esquema XML *xs:integer* permite un mayor rango de valores que el tipo de datos SQL de entero. Si se encuentra un valor fuera de rango, se devuelve un error. Si un valor se ajusta al formato léxico de *xs:double*, pero no se ajusta al formato léxico de *xs:int*, por ejemplo 3.5, 3.0 o 3E1, también se devuelve un error.

DECIMAL(*entero*, *entero*)

Especifica que se utiliza el tipo de datos DECIMAL para indexar valores XML. El tipo DECIMAL adopta dos parámetros: *precisión* y *escala*. El primer parámetro, *precisión*, es una constante de entero con un valor incluido en el rango de 1 a 31 que especifica el número total de dígitos. El segundo parámetro, *escala*, es una constante de entero superior o igual a cero e inferior o igual al valor de la precisión. La *escala* especifica el número de dígitos situados a la derecha de la coma decimal.

Los dígitos no se truncan a partir del final de un número decimal. Se devuelve un error si el número de dígitos situados a la derecha del carácter separador de decimales es superior al valor de la escala. Además, se devuelve un error si el número de dígitos significativos situados a la izquierda del carácter decimal (la parte entera del número) es superior al valor de la precisión.

DATE

Especifica que se utiliza el tipo de datos DATE para indexar valores XML. Tenga en cuenta que el tipo de datos de esquema XML para *xs:date* permite un mayor rango de valores que la del tipo de datos de *xs:date* DB2 pureXML que corresponde al tipo de datos de SQL. Si se encuentra un valor fuera de rango, se devuelve un error.

TIMESTAMP

Especifica que se utiliza el tipo de datos TIMESTAMP para indexar valores XML. Tenga en cuenta que el tipo de datos de esquema XML para *xs:dateTime* permite un mayor rango de valores y precisión de segundos

fraccionarios que el tipo de datos `xs:dateTime` de DB2 `pureXML` que corresponde al tipo de datos de SQL. Si se encuentra un valor fuera de rango, se devuelve un error.

IGNORE INVALID VALUES

Especifica que los valores de patrón XML que no son formatos léxicos válidos para el tipo de datos XML de índice de destino se ignoran y que la sentencia `CREATE INDEX` no indexa los valores correspondientes de los documentos XML almacenados. Por omisión, los valores no válidos se ignoran. Durante las operaciones de inserción y actualización, los valores de patrón XML no válidos no se indexan, pero los documentos XML se siguen insertando en la tabla. No se genera ningún error o aviso, porque la especificación de estos tipos de datos no es una restricción en los valores de patrón XML (las expresiones `XQuery` que buscan el tipo de datos de índice XML específico no tendrán en cuenta estos valores).

Las reglas para establecer los valores de patrón XML que se pueden ignorar se determina mediante el tipo de datos SQL especificado.

- Si el tipo de datos SQL es `VARCHAR(entero)` o `VARCHAR HASHED`, los valores de patrón XML no se ignoran nunca porque cualquier secuencia de caracteres es válida.
- Si el tipo de datos SQL es `DOUBLE`, `DECIMAL` o `INTEGER`, los valores de patrón XML que no se ajusten al formato léxico del tipo de datos XML `xs:double` se ignoran. Si el tipo de datos SQL es `DECIMAL` o `INTEGER` y el valor de patrón XML se ajusta al formato léxico del tipo de datos XML `xs:double`, pero no al formato léxico de `xs:decimal` o `xs:int`, respectivamente, se devuelve un error. Por ejemplo, si el tipo de datos SQL es `INTEGER`, los valores de patrón XML `3.5`, `3.0` y `3e0` se ajustan al formato léxico de `xs:double`, pero devuelven un error (SQLSTATE 23525) porque no se ajustan al formato léxico de `xs:int`. Los valores de patrón XML del tipo `'A123'` o `'hello'` se ignoran para el mismo índice.
- Si el tipo de datos SQL es un tipo de datos de indicación de fecha y hora, cualquier valor de patrón XML que no se ajuste al formato léxico del tipo de datos XML correspondiente (`xs:date` o `xs:dateTime`) se ignora.

Si un valor de patrón XML no se ajusta al formato léxico adecuado, se devuelve un error si el valor se sale del espacio de valores del tipo de datos o supera la longitud o la precisión y la escala máxima del tipo de datos SQL especificado. Si el índice no existe, no se crea ningún índice (SQLSTATE 23526).

REJECT INVALID VALUES

Todos los valores de patrón XML deben ser válidos en el contexto de la definición léxica del tipo de datos XML de índice. Además del valor debe incluirse en el rango del espacio de valores del tipo de datos XML de índice. Consulte la sección sobre referencia relacionada más adelante para acceder a enlaces con información detallada sobre la definición léxica y el espacio de valores de cada tipo de datos. Por ejemplo, cuando se especifica la cláusula `REJECT INVALID VALUES`, si se crea un índice del tipo `INTEGER`,

CREATE INDEX

los valores de patrón XML del tipo 3.5, 3.0, 3e0, 'A123' y 'hello' devolverán un error (SQLSTATE 23525). Los datos XML no se insertan ni se actualizan en la tabla si el índice ya existe (SQLSTATE 23525). Si el índice no existe, no se crea ningún índice (SQLSTATE 23526).

CLUSTER

Especifica que el índice es el índice de clústeres de la tabla. El factor de clúster de un índice de clústeres 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 clústeres 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 clústeres en una tabla que esté definida para utilizar la modalidad APPEND (SQLSTATE 428D8).

CLUSTER no está permitido si se especifica *apodo* o si el índice es un índice sobre datos XML (SQLSTATE 42601). Esta cláusula no se puede utilizar con tablas temporales creadas, tablas temporales declaradas (SQLSTATE 42995) ni tablas agrupadas por clústeres de rangos (SQLSTATE 429BG).

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).

Esta cláusula no se puede utilizar con tablas temporales creadas o tablas temporales declaradas (SQLSTATE 42995).

Esta cláusula no se puede utilizar en un entorno DB2 pureScale (SQLSTATE 56038).

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). Esta cláusula no debe tener una longitud de más de 32.768 en la página de códigos de base de datos (SQLSTATE 22001).

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. Si se especifica un valor superior a 10, sólo se dejará un 10 por ciento de espacio libre en las páginas que no son hojas.

Si no se proporciona un valor explícito para PCTFREE y no se ha establecido **DB2_INDEX_PCTFREE_DEFAULT**, PCTFREE tendrá un valor por omisión de 10.

PCTFREE no está permitido si *apodo* está especificado (SQLSTATE 42601). Esta cláusula no se puede utilizar con tablas temporales creadas o tablas temporales declaradas (SQLSTATE 42995).

LEVEL2 PCTFREE *entero*

Especifica qué porcentaje de cada página de nivel 2 de índice se va a dejar como espacio libre cuando se cree el índice. El valor de *entero* entra en un rango que va de 0 a 99. Si LEVEL2 PCTFREE no está establecido, se deja un mínimo de 10 o PCTFREE por ciento de espacio libre en todas las páginas que no son hojas. Si LEVEL2 PCTFREE está establecido, se deja *entero* por ciento de espacio libre en páginas intermedias de nivel 2 y se deja un mínimo de 10 o *entero* por ciento de espacio libre en páginas intermedias de nivel 3 y superiores.

LEVEL2 PCTFREE no está permitido si se especifica *apodo* (SQLSTATE 42601). Esta cláusula no se puede utilizar con tablas temporales creadas o tablas temporales declaradas (SQLSTATE 42995).

MINPCTUSED *entero*

Indica si las páginas de hoja de índice deben fusionarse en línea y el umbral del porcentaje mínimo de espacio utilizado en una página de hoja de índice. Si, después de eliminarse una clave de una página de hoja de índice, el porcentaje de espacio utilizado en la página es el porcentaje de *entero* o está por debajo de este, se intentarán fusionar las claves restantes de esta 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. Se recomienda un valor de 50 o inferior por motivos de rendimiento. La especificación de esta opción influirá en el rendimiento de la actualización y de la supresión. La fusión sólo tiene lugar durante las operaciones de actualización y supresión cuando existe un bloqueo de tabla exclusivo. Si no existe un bloqueo de tabla exclusivo, las claves se marcan como claves a las que se ha aplicado una supresión falsa durante las operaciones de actualización y supresión y no se realiza ninguna fusión. Considere la utilización de la opción CLEANUP ONLY ALL de REORG INDEXES para fusionar páginas de hoja en lugar de utilizar la opción MINPCTUSED de CREATE INDEX.

MINPCTUSED no está permitido si *apodo* está especificado (42601). Esta cláusula no se puede utilizar con tablas temporales creadas o tablas temporales declaradas (SQLSTATE 42995).

DISALLOW REVERSE SCANS

Especifica que el índice sólo permite búsquedas o exploraciones hacia delante del índice en el orden definido en el momento de crear el índice.

DISALLOW REVERSE SCANS no puede especificarse junto con *apodo* (SQLSTATE 42601).

ALLOW REVERSE SCANS

Especifica que un índice permite búsquedas hacia delante y hacia atrás; es decir, permite explorar el índice en el orden definido en el momento de su creación y en el orden opuesto.

ALLOW REVERSE SCANS no puede especificarse junto con *apodo* (SQLSTATE 42601).

PAGE SPLIT

Especifica un comportamiento de división de índice. El valor por omisión es SYMMETRIC.

CREATE INDEX

SYMMETRIC

Especifica que las páginas se deben dividir por la mitad.

HIGH

Especifica un comportamiento de división de páginas de índice que utiliza espacio de páginas de índice de forma eficiente cuando los valores de las claves de índice que se insertan siguen un determinado patrón. Para un subconjunto de valores de claves de índices, la columna o columnas que hay más a la izquierda del índice deben contener el mismo valor y la columna o columnas que hay más a la derecha del índice deben contener valores que aumentan con cada inserción.

LOW

Especifica un comportamiento de división de páginas de índice que utiliza espacio de páginas de índice de forma eficiente cuando los valores de las claves de índice que se insertan siguen un determinado patrón. Para un subconjunto de valores de claves de índices, la columna o columnas que hay más a la izquierda del índice deben contener el mismo valor y la columna o columnas que hay más a la derecha del índice deben contener valores que disminuyen con cada inserción.

COLLECT STATISTICS

Especifica que van a recopilarse estadísticas de índice básicas durante la creación del índice.

SAMPLED

Especifica que se debe utilizar una técnica de muestreo cuando se procesen entradas de índice para recopilar estadísticas de índice ampliadas. Esta opción se utiliza para equilibrar las consideraciones sobre rendimiento con la necesidad de precisión de las estadísticas. Esta opción es el valor por omisión cuando se especifica DETAILED inmediatamente después de la palabra clave COLLECT.

UNSAMPLED

Especifica que no se debe utilizar el muestreo cuando se procesen entradas de índice para recopilar estadísticas de índice ampliadas. Cada entrada de índice, en cambio, se examina de forma individual. Esta opción puede aumentar considerablemente el consumo de CPU y memoria.

DETAILED

Especifica que también van a recopilarse estadísticas de índice ampliadas (CLUSTERFACTOR y PAGE_FETCH_PAIRS) durante la creación del índice.

COMPRESS

Especifica si la compresión de índice está habilitada. Por omisión, la compresión de índice se habilitará si la compresión de filas de datos está habilitada, y se deshabilitará si está deshabilitada. Esta opción se puede utilizar para alterar temporalmente el comportamiento por omisión. COMPRESS no está permitido si *apodo* está especificado (SQLSTATE 42601).

YES

Especifica que la compresión de índice está habilitada. Las operaciones de inserción y actualización del índice estarán sujetas a compresión.

NO Especifica que la compresión de índice está inhabilitada.

Normas

- La sentencia CREATE INDEX fallará (SQLSTATE 01550) cuando se intenta crear un índice que coincida con un índice ya existente.

Los factores siguientes se utilizan para determinar si dos índices coinciden. Estos factores se combinan de varias maneras diferentes en las normas que determinan si dos índices coinciden. Los factores siguientes se utilizan para determinar si dos índices coinciden:

1. Los conjuntos de columnas de índice, incluida cualquier columna INCLUDE, son iguales en ambos índices.
2. El orden de las columnas de claves de índice, incluida cualquier columna INCLUDE, es el mismo en ambos índices.
3. Las columnas de claves del nuevo índice son las mismas o un superconjunto de las columnas de claves en el índice existente.
4. El orden de los atributos de las columnas es el mismo en ambos índices.
5. El índice existente es exclusivo.
6. Ambos índices no son únicos.

Las combinaciones siguientes de estos factores forman las normas que determinan cuando dos índices se consideran duplicados:

- 1 + 2 + 4 + 5
- 1 + 2 + 4 + 6
- 1 + 2 + 3 + 5

Excepciones:

- Si uno de los índices comparados está particionado y el otro no está particionado, los índices no se consideran duplicados si tienen nombres distintos, aunque se cumplan otras condiciones de índices coincidentes.
- Para los índices de datos XML, dos descripciones de índice no se consideran duplicadas si los nombres del índice son distintos, aun cuando la columna XML indexada, los patrones XML y los tipos de datos, incluidas sus opciones, sean idénticos.
- Los índices exclusivos de las MQT mantenidas por el sistema no reciben soporte (SQLSTATE 42809).
- Las opciones COLLECT STATISTICS no reciben soporte si se ha especificado un apodo (SQLSTATE 42601).

Notas

- El acceso de lectura/grabación simultáneo durante la creación de índices y el comportamiento de creación de índices por omisión es distinto para los índices de tablas no particionadas, los índices no particionados, los índices particionados y los índices de un entorno DB2 pureScale:
 - En el caso de índices no particionados, se permite el acceso de lectura/grabación simultáneo a la tabla mientras se está creando un índice, salvo cuando se especifica una cláusula EXTEND USING. Una vez creado el índice, los cambios que se han realizado en la tabla durante la creación del índice se reenvían e incorporan al nuevo índice. El acceso de grabación a la tabla queda bloqueado brevemente mientras se completa la creación del índice, tras lo cual, el nuevo índice estará disponible.
 - En el caso de índices particionados, se permite el acceso de lectura/grabación simultáneo a la tabla mientras se está creando un índice, salvo cuando se especifica una cláusula EXTEND USING. Una vez creada la partición de índice, los cambios que se han realizado en la partición durante la creación de esa partición de índice se reenvían e incorporan a la nueva partición de índice. El acceso de grabación a la partición de datos queda bloqueado mientras se completa la creación del índice en el resto de particiones de datos.

CREATE INDEX

Después de crear la partición de índice para la última partición de datos y confirmar la transacción, todas las particiones de datos están disponibles para acceso de lectura y grabación.

- En un entorno DB2 pureScale, el acceso de lectura simultáneo es el comportamiento por omisión. No se permite el acceso de grabación simultáneo durante la creación de índices.

Para eludir este comportamiento por omisión, utilice la sentencia LOCK TABLE para bloquear explícitamente la tabla antes de emitir una sentencia CREATE INDEX. (La tabla puede bloquearse en modalidad SHARE o EXCLUSIVE, en función de si va a estar permitido el acceso de lectura.)

- 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.
 - b
 - Una vez que se ha creado el índice y se han cargado los datos en la tabla, es aconsejable emitir el mandato 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 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. Si se habían cargado datos antes de la emisión de la sentencia CREATE INDEX, se recomienda utilizar la opción COLLECT STATISTICS de la sentencia CREATE INDEX como alternativa a la utilización del mandato RUNSTATS.
 - La creación de un índice 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
 - El optimizador puede recomendar índices antes de crear el índice real.
 - 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 en las que se aplican las especificaciones.
 - *Recopilación de estadísticas de índice:* la opción UNSAMPLED DETAILED está disponible para cambiar el modo en que se recopilan las estadísticas de índice. No obstante, sólo debe utilizarse en los casos donde quede claro que DETAILED no genera estadísticas precisas.
 - *Alternativas de sintaxis:* Se tolera y se pasa por alto la sintaxis siguiente:
 - CLOSE
 - DEFINE
 - FREEPAGE
 - GBPCACHE
 - PIECESIZE
 - TYPE 2
 - utilizando-bloque
- La sintaxis siguiente se acepta como comportamiento por omisión:
- COPY NO
 - DEFER NO

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 deben 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)
```

- *Ejemplo 3:* El apodo EMPLOYEE hace referencia a una tabla de fuente de datos denominada CURRENT_EMP. Tras la creación de este apodo, se ha definido 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:* Cree un tipo de índice ampliado denominado SPATIAL_INDEX en una ubicación de columna de tipo estructurado. La descripción de la extensión de índice GRID_EXTENSION se utiliza para mantener SPATIAL_INDEX. El literal se proporciona a GRID_EXTENSION para crear el tamaño de cuadrícula de índice.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
EXTEND USING (GRID_EXTENSION (x'000100100010001000400010'))
```

- *Ejemplo 5:* Cree un índice denominado IDX1 en una tabla denominada TAB1 y recopile estadísticas de índice básicas en el índice IDX1.

```
CREATE INDEX IDX1 ON TAB1 (co11) COLLECT STATISTICS
```

- *Ejemplo 6:* Cree un índice denominado IDX2 en una tabla denominada TAB1 y recopile estadísticas de índice detalladas en el índice IDX2.

```
CREATE INDEX IDX2 ON TAB1 (co12) COLLECT DETAILED STATISTICS
```

- *Ejemplo 7:* Cree un índice denominado IDX3 en una tabla denominada TAB1 y recopile estadísticas de índice detalladas en el índice IDX3 utilizando el muestreo.

```
CREATE INDEX IDX3 ON TAB1 (co13) COLLECT SAMPLED DETAILED STATISTICS
```

- *Ejemplo 8:* Crea un índice exclusivo denominado A_IDX en una tabla particionada denominada MYNUMBERDATA en el espacio de tablas IDX_TBSP.

```
CREATE UNIQUE INDEX A_IDX ON MYNUMBERDATA (A) IN IDX_TBSP
```

- *Ejemplo 9:* Cree un índice no exclusivo denominado B_IDX en una tabla particionada denominada MYNUMBERDATA en el espacio de tablas IDX_TBSP.

```
CREATE INDEX B_IDX ON MYNUMBERDATA (B)
NOT PARTITIONED IN IDX_TBSP
```

- *Ejemplo 10:* Cree un índice sobre datos XML en una tabla denominada COMPANYINFO, que contiene una columna XML denominada COMPANYDOCS. La columna XML COMPANYDOCS contiene un número elevado de documentos XML parecidos al siguiente:

```
<company name="Empresa1">
  <emp id="31201" salary="60000" gender="Mujer">
    <name>
```

CREATE INDEX

```
<first>Laura</first>
<last>Brown</last>
</name>
<dept id="M25">
  Finanzas
</dept>
</emp>
</company>
```

Los usuarios de la tabla COMPANYINFO a menudo deben recuperar información de los empleados utilizando el ID de empleado. Un índice como el siguiente puede hacer que la recuperación de datos sea más eficiente.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
GENERATE KEY USING XMLPATTERN '/company/emp/@id'
AS SQL DOUBLE
```

- *Ejemplo 11:* Por lo general, el índice siguiente es equivalente desde el punto de vista lógico al índice creado en el ejemplo anterior, excepto que utiliza una sintaxis abreviada.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
GENERATE KEY USING XMLPATTERN '/child::company/child::emp/attribute::id'
AS SQL DOUBLE
```

- *Ejemplo 12:* Cree un índice en una columna denominada DOC, indexando solamente el título del manual como VARCHAR(100). Puesto que el libro del manual debe ser exclusivo entre todos los manuales, el índice también debe ser exclusivo.

```
CREATE UNIQUE INDEX MYDOCSIDX ON MYDOCS(DOC)
GENERATE KEY USING XMLPATTERN '/book/title'
AS SQL VARCHAR(100)
```

- *Ejemplo 13:* Cree un índice en una columna denominada DOC, indexando el número del capítulo como DOUBLE. Este ejemplo incluye declaraciones de espacios de nombres.

```
CREATE INDEX MYDOCSIDX ON MYDOCS(DOC)
GENERATE KEY USING XMLPATTERN
'declare namespace b="http://www.example.com/book/";
declare namespace c="http://acme.org/chapters";
/b:book/c:chapter/@number'
AS SQL DOUBLE
```

- *Ejemplo 14:* crear un índice exclusivo denominado IDXPROJEST en la tabla PROJECT e incluir la columna PRSTAFF para permitir el acceso sólo a índice de la información de dotación media de personal estimada.

```
CREATE UNIQUE INDEX IDXPROJEST ON PROJECT (PROJNO) INCLUDE (PRSTAFF)
```

SCREATE INDEX EXTENSION

La sentencia CREATE INDEX EXTENSION define un objeto de extensión para utilizar con índices de tablas que tienen columnas de tipo estructurado o de tipo diferenciado.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- 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
- Autorización DBADM

Sintaxis

```

▶ CREATE INDEX EXTENSION nombre-extensión-índice
  (
    (nombre1-parámetro-tipo1-datos)
  )
  mantenimiento-índice | búsqueda-índice

```

mantenimiento-índices:

```

  FROM SOURCE KEY (nombre2-parámetro-tipo2-datos)
  GENERATE KEY USING invocación-función-tabla

```

búsqueda-índice:

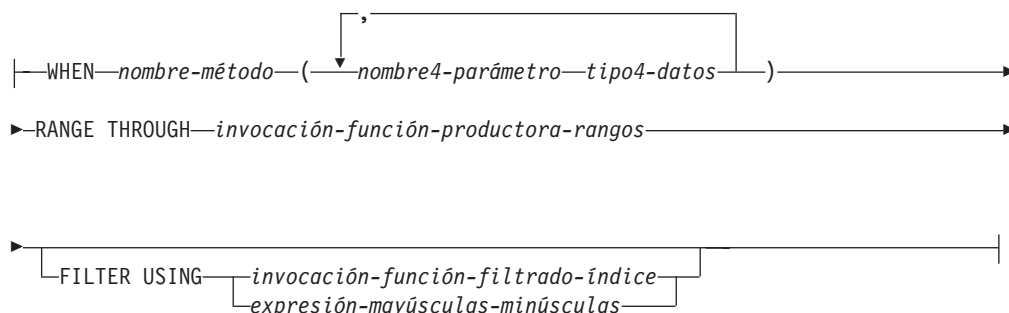
```

  WITH TARGET KEY (nombre3-parámetro-tipo3-datos)
  SEARCH METHODS definición-método-búsqueda

```

definición-método-búsqueda:

SCREATE INDEX EXTENSION



Descripción

nombre-extensión-índice

Designa la extensión de índice. El nombre (incluido el calificador implícito o explícito) no debe designar una extensión de índice descrita en el catálogo. Si se especifica un *nombre-extensión-índice* compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS'; de lo contrario se devuelve 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).

tipo1-datos

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 (SQLSTATE 429B5). No se puede especificar el tipo de datos de coma flotante decimal (SQLSTATE 429B5). 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 (*nombre2-parámetro tipo2-datos*)

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*.

tipo2-datos

Especifica el tipo de datos para *nombre-parámetro2*; *tipo2-datos* debe ser un tipo estructurado definido por el usuario o un tipo diferenciado que no se haya originado en LOB, XML o DECFLOAT (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 que se utiliza en *invocación-función-tabla* debe cumplir estas condiciones:

- Su resolución debe producir una función de tabla (SQLSTATE 428E4)
- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
- No debe estar definida como NOT DETERMINISTIC (SQLSTATE 428E4) ni como EXTERNAL ACTION (SQLSTATE 428E4)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).
- No debe tener un tipo de datos estructurado, LOB o XML (SQLSTATE 428E3) en el tipo de datos de los parámetros, con la excepción de métodos observadores generados por el sistema
- No debe incluir una subconsulta (SQLSTATE 428E3)
- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E3)
- 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).

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

SCREATE INDEX EXTENSION

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.

nombre3-parámetro

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 para cada *nombre-parámetro3* correspondiente. *tipo3-datos* 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.

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 de SQL que está asociado con el nombre de método especificado en la norma 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).

nombre4-parámetro

Identifica el parámetro de un argumento de búsqueda. Estos nombres se utilizan en las cláusulas RANGE THROUGH y FILTER USING.

tipo4-datos

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 que se utiliza en *invocación-función-producción-rango* debe cumplir estas condiciones:

- Su resolución debe producir una función de tabla (SQLSTATE 428E4)
- No debe incluir una subconsulta (SQLSTATE 428E3) ni una función SQL (SQLSTATE 428E4) en sus argumentos
- No debe incluir una expresión XMLQUERY o XMLEXISTS en sus argumentos (SQLSTATE 428E3)
- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)

- No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 428E4)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).

El número y los tipos de los resultados de esta función deben estar relacionados con los resultados de la función de tabla especificada en la cláusula GENERATE KEY USING (SQLSTATE 428E1) de la manera siguiente:

- Devolviendo un número máximo de columnas igual al doble de las devueltas por la función de transformación de claves
- Teniendo 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)
- Teniendo cada columna de clave de inicio el mismo tipo que la correspondiente columna de clave de parada
- Teniendo cada columna de clave de inicio el mismo tipo que la correspondiente columna de la función de transformación de claves

Más concretamente, $a_1:t_1, \dots, a_n:t_n$ serán 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 las columnas $m \leq n$ y "b" son las columnas de clave de inicio y las columnas "c" son las columnas de clave de detención.

Cuando la *invocación-función-productora-rangos* devuelve un valor nulo como valor de clave de inicio o de parada, la semántica no está definida.

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

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 que se utiliza en la *invocación-función-filtro-índice* debe cumplir las condiciones siguientes:

- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 42845)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).
- No debe tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- No debe incluir una subconsulta (SQLSTATE 428E3)

SCREATE INDEX EXTENSION

- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E3)

Si un argumento invoca otra función u otro método, estas normas también se aplican para la función o el 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 método utilizado como argumento), siempre que el resultado sea un argumento de tipo de datos incorporado.

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

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 normas especificadas en FILTER USING *invocación-función-filtrado-índice*. Cualquier función referenciada en la *expresión-case* debe también seguir las normas descritas para *invocación-función-filtrado-índice*. Además, no se pueden utilizar subconsultas ni expresiones XMLQUERY o XMLEXISTS 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; de lo contrario, ésta se descarta.

Notas

- La creación de una extensión de índice 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.

Ejemplo

El ejemplo siguiente crea una extensión de índice denominada *grid_extension* que utiliza una columna SHAPE de tipo estructurado en una función de tabla denominada *gridEntry* para generar siete claves de destino 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
```

SCREATE INDEX EXTENSION

```
SEARCHARG..MBR..YMAX < YMIN) THEN 0
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 MASK

La sentencia CREATE MASK crea una máscara de columna en el servidor actual. Una máscara de columna especifica el valor que ha de devolverse para una columna especificada.

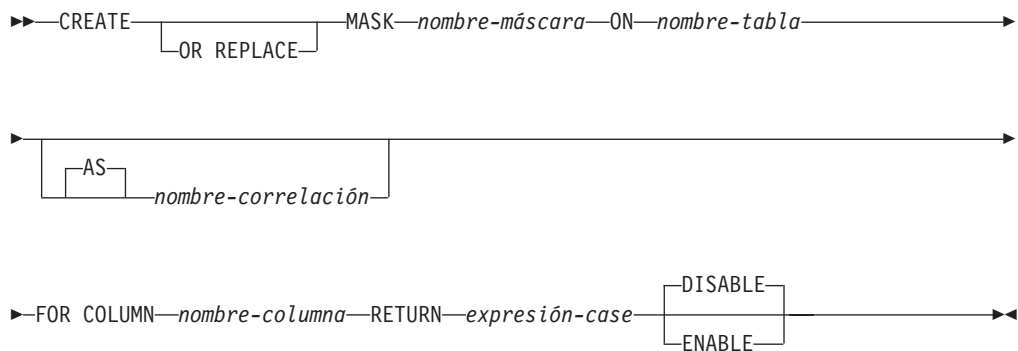
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES se especifica implícita o explícitamente.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM. La autorización SECADM puede crear una máscara de columna en cualquier esquema. No son necesarios privilegios adicionales para hacer referencia a otros objetos en la definición de máscara. Por ejemplo, no se necesita el privilegio SELECT para recuperar de una tabla, y no se necesita el privilegio EXECUTE para llamar a una función definida por el usuario.

Sintaxis



Descripción

OR REPLACE

Especifica que se debe sustituir la máscara de columna si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que se sustituya la nueva definición en el catálogo.

nombre-máscara

Asigna un nombre a la máscara de columna. El nombre, incluyendo el calificador implícito o explícito, no debe identificar una máscara de columna o un permiso de fila que ya existe en el servidor actual (SQLSTATE 42710).

nombre-tabla

Identifica la tabla en la que se crea la máscara de columna. El nombre debe identificar una tabla existente en el servidor actual (SQLSTATE 42704). No debe identificar apodos, tablas temporales creadas o declaradas, vistas, sinónimos, tablas con tipo, alias (SQLSTATE 42809) o tablas de catálogo (SQLSTATE 42832).

nombre-correlación

Especifica un nombre de correlación que se puede utilizar dentro de la *expresión-case* para designar la tabla.

FOR COLUMN *nombre-columna*

Identifica la columna a la que se aplica la máscara. *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). No debe existir ya una máscara para la columna (SQLSTATE 428HC). La columna no debe ser ninguna de las columnas siguientes:

- Una columna LOB o una columna de tipo diferenciado que se base en un LOB (SQLSTATE 42962).
- Una columna XML (SQLSTATE 42962).
- Una columna a la que se haga referencia en una expresión que define una columna generada (SQLSTATE 428HB).

RETURN *expresión-case*

Especifica una expresión CASE que se evaluará para determinar el valor que se devolverá a la columna (SQLSTATE 42601). El resultado de la expresión CASE se devuelve en lugar del valor de columna en una fila. El tipo de datos de resultado, el atributo nulo y el atributo de longitud de la expresión CASE deben ser idénticos a los de *nombre-columna* (SQLSTATE 428HB). Si el tipo de datos de *nombre-columna* es un tipo de datos definido por el usuario, el tipo de datos de resultado de la expresión CASE debe ser el mismo tipo de datos definido por el usuario. La expresión CASE no debe hacer referencia a ninguno de los objetos o elementos siguientes (SQLSTATE 428HB):

- Una tabla temporal global creada o una tabla temporal global declarada.
- Un apodo.
- Una función de tabla.
- Un método.
- Un marcador de parámetro (SQLSTATE 42601).
- Una función definida por el usuario que no se define como segura.
- Una función o expresión (como una expresión de cambio de fila o una expresión de secuencia) que no es determinante o que tiene una acción externa.
- Una función escalar XMLQUERY.
- Un predicado XMLEXISTS.
- Una especificación OLAP.
- Un * o name.* en una cláusula SELECT.
- Una pseudocolumna.
- Una función agregada sin especificar la cláusula SELECT.
- Una vista que incluya alguna de las restricciones incluidas anteriormente en su definición.

Si la expresión CASE hace referencia a tablas para las cuales el control de acceso de fila o columna está actualmente activado, el control de acceso de las tablas en cuestión no será en cascada. Consulte la sección Notas para obtener información detallada.

ENABLE o DISABLE

Especifica que la máscara de columna se debe habilitar o inhabilitar para el control de acceso a columnas. El valor por omisión es DISABLE.

DISABLE

Especifica que la máscara de columna se debe inhabilitar para el control de

CREATE MASK

acceso a columnas. Si el control de acceso a columnas de la tabla no está activado actualmente, la máscara de columna no será efectiva cuando se active el control de acceso a columnas de la tabla.

ENABLE

Especifica que la máscara de columna se debe habilitar para el control de acceso a columnas. Si el control de acceso a columnas de la tabla no está activado actualmente, la máscara de columna será efectiva cuando se active el control de acceso a columnas de la tabla. Si el control de acceso a columnas está activado actualmente para la tabla, la máscara de columna será efectiva inmediatamente y todos los paquetes y las sentencias en memoria caché dinámicas que hacen referencia a la tabla se invalidarán.

La aplicación de las máscaras de columna habilitadas no interfieren con las operaciones de las demás cláusulas dentro de la sentencia, como puede ser WHERE, GROUP BY, HAVING, SELECT DISTINCT y ORDER BY. Las filas devueltas en la tabla de resultados final no cambian, salvo que puede que las máscaras de columna hayan enmascarado los valores de las filas resultantes. Por lo tanto, si la columna enmascarada también aparece en una *clave-clasificación* ORDER BY, el orden se basa en los valores de columna originales y puede que los valores enmascarados de la tabla de resultados final no reflejen ese orden. Asimismo, es posible que los valores enmascarados no reflejen la exclusividad que SELECT DISTINCT aplica. Si la columna enmascarada está incluida en una expresión, el resultado de la expresión podría ser distinto porque la máscara de columna se aplica a la columna antes de que se realice la evaluación de la expresión. Por ejemplo, la aplicación de una máscara de columna en un SSN de columna podría cambiar el resultado de la función agregada COUNT(DISTINCT SSN) porque la operación DISTINCT se realiza en los valores enmascarados. Por otra parte, si la expresión de la consulta es la misma que la expresión utilizada para enmascarar el valor de columna en la definición de la máscara de columna, el resultado de la expresión podría no cambiar. Por ejemplo, la expresión de la consulta es 'XXX-XX-' || SUBSTR(SSN, 8, 4) y la misma expresión aparece en la definición de la máscara de columna. En este ejemplo concreto, sustituya la expresión de la consulta con un SSN de columna para evitar que la misma expresión se evalúe dos veces.

Una máscara de columna se crea como un objeto autónomo sin conocer todos los contextos en los que podría utilizarse. Para enmascarar un valor de columna en la tabla resultante final, la base de datos DB2 fusiona en una consulta la definición de máscara de columna. Cuando la definición de máscara de columna aparece en el contexto de la sentencia, es posible que entre en conflicto con determinada semántica de SQL de la sentencia. Por tanto, en algunas situaciones, la combinación de la sentencia y la aplicación de la máscara de columna podría devolver un error (SQLSTATE 428HD). Cuando esto ocurra, será necesario modificar la sentencia o descartar o volver a crear la máscara de columna con una definición diferente. Consulte la descripción de la sentencia ALTER TABLE para conocer las situaciones en las que podría emitirse un error de tiempo de vinculación para la sentencia.

Si la columna no es anulable, la definición de la máscara de columna no se considerará un valor nulo para la columna. Después de activar el control de acceso a columnas para la tabla de destino, si la tabla de destino es una tabla rellena con nulos en una operación de unión externa, el valor de columna en la tabla de resultados final podría ser un nulo. Para asegurarse de que la máscara de columna puede enmascarar un valor nulo, cuando la base de datos DB2 fusiona la definición de la máscara de columna en la consulta, si la tabla

de destino es la tabla rellena con nulos en una operación de unión externa, la base de datos DB2 añadirá "WHEN *columna-destino* IS NULL THEN NULL" como primera cláusula WHEN a la definición de la máscara de columna. Esto siempre forzará que un valor nulo se enmascare como nulo. Para una columna con posibilidad de nulos, esto elimina la posibilidad de enmascarar un valor nulo en otro valor, pero es una restricción aceptable desde el punto de vista de la seguridad y la funcionalidad.

Cuando se utiliza una columna para derivar el valor nuevo para una sentencia de asignación de *variable-transición* INSERT, UPDATE, MERGE o SET, se utilizará el valor de columna original, y no el valor enmascarado, para derivar un valor nuevo. Si la columna tiene una máscara de columna, esta se aplicará para asegurar que la evaluación de las reglas de control de acceso en tiempo de ejecución enmascara la propia columna y no una constante o de una expresión. De este modo se garantiza que los valores enmascarados sean los mismos que los valores de la columna original. Si una máscara de columna no enmascara la columna a sí misma, la fila existente no se actualizará o la fila nueva no se insertará y se devolverá un error en tiempo de ejecución (SQLSTATE 428HD). Si es necesario insertar los datos enmascarados en una tabla, se puede hacer asignando en primer lugar los datos a una variable. Por ejemplo, una variable de matriz se puede volver a crear con elementos de matriz que tengan un tipo de datos de fila. Los datos de tabla que tienen máscaras de columna aplicadas se pueden asignar a la variable de matriz, que se podrá utilizar para insertar los datos en otra tabla. Las reglas que se utilizan para aplicar máscaras de columna para derivar los nuevos valores siguen las mismas reglas descritas anteriormente para la tabla de resultados final de una consulta. Consulte las sentencias INSERT, UPDATE y MERGE para consultar cómo se utilizan las máscaras de columna para afectar a la operación de inserción y de actualización.

Consulte la sentencia ALTER TABLE con la cláusula ACTIVATE COLUMN ACCESS CONTROL para obtener información acerca de cómo activar el control de acceso a columnas para la tabla y cómo aplicar una máscara de columna.

Notas

- **Máscaras de columna que se han creado antes de activar el control de acceso a columnas para una tabla:** la sentencia CREATE MASK es una sentencia independiente que se puede utilizar para crear una máscara de control de acceso a columnas antes de activar el control de acceso a columnas para una tabla. El único requisito será que la tabla y las columnas existan antes de crear la máscara. Se pueden crear varias máscaras de columna para una tabla, pero una columna sólo puede tener una máscara.

La definición de una máscara se almacena en el catálogo de DB2. Se registrará la dependencia de la tabla para la cual se está creando la máscara y las dependencias de otros objetos a los que se hace referencia en la definición. No se invalidará ningún paquete o sentencia en memoria caché dinámicas. Se puede crear una máscara de columna como habilitada o inhabilitada para el control de acceso a columnas. Una máscara de columna habilitada no entra en vigor hasta que se utiliza la sentencia ALTER TABLE con la cláusula ACTIVATE COLUMN ACCESS CONTROL para activar el control de acceso a columnas para la tabla. Se necesita la autorización SECADM para emitir una sentencia ALTER TABLE. Una máscara de columna inhabilitada sigue sin ser efectiva incluso cuando el control de acceso a columnas está activado para la tabla. Se puede utilizar la sentencia ALTER MASK para alternar entre ENABLE y DISABLE.

CREATE MASK

Después de activar el control de acceso a columnas para una tabla, cuando se hace referencia a la tabla en una sentencia de manipulación de datos, la base de datos DB2 aplica de forma implícita todas las máscaras de columna habilitadas que se hayan creado para la tabla con el fin de enmascarar los valores devueltos para las columnas referenciadas en la tabla resultante final de las consultas o para determinar los valores nuevos utilizados en las sentencias de cambio de datos.

La creación de máscaras de columna antes de activar el control de acceso a columnas para una tabla es la secuencia recomendada para evitar invalidaciones múltiples de paquetes y de sentencias en memoria caché dinámicas que hacen referencia a la tabla.

- **Máscaras de columna que se han creado después de activar el control de acceso a columnas para una tabla:** las máscaras de columna habilitadas serán efectivas cuando se confirmen. Todos los paquetes y las sentencias en memoria caché dinámicas que hacen referencia a la tabla se invalidan. En adelante, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, la base de datos DB2 aplicará de forma implícita todas las máscaras de columna habilitadas a la sentencia. Las máscaras de columna inhabilitadas siguen sin ser efectivas incluso cuando el control de acceso a columnas está activado para la tabla.
- **No hay efecto en cascada cuando se hace referencia a tablas aplicadas de control de acceso a columnas o filas en las definiciones de máscara de columna:** Una definición de máscara de columna puede hacer referencia a las tablas y columnas que el control de acceso a filas o columnas aplica actualmente. El control de acceso de estas tablas y columnas se ignora cuando se hace referencia a la tabla para la cual se está creando la máscara de columna en una sentencia de manipulación de datos.
- **Consideración sobre los límites de DB2:** Si en la sentencia de manipulación de datos ya se consideran algunos de los límites de DB2 en la sentencia, cuantos más permisos de fila y máscaras de columna habilitados se hayan creado más probable será que éstos afecten a algunos límites. Esto ocurre porque las definiciones de máscara de columna y los permisos de fila habilitados están fusionados de forma implícita en la sentencia cuando se hace referencia a la tabla en la sentencia de manipulación de datos.
- **Máscaras de columna que están habilitadas pero en estado no válido:** Si una máscara de columna está habilitada para el control de acceso a columnas pero tiene el estado como no válido, el acceso a la tabla en la que se ha definido la máscara de columna permanece bloqueado hasta que se resuelve esta situación (SQLSTATE 560D0).
- **Máscaras de columna que devuelven datos no asignables a la columna en la que se ha definido la máscara:** se puede definir una máscara de columna para que devuelva datos que no sean asignables al tipo de datos de la columna en la que se ha definido la máscara. Cuando esto ocurra, la sentencia CREATE MASK es correcta, pero se notificará un error de conversión cuando se aplique la máscara en una consulta de usuario.

Ejemplos

- *Ejemplo 1:* después de activar el control de acceso a columnas para la tabla EMPLOYEE, Paul, del departamento de contabilidad, podrá ver el número de la seguridad social del empleado cuyo número de empleado es 123456. Mary, gerente, podrá ver únicamente los cuatro últimos caracteres del número de la seguridad social. Peter, que no tiene ningún rol, no podrá ver el número de la seguridad social.

```

CREATE MASK SSN_MASK ON EMPLOYEE
FOR COLUMN SSN RETURN
CASE WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'PAYROLL') = 1)
THEN SSN
WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER, 'MGR') = 1)
THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
ELSE NULL
END
ENABLE;
ALTER TABLE EMPLOYEE ACTIVATE COLUMN ACCESS CONTROL;
SELECT SSN FROM EMPLOYEE WHERE EMPNO = 123456;

```

- *Ejemplo 2:* en la sentencia SELECT, el SSN de columna está incluido en una expresión que es igual a la que se utiliza en la máscara de columna SSN_MASK. Después de activar el control de acceso a columnas para la tabla EMPLOYEE, la máscara de columna SSN_MASK se aplicará al SSN de columna en la sentencia SELECT. Para esta expresión en concreto, la sentencia SELECT produce el mismo resultado que antes de activar el control de acceso a columnas para todos los usuarios. El usuario puede sustituir la expresión de la sentencia SELECT con un SSN de columna para evitar que la misma expresión se evalúe dos veces.

```

CREATE MASK SSN_MASK ON EMPLOYEE
FOR COLUMN SSN RETURN
CASE WHEN (1 = 1) THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
ELSE NULL
END
ENABLE;
ALTER TABLE EMPLOYEE ACTIVATE COLUMN ACCESS CONTROL;
SELECT 'XXX-XX-' || SUBSTR(SSN,8,4) FROM EMPLOYEE WHERE EMPNO = 123456;

```

- *Ejemplo 3:* el gobierno del estado de California realizó un estudio para conocer el uso de las bibliotecas domésticas en cada ciudad. Participaron en el estudio cincuenta hogares de cada ciudad. Cada participante tenía la opción de permitir o no permitir mostrar su uso en los informes generados a partir del resultado del estudio.

Se utilizó una sentencia SELECT para generar un informe que mostrara el promedio de horas que dedicaba cada participante en cada ciudad. Se creó una máscara de columna CITY_MASK para enmascarar el nombre de la ciudad en función de la información de permiso o no permiso elegida por los participantes. No obstante, después de activar el control de acceso a columnas para la tabla LIBRARY_USAGE, la sentencia SELECT recibió un error de tiempo de vinculación. Esto ocurre porque la máscara de columna CITY_MASK hace referencia a otra columna LIBRARY_OPT y LIBRARY_OPT no identifica una columna de agrupación.

```

CREATE MASK CITY_MASK ON LIBRARY_USAGE
FOR COLUMN CITY RETURN
CASE WHEN (LIBRARY_OPT = 'OPT-IN') THEN CITY
ELSE ' '
END
ENABLE;
ALTER TABLE LIBRARY_USAGE ACTIVATE COLUMN ACCESS CONTROL;
SELECT CITY, AVG(LIBRARY_TIME) FROM LIBRARY_USAGE GROUP BY CITY;

```

- *Ejemplo 4:* el empleado con el número de empleado 123456 gana un bono de 8000 dólares y un salario de 80.000 dólares en el mes de mayo. Cuando el gerente recupera su salario, recibe su salario y no un valor nulo. Esto ocurre porque se no ha definido el efecto cascada cuando la máscara de columna SALARY_MASK hace referencia a la columna BONUS para la cual se ha definido la máscara de columna BONUS_MASK.

```

CREATE MASK SALARY_MASK ON EMPLOYEE
FOR COLUMN SALARY RETURN
CASE WHEN (BONUS < 10000) THEN SALARY
ELSE NULL

```

CREATE MASK

```
END
ENABLE;
CREATE MASK BONUS_MASK ON EMPLOYEE
FOR COLUMN BONUS RETURN
CASE WHEN (BONUS > 5000) THEN NULL
ELSE BONUS
END
ENABLE;
ALTER TABLE EMPLOYEE ACTIVATE COLUMN ACCESS CONTROL;
SELECT SALARY FROM EMPLOYEE WHERE EMPNO = 123456;
```

CREATE METHOD

La sentencia CREATE METHOD se utiliza para asociar un cuerpo de método a una especificación de método que ya forma parte de la definición de un tipo estructurado definido por el usuario.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CREATEIN para el esquema del tipo estructurado al que se hace referencia en la sentencia CREATE METHOD
- Propietario del tipo estructurado al que se hace referencia en la sentencia CREATE METHOD
- Autorización DBADM

Para asociar un cuerpo de método externo a su especificación de método, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos
- Autorización DBADM

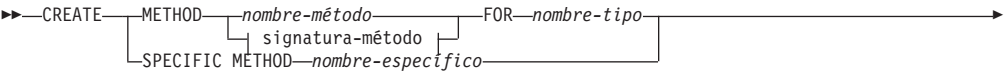
Cuando se crea un método de SQL, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla, vista o apodo identificados en cualquier selección completa:

- Privilegio CONTROL para esa tabla, vista o apodo
- Privilegio SELECT para esa tabla, vista o apodo
- Autorización DATAACCESS

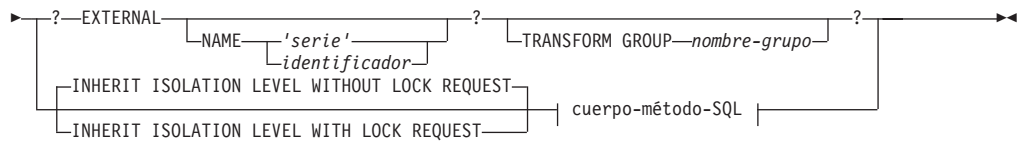
Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE METHOD.

Los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo se aplican al invocarse el método. El ID de autorización de la conexión se puede correlacionar con un ID de autorización remoto diferente.

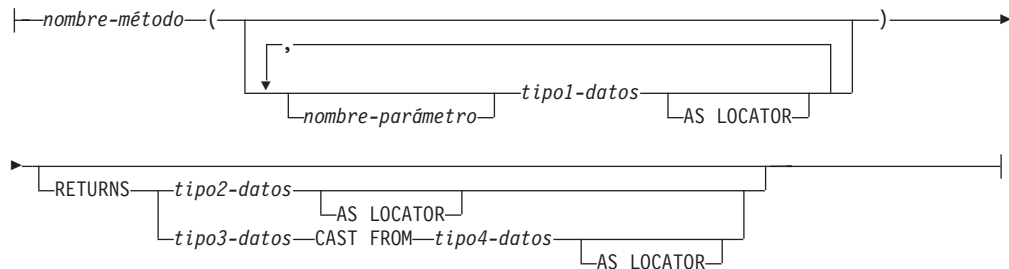
Sintaxis



CREATE METHOD



signatura-método:



cuerpo-método-SQL:



Notas:

- 1 La sentencia de SQL compuesto (en línea) sólo se soporta para un cuerpo-método-SQL de una definición de método de SQL de una base de datos no particionada.

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.

tipo1-datos

Especifica el tipo de datos de cada parámetro. No se da soporte a los tipos de matriz (SQLSTATE 42815).

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

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 signature de método proporciona una cláusula RETURNS, esta 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.

tipo2-datos

Especifica el tipo de datos de la salida. No se da soporte a los tipos de matriz (SQLSTATE 42815).

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.

tipo3-datos **CAST FROM** *tipo4-datos*

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 dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto 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 especificado 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

CREATE METHOD

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 bytes. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada. Para obtener más información acerca de los convenios específicos de lenguaje, consulte la "sentencia CREATE FUNCTION (escalar externa)".

identificador

Este identificador especificado es un identificador de 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, se ignora la parte correspondiente al nombre del 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.

Se recomienda especialmente especificar un nombre de grupo de transformación; si no se especifica esta cláusula, el nombre de grupo por omisión que se utiliza es 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).

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST o INHERIT ISOLATION LEVEL WITH LOCK REQUEST

Especifica si una petición de bloqueo puede asociarse o no a una cláusula de aislamiento de la sentencia cuando el método hereda el nivel de aislamiento de la sentencia que invoca el método. El valor por omisión es INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST

Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, no se puede invocar en el contexto de una sentencia de SQL que incluya una cláusula de petición de bloqueo como parte de una cláusula de aislamiento especificada (SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST

Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, también hereda la cláusula de petición de bloqueo especificada.

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 más información, consulte los apartados "Sentencia de SQL compuesto (en línea)" y "Sentencia RETURN".

Normas

- Para poder utilizar CREATE METHOD, se debe definir previamente la especificación de método utilizando la sentencia CREATE TYPE o ALTER TYPE (SQLSTATE 42723).
- Si el método que está creándose es un método de alteración temporal, se invalidan los paquetes que dependen de los métodos siguientes:
 - El método original
 - Otros métodos de alteración temporal cuyo sujeto es un supertipo del método que está creándose
- El tipo de datos XML no se puede utilizar en un método.

Notas

- Si el método permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- **Privilegios:** el definidor de un método siempre recibe el privilegio EXECUTE sobre el método, así como el derecho de descartar el método.

Si se crea un método EXTERNAL, el usuario que define el método siempre recibe el privilegio EXECUTE y WITH GRANT OPTION.

Si se crea un método de SQL, el usuario que define el método sólo recibirá el privilegio EXECUTE y WITH GRANT OPTION para el método cuando dicho usuario disponga de WITH GRANT OPTION para todos los privilegios necesarios para definir el método o si dicho usuario dispone de autorización SYSADM o DBADM. El usuario que define un método de SQL sólo adquiere privilegios si, en el momento de crearse el método, existen los privilegios de los que estos privilegios se obtienen. El usuario que define el método debe disponer de estos privilegios directamente o bien porque PUBLIC tiene los privilegios. Los privilegios que tienen los grupos de los que es miembro el usuario que define el método no se consideran. Cuando se utiliza el método, el ID de autorización del usuario conectado debe disponer de privilegios válidos para la tabla o vista a la que hace referencia el apodo en la fuente de datos.

- **Restricciones de acceso a las tablas:** si un método se ha definido como READS SQL DATA, ninguna sentencia del método podrá acceder a una tabla que la sentencia que ha invocado el método está modificando (SQLSTATE 57053).

Ejemplos

- *Ejemplo 1:*

CREATE METHOD

```
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 MODULE

La sentencia CREATE MODULE crea un módulo en el servidor de aplicaciones.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito del módulo no existe.
- Privilegio CREATEIN para el esquema, si el nombre de esquema del módulo hace referencia a un esquema existente.
- Autorización DBADM

Para sustituir un módulo existente, el ID de autorización de la sentencia debe ser el propietario del módulo existente (SQLSTATE 42501).

Sintaxis

```

▶▶ CREATE OR REPLACE MODULE module-name ▶▶

```

Descripción

OR REPLACE

Especifica que se debe sustituir la definición del módulo si existe una en el servidor actual. La definición de módulo existente se descarta de forma efectiva, incluidos todos los objetos del módulo, antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre el módulo no se ven afectados por ello. Esta opción se ignora si no existe una definición para el módulo en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

module-name

Da nombre al módulo. El nombre (incluido el calificador de esquema implícito o explícito) no debe identificar un módulo existente en el servidor actual. El nombre de módulo y el nombre de esquema no deben comenzar por los caracteres 'SYS' (SQLSTATE 42939) y no se recomienda utilizar SESSION.

Notas

- Un módulo debe ser una recopilación de otros objetos de base de datos. Una vez creado un módulo, los objetos contenidos en él se gestionan mediante la sentencia ALTER MODULE. Un módulo puede incluir funciones, procedimientos, tipos, variables globales y condiciones. Los objetos de un módulo se pueden publicar para que estén disponibles de forma que se pueda hacer referencia a ellos desde el exterior del módulo. Si un objeto no está publicado, sólo se puede hacer referencia a él desde dentro del módulo. Se considera que un módulo consta de dos partes:

CREATE MODULE

- La especificación del módulo está formada por todos los objetos publicados, excluidos los cuerpos de rutinas.
- El cuerpo del módulo está formado por todos los objetos que no están publicados y por los cuerpos de cualquier rutina publicada.

Las acciones de gestión de módulos son las siguientes:

- ADD para agregar un objeto al módulo sin publicarlo o para sustituir un prototipo de rutina por la definición de rutina implementada.
- PUBLISH para añadir un objeto al módulo y publicarlo.
- COMMENT para comentar objetos del módulo.
- DROP para descartar un objeto del módulo o para descartar el cuerpo del módulo.

Debe existir un objeto publicado como mínimo en un módulo para que se pueda hacer referencia al módulo.

Ejemplo

Cree un módulo denominado *salesModule*

```
CREATE MODULE salesModule
```

CREATE NICKNAME

La sentencia CREATE NICKNAME define un apodo para un objeto de fuente de datos.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

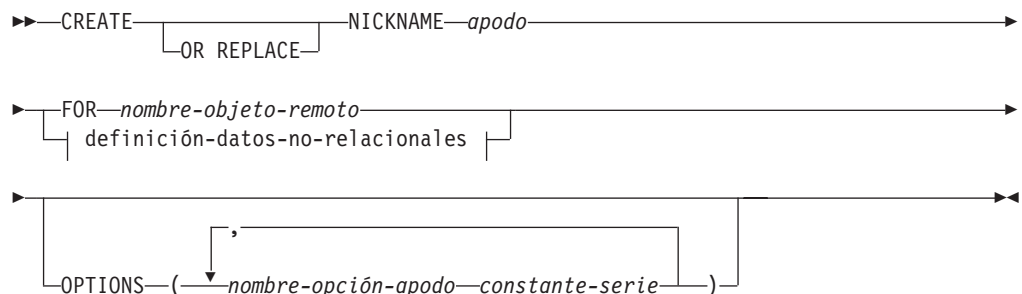
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATETAB para la base de datos federada, así como uno de los siguientes:
 - Autorización IMPLICIT_SCHEMA para la base de datos federada, si el nombre de esquema implícito o explícito del apodo no existe
 - Privilegio CREATEIN en el esquema, si el nombre de esquema del apodo hace referencia a un esquema existente
- Autorización DBADM

Para las fuentes de datos que requieren una correlación de usuarios, los privilegios contenidos en el ID de autorización en la fuente de datos deben incluir el privilegio para seleccionar datos del objeto que el apodo representa.

Para sustituir un apodo existente, el ID de autorización de la sentencia debe ser el propietario del apodo existente (SQLSTATE 42501).

Sintaxis



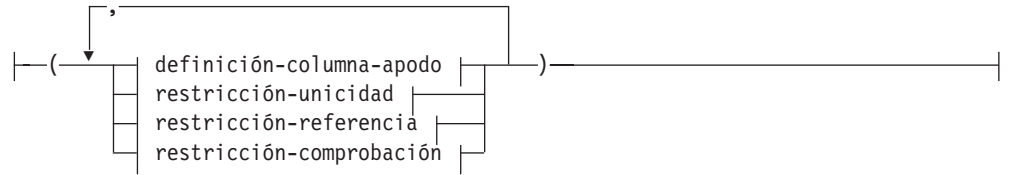
definición-datos-no-relacionales:

```

[ lista-columna-apodo ] [FOR SERVER nombre-servidor]
  
```

lista-columna-apodo:

CREATE NICKNAME



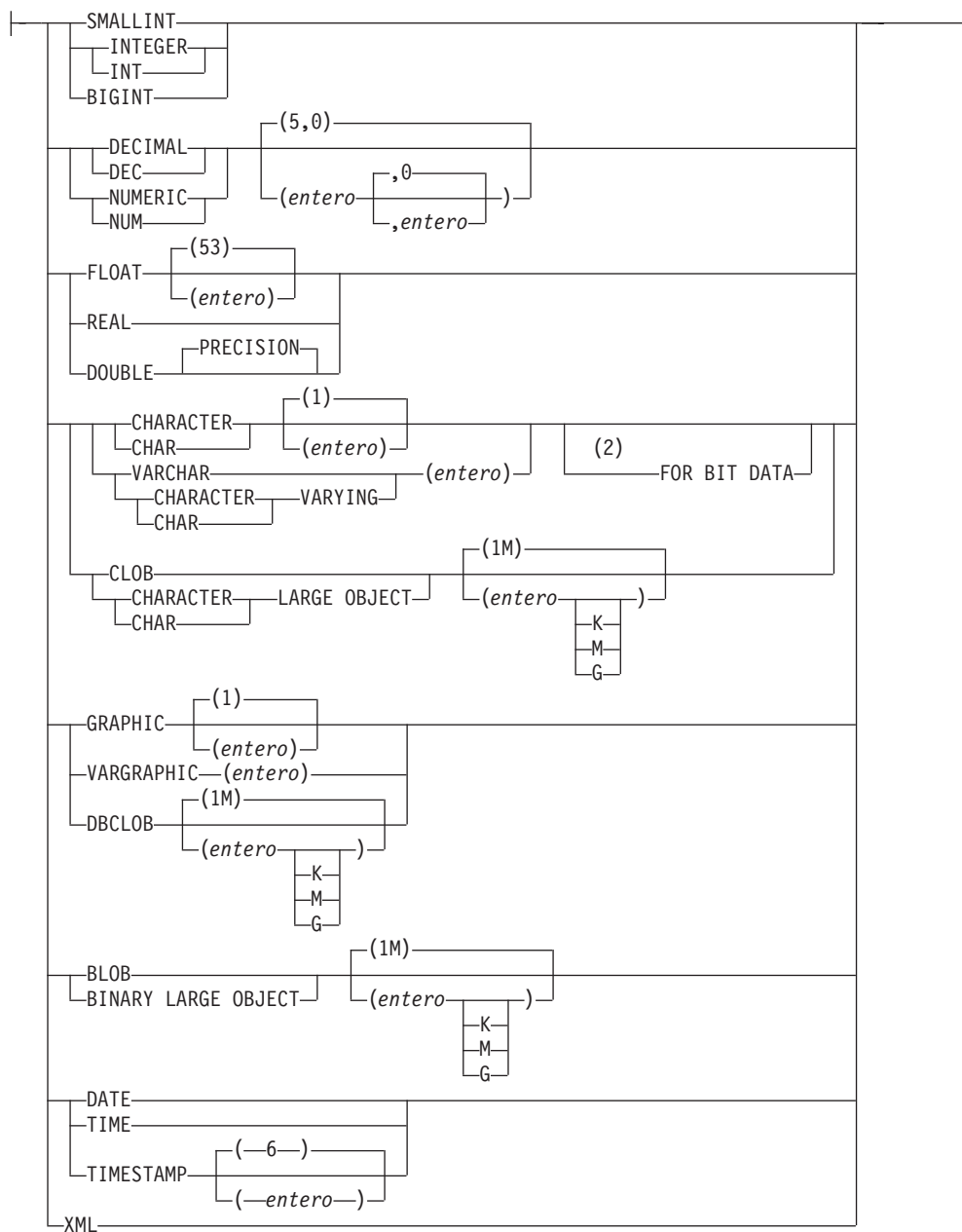
definición-columna-apodo:



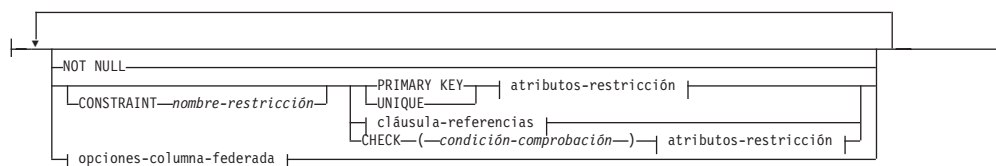
tipo-datos-locales:



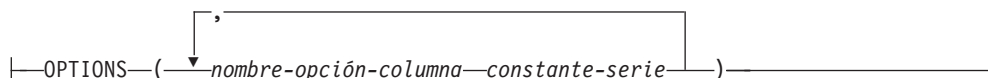
tipo-incorporado:



opciones-columna-apodo:

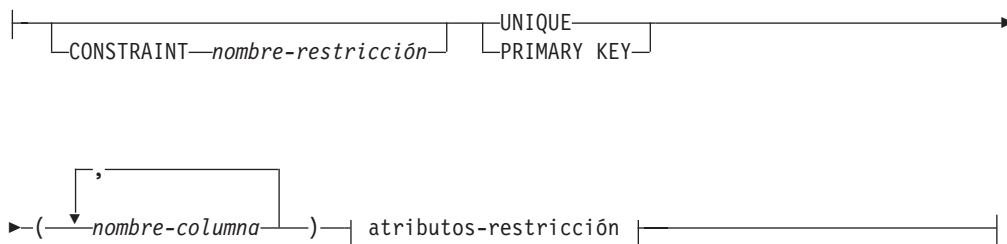


opciones-columna-federada:

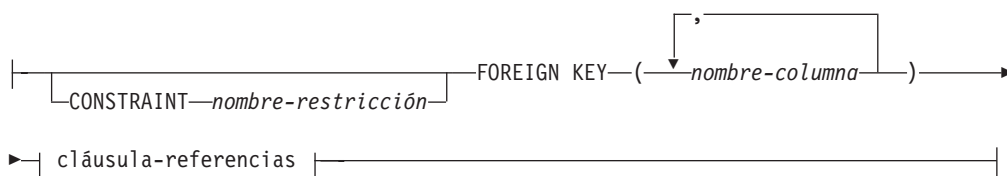


CREATE NICKNAME

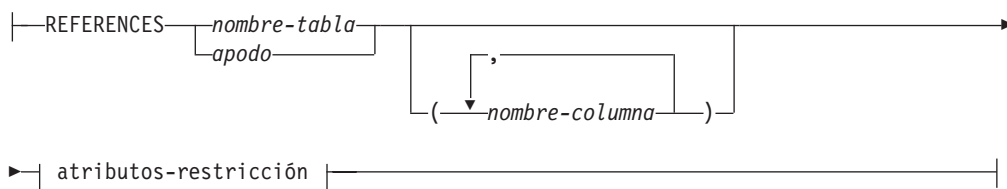
restricción-unicidad:



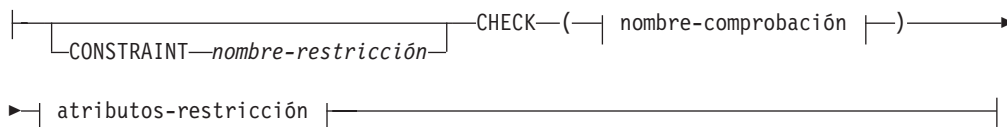
restricción-referencia:



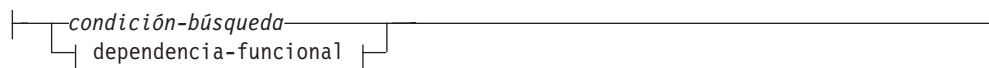
cláusula-referencias:



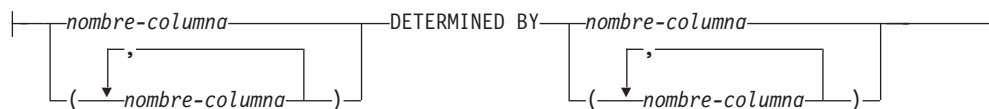
restricción-comprobación:



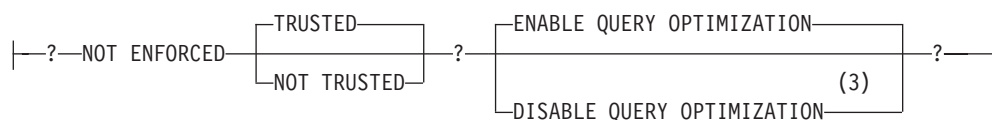
condición-error:



dependencia-funcional:



atributos-restricción:

**Notas:**

- 1 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2).
- 2 La cláusula FOR BIT DATA se puede especificar en cualquier orden con las restricciones de columna siguientes.
- 3 No se da soporte a DISABLE QUERY OPTIMIZATION para una restricción de clave primaria o exclusiva.

Descripción**OR REPLACE**

Especifica que se debe sustituir la definición del apodo si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre el apodo no se ven afectados por ello. Esta opción se ignora si no existe una definición para el apodo en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

apodo

Especifica un apodo, el identificador utilizado por el servidor federado para el objeto de fuente de datos. El apodo, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo ni alias descrito en el catálogo. El nombre de esquema no debe empezar por 'SYS' (SQLSTATE 42939).

FOR nombre-objeto-remoto

Especifica un identificador. Para fuentes de datos que dan soporte a nombres de esquema, se trata de un identificador compuesto de tres partes con el formato *nombre-fuente-datos.nombre-esquema-remoto.nombre-tabla-remota*. Para las fuentes de datos que no dan soporte a nombres de esquema, se trata de un identificador compuesto de dos partes con el formato *nombre-fuente-datos.nombre-tabla-remota*.

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 al *nombre-servidor* en la sentencia CREATE SERVER.

nombre-esquema-remoto

Nombra el esquema al que pertenece la tabla o la vista. Si el nombre de esquema remoto contiene caracteres especiales o caracteres en minúsculas, debe especificarse entre comillas dobles.

nombre-tabla-remota

Especifica el nombre del objeto de fuente de datos específico (como, por ejemplo, una tabla o una vista) para el que se crea el apodo. La tabla no puede ser una tabla temporal declarada (SQLSTATE 42995). Si el nombre de tabla remota contiene caracteres especiales o caracteres en minúsculas, debe especificarse entre comillas dobles. En el caso de DB2 para Linux, UNIX y Windows, también puede especificar el alias de una tabla, una vista o un apodo. Para DB2 para z/OS o DB2 para i, puede especificar el alias de una tabla o una vista.

CREATE NICKNAME

definición-datos-no-relacionales

Define los datos a los que se debe acceder a través de un derivador no relacional.

definición-columna-apodo

Define los atributos locales de la columna para el apodo. Algunos derivadores necesitan que se especifiquen estos atributos, mientras que otros derivadores admiten que los atributos se determinen a partir de la fuente de datos.

nombre-columna

Especifica el nombre local de la columna. El nombre puede ser diferente del de la columna correspondiente del *nombre-objeto-remoto*.

tipo-datos-locales

Especifica el tipo de datos locales para la columna. Algunos derivadores sólo dan soporte a un subconjunto de tipos de datos SQL. Para ver descripciones de tipos de datos específicos, consulte "CREATE TABLE" .

opciones-columna-apodo

Especifica opciones adicionales relacionadas con columnas del apodo.

NOT NULL

Especifica que la columna no admite valores nulos.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar una restricción que ya se haya especificado en la misma sentencia CREATE NICKNAME (SQLSTATE 42710).

Si se omite esta cláusula, el sistema genera un identificador de 18 bytes que es exclusivo entre los identificadores de las restricciones existentes que se han definido en el apodo. (El identificador se compone de la palabra 'SQL' seguida de una secuencia de 15 caracteres numéricos generados por una función basada en la indicación de fecha y hora.)

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, se puede utilizar el *nombre-restricción* como nombre de una especificación de índice que se crea 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.

Consulte PRIMARY KEY en la descripción de *restricción-unicidad*.

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.

Consulte UNIQUE en la descripción de *restricción-unicidad*.

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 *cláusula-referencias* en la descripción de *restricción-referencial*.

CHECK (*condición-comprobación*)

Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Consulte la descripción de CHECK (*condición-comprobación*).

OPTIONS

Indica las opciones de columna que se añaden cuando se crea el apodo. Algunos derivadores requieren que se especifiquen algunas opciones de columna.

nombre-opción-columna

Especifica el nombre de la opción.

constante-serie

Especifica el valor para *nombre-opción-columna* como una constante de serie de caracteres.

restricción-unicidad

Define una restricción de clave primaria o de unicidad.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de clave primaria o de unicidad.

UNIQUE (*nombre-columna, ...*)

Define una clave exclusiva compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XQuery". No se pueden utilizar columnas LOB, columnas de tipo diferenciado basadas en LOB o columnas de tipo estructurado como parte de una clave exclusiva, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

La descripción del apodo, tal como se ha registrado en el catálogo, incluye la clave exclusiva y su especificación de índice. Se creará automáticamente una especificación de índice para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre de la especificación de índice será el mismo que el *nombre-restricción* si no entra en conflicto

CREATE NICKNAME

con un índice o una especificación de índice existente en el esquema donde se crea el apodo. Si el nombre de la especificación de índice entra en conflicto, el nombre será la palabra 'SQL' seguida de los caracteres de la indicación de fecha y hora (*aammddhhmmssxxx*), con SYSIBM como 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 definirse como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XQuery". No se pueden utilizar columnas LOB, columnas de tipo diferenciado basadas en LOB o columnas de tipo estructurado como parte de una clave primaria, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

Sólo se puede definir una única clave primaria en un apodo.

La descripción del apodo, tal como está registrada en el catálogo, incluye la clave primaria y su especificación de índice. Se creará automáticamente una especificación de índice para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre de la especificación de índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice o una especificación de índice existente en el esquema donde se crea el apodo. Si el nombre de la especificación de índice entra en conflicto, el nombre será la palabra 'SQL', seguida por los caracteres de la indicación de fecha y hora (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

restricción-referencia

Define una restricción de referencia.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de referencia.

FOREIGN KEY (*nombre-columna*,...)

Define una restricción de referencia con el *nombre-restricción* especificado.

Deje que N1 indique el apodo de objeto 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 N1 y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XQuery". Las claves foráneas

pueden definirse en columnas de longitud variable cuya longitud sea superior a 255 bytes. No se pueden utilizar columnas LOB, columnas de tipo diferenciado basadas en LOB o columnas de tipo estructurado como parte de una clave exclusiva (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 contienen tipos de datos compatibles (ambas columnas son numéricas, de serie de caracteres, gráficas, de fecha y hora o tienen el mismo tipo diferenciado).

cláusula-referencias

Especifica la tabla padre o el apodo padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla* **o** *apodo*

La tabla o el apodo especificados en una cláusula REFERENCES debe identificar una tabla base o un apodo que se describa en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción de referencia está duplicada si su clave foránea, clave padre y tabla padre o apodo padre son iguales a la clave foránea, clave padre y tabla padre o apodo padre de una restricción de referencia especificada previamente. Las restricciones de referencia duplicadas se pasan por alto y se emite un aviso (SQLSTATE 01543).

En la siguiente explicación, N2 indicará la tabla padre identificada o el apodo padre y N1 indicará el apodo que se está creando (o modificando). N1 y N2 pueden ser el mismo apodo.

La clave foránea especificada debe tener el mismo número de columnas que la clave padre de N2, y la descripción de la columna número *n* de la clave foránea debe ser comparable a la descripción de la columna número *n* de esa clave padre. Las columnas de indicación de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta norma.

La restricción de referencia especificada por una cláusula FOREIGN KEY define una relación en la que N2 es el padre y N1 es dependiente.

(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 N2. La misma columna no se puede identificar más de una vez.

La lista de nombres de columnas debe coincidir con el conjunto de columnas (en cualquier orden) de la clave primaria o una restricción exclusiva que exista en N2 (SQLSTATE 42890). Si no se especifica una lista de nombres de columna, N2 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.

atributos-restricción

Define los atributos que se asocian a las restricciones de comprobación o de integridad referencial.

NOT ENFORCED

El gestor de bases de datos no aplica la restricción durante la

realización de las operaciones normales como, por ejemplo, la inserción, la actualización o la supresión.

TRUSTED

Se puede confiar en los datos para que se ajusten a la restricción. TRUSTED sólo debe utilizarse si se reconoce de forma independiente que los datos de la tabla se ajustan a la restricción. Los resultados de la consulta pueden ser imprevisibles si los datos no se ajustan realmente a la restricción. Es la opción por omisión.

NOT TRUSTED

No se puede confiar en los datos para que se ajusten a la restricción. NOT TRUSTED está pensado para casos en los que los datos se ajustan a la restricción para la mayor parte de las filas, pero no se reconoce de forma independiente que todas las filas o las futuras adiciones se ajustarán a la restricción. Si una restricción es NOT TRUSTED y se ha habilitado para la optimización de consultas, no se utilizará para realizar optimizaciones que dependan de los datos que se ajusten por completo a la restricción. Sólo se puede especificar NOT TRUSTED para las restricciones de integridad referenciales (SQLSTATE 42613).

ENABLE QUERY OPTIMIZATION

Se supone que la restricción es verdadera y se puede utilizar para la optimización de la consulta bajo las circunstancias adecuadas.

DISABLE QUERY OPTIMIZATION

No se puede utilizar la restricción para la optimización de consulta.

restricción-comprobación

Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o que define una dependencia funcional entre columnas.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de comprobación.

CHECK (*condición-comprobación*)

Define una restricción de comprobación. La *condición-comprobación* debe ser verdadera o desconocida para cada fila del apodo.

condición-búsqueda

La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia a columna debe ser a una columna del apodo que se crea.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- No puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
 - Subconsultas
 - Operaciones de eliminación de referencia o funciones Deref donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
 - Especificaciones CAST con una cláusula SCOPE
 - Funciones de columna
 - Funciones que no sean deterministas

- Funciones definidas para que exista una acción externa
- Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
- Variables del lenguaje principal
- Marcadores de parámetro
- Registros especiales y funciones incorporadas que dependen del valor de un registro especial
- Variables globales
- Referencias a columnas generadas que no correspondan a la columna de identidad

dependencia-funcional

Define una dependencia funcional entre columnas.

El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la columna no debe ser un tipo de datos LOB, un tipo diferenciado basado en el tipo de datos LOB ni un tipo estructurado (SQLSTATE 42962). Ninguna columna del conjunto de columnas hijo puede ser una columna con posibilidad de nulos (SQLSTATE 42621).

Si se especifica una restricción de comprobación como parte de una *definición-columna*, sólo puede establecerse una referencia de columna que haga referencia a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de apodo pueden contener referencias a columna que identifiquen columnas definidas previamente en la sentencia CREATE NICKNAME. No se comprueba si hay incoherencias, condiciones duplicadas ni condiciones equivalentes en las restricciones de comprobación. Por lo tanto, se pueden definir restricciones de comprobación contradictorias o redundantes, lo que podría dar lugar a posibles errores en tiempo de ejecución.

FOR SERVER *nombre-servidor*

Especifica el servidor que se ha registrado utilizando la sentencia CREATE SERVER. Este servidor se utilizará para acceder a los datos para el apodo.

OPTIONS

Indica las opciones de apodo que se habilitan cuando se crea el apodo.

nombre-opción-apodo

Especifica el nombre de la opción.

constante-serie

Especifica el valor para *nombre-opción-apodo* como una constante de serie de caracteres.

Notas

- Las tablas y las vistas son ejemplos de objetos de fuente de datos relacionales. Los ejemplos de objetos de fuente de datos relacionales son: los objetos de Documentum o tablas registradas, los archivos de texto (.txt) y los archivos Microsoft Excel (.xls).
- El objeto de fuente de datos al que el apodo hace referencia ya debe existir en la fuente de datos indicada por el primer calificador en el *nombre-objeto-remoto*.
- La lista de los tipos de datos de fuente de datos soportados varía según el derivador. Los derivadores no dan soporte a los tipos de datos de fuente de datos XML y REF. El tipo de datos de fuente de datos DECFLOAT sólo es compatible con el derivador DB2 para IBM DB2 para Linux, UNIX y Windows Versión 9.5 o posterior. Cuando la sentencia CREATE NICKNAME especifica un *nombre-objeto-remoto* que tiene columnas con tipos de datos no soportados, se devuelve un error.

Los tipos de datos de fuente de datos LONG VARCHAR y LONG VARCHARIC se correlacionan con los tipos de datos CLOB y DBCLOB, respectivamente. LONG VARCHAR FOR BIT DATA se correlaciona con BLOB.

- La longitud máxima permitida de los nombres de índice de DB2 es de 128 bytes. Si se está creando un apodo para una tabla relacional que tiene un índice cuyo nombre sobrepasa esta longitud, no se cataloga todo el nombre. En su lugar, DB2 lo trunca a 128 bytes. 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 2 a 9 y, si es necesario, con los números 0 a 9 para el 127 carácter del nombre, para el 126 carácter y así sucesivamente hasta que se genera un número exclusivo. A modo ilustrativo: El nombre de 130 bytes de un índice de una tabla de fuente de datos es AREALLY...REALLYLONGNAME. Los nombres AREALLY...REALLYLONGNA y AREALLY...REALLYLONGN0 ya existen en el esquema al que pertenece el índice. El nombre nuevo tiene más de 128 bytes; por lo tanto, DB2 lo trunca a AREALLY...REALLYLONGNA. Puesto que este nombre ya existe en el esquema, DB2 cambia la versión truncada por AREALLY...REALLYLONGN0. Y puesto que este nombre ya existe, DB2 cambia la versión truncada por AREALLY...REALLYLONGN1. Este nombre aún no existe en el esquema, por lo que DB2 lo acepta como el nuevo nombre.
- Cuando se crea un apodo para un objeto de fuente de datos, DB2 almacena los nombres de las columnas de apodo en el catálogo. Cuando el objeto de fuente de datos es una tabla o una vista, DB2 hace que los nombres de las columnas de apodo sean los mismos que los nombres de las columnas de tabla o de vista. Si un nombre excede la longitud máxima permitida para los nombres de columna de DB2, DB2 trunca el nombre en función de esa longitud. Si la versión truncada no es exclusiva entre los demás nombres de las columnas de la tabla o vista, DB2 hace que sea exclusivo siguiendo el procedimiento que se describe en el párrafo anterior.
- Si el objeto de fuente de datos tiene índices definidos, se crean especificaciones de índice para cada índice cuando se crea el apodo. No se crean especificaciones de índice en la fuente de datos para los índices que tienen:
 - Nombres de columnas duplicados
 - Más de 64 columnas
 - Más de 1.024 bytes en la suma de la longitud de las partes de la clave de índice
- Si se cambia la definición de un objeto de fuente de datos remota (por ejemplo, se suprime una columna o se cambia el tipo de datos), se debe descartar y

volver a crear el apodo; de lo contrario, se pueden producir errores cuando se utilice el apodo en una sentencia de SQL.

- **Almacenamiento en antememoria y objetos protegidos:** Cuando se crea un apodo, si el objeto de fuente de datos no está protegido, ALLOW CACHING está en vigor para el apodo. Si el servidor federado puede detectar que el objeto de fuente de datos está protegido, DISALLOW CACHING está en vigor para el apodo. La opción DISALLOW CACHING garantiza que cada vez que se utiliza el apodo se devuelven los datos del ID de autorización apropiados desde la fuente de datos en el momento de la ejecución de la consulta. Esto se realiza limitando el uso del apodo en la definición de una tabla de consulta materializada en el servidor federado, que podría utilizarse para guardar en la antememoria los datos de apodo. La sentencia ALTER NICKNAME puede utilizarse para cambiar entre ALLOW CACHING y DISALLOW CACHING.
- **Alternativas de sintaxis:** se soporta la sintaxis siguiente para mantener la compatibilidad con versiones anteriores de DB2:
 - ADD puede especificarse antes de *nombre-opción-apodo constante-serie*.
 - ADD se puede especificar antes de *nombre-opción-columna constante-serie*.

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 para z/OS denominada OS390A.

```
CREATE NICKNAME DEPT
FOR OS390A.HEDGES.DEPARTMENT
```

- *Ejemplo 2:* seleccione todos los registros de la vista para la cual se ha creado un apodo en el Ejemplo 1. Se debe hacer referencia a la vista mediante su apodo. Se puede hacer referencia a la vista remota utilizando el nombre por el cual se conoce en la fuente de datos únicamente en sesiones de paso a través.

La siguiente sentencia es válida tras crearse el apodo DEPT:

```
SELECT * FROM DEPT
```

La sentencia siguiente no es válida:

```
SELECT * FROM OS390A.HEDGES.DEPARTMENT
```

- *Ejemplo 3:* Cree un apodo para la tabla remota JAPAN que está en un esquema denominado salesdata. Puesto que el nombre de esquema y el nombre de tabla de la fuente de datos se almacenan en minúsculas, especifique el nombre de esquema remoto y el nombre de tabla entre comillas dobles:

```
CREATE NICKNAME JPSALES
FOR asia."salesdata"."japón"
```

- *Ejemplo 4:* Cree un apodo para el archivo estructurado de tabla DRUGDATA1.TXT. Incluya las opciones de apodo FILE_PATH, COLUMN DELIMITER, KEY_COLUMN y VALIDATE_DATA_FILE en la sentencia.

```
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',
SORTED 'Y',
VALIDATE_DATA_FILE 'Y')
```

CREATE NICKNAME

- *Ejemplo 5:* Cree el apodo padre CUSTOMERS en múltiples archivos XML bajo la vía de acceso de directorio especificada /home/db2user. Incluya las opciones siguientes:
 - Opciones de columna:
 - La opción de columna XPATH para la columna VARCHAR(5) denominada ID, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
 - La opción de columna XPATH para la columna VARCHAR(16) denominada NAME, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
 - La opción de columna XPATH para la columna VARCHAR(30) denominada ADDRESS, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
 - La opción de columna PRIMARY_KEY para la columna VARCHAR(16) denominada CID, que identifica el apodo de los clientes como apodo padre en una jerarquía de apodos
 - Opciones de apodo:
 - La opción de apodo DIRECTORY_PATH para indicar la ubicación de los archivos XML que proporcionan los datos
 - La opción de apodo XPATH para indicar el elemento de los archivos XML donde empiezan los datos
 - La opción de apodo STREAMING para indicar que los datos fuente XML se separan y procesan elemento por elemento. En este ejemplo, el elemento es un registro de cliente.

```
CREATE NICKNAME clientes
(id      VARCHAR(5)  OPTIONS(XPATH './@id'),
 name   VARCHAR(16) OPTIONS(XPATH './name'),
 address VARCHAR(30) OPTIONS(XPATH './address/@street'),
 cid    VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
OPTIONS
(DIRECTORY_PATH '/home/db2user',
 XPATH '//customer',
 STREAMING 'YES')
```

CREATE PERMISSION

La sentencia CREATE PERMISSION crea un permiso de fila en el servidor actual.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES se especifica implícita o explícitamente.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM. La autorización SECADM puede crear un permiso de fila en cualquier esquema. No son necesarios privilegios adicionales para hacer referencia a otros objetos en la definición de permiso. Por ejemplo, no se necesita el privilegio SELECT para recuperar de una tabla, y no se necesita el privilegio EXECUTE para llamar a una función definida por el usuario.

Sintaxis

```

▶ CREATE [OR REPLACE] PERMISSION nombre-permiso ON nombre-tabla
[AS nombre-correlación]
▶ FOR ROWS WHERE condición-búsqueda ENFORCED FOR ALL ACCESS [DISABLE | ENABLE]

```

Descripción

OR REPLACE

Especifica que se debe sustituir la definición del permiso de fila si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que se sustituya la nueva definición en el catálogo.

nombre-permiso

Asigna un nombre al permiso de fila. El nombre, incluyendo el calificador implícito o explícito, no debe identificar un permiso de fila o una máscara de columna que ya exista en el servidor actual (SQLSTATE 42710).

nombre-tabla

Identifica la tabla en la que se crea el permiso de fila. El nombre debe identificar una tabla existente en el servidor actual (SQLSTATE 42704). No debe identificar apodos, tablas temporales creadas o declaradas, vistas, alias, sinónimos o tablas con tipo (SQLSTATE 42809). Tampoco debe identificar tablas de catálogo (SQLSTATE 42832).

nombre-correlación

Especifica un nombre de correlación que se puede utilizar dentro de la *expresión-case* para designar la tabla.

CREATE PERMISSION

FOR ROWS WHERE

Indica que se ha creado un permiso de fila. Un permiso de fila especifica una condición de búsqueda bajo las filas de la cual se puede acceder.

condición-búsqueda

Especifica una condición que puede ser verdadera o falsa para la fila de una tabla. Sigue las mismas reglas que utiliza la condición de búsqueda en una cláusula WHERE de una consulta de subselección. Además, la condición de búsqueda no debe hacer referencia a ninguno de los objetos o elementos siguientes (SQLSTATE 428HB):

- Una tabla temporal global creada o una tabla temporal global declarada.
- Un apodo.
- Una función de tabla.
- Un método.
- Un marcador de parámetro (SQLSTATE 42601).
- Una función definida por el usuario que no se define como segura.
- Una función o expresión (como una expresión de cambio de fila o una expresión de secuencia) que no es determinante o que tiene una acción externa.
- Una función escalar XMLQUERY.
- Un predicado XMLEXISTS.
- Una especificación OLAP.
- Un * o name.* en una cláusula SELECT.
- Una pseudocolumna.
- Una función agregada sin especificar la cláusula SELECT.
- Una vista que incluya alguna de las restricciones incluidas anteriormente en su definición.

Si la *condición-búsqueda* hace referencia a tablas con el control de acceso a filas o columnas activado actualmente, el control de acceso a dichas tablas no será en cascada. Consulte "Notes" para obtener más detalles.

ENFORCED FOR ALL ACCESS

Especifica que el permiso de fila se aplica a todas las referencias de la tabla. Si el control de acceso a filas está activado para la tabla, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, la base de datos DB2 aplicará de forma implícita el permiso de fila para controlar el acceso a la tabla. Si la referencia de la tabla es para una operación de captación como SELECT, la aplicación del permiso de fila determina el conjunto de filas que el usuario que ha solicitado la operación de captación puede recuperar. Si la referencia de la tabla es para una operación de cambio de datos como INSERT, la aplicación del permiso de fila determina si el usuario que ha solicitado la operación de cambio de datos puede insertar o actualizar todas las filas que se deben cambiar.

ENABLE o DISABLE

Especifica que se debe habilitar o inhabilitar el permiso de fila. El valor por omisión es DISABLE.

DISABLE

Especifica que se debe inhabilitar el permiso de fila. Si el control de acceso a filas de la tabla no está activado actualmente, el permiso de filas no será efectivo cuando se active el control de acceso a filas de la tabla.

ENABLE

Especifica que se debe habilitar el permiso de fila para el control de acceso a filas. Si el control de acceso a filas no está activado actualmente para la tabla, el permiso de fila no será efectivo cuando se active el control de acceso a filas de la tabla. Si el control de acceso a filas está activado actualmente para la tabla, el permiso de fila será efectivo inmediatamente y todos los paquetes y las sentencias colocadas dinámicamente en antememoria que hacen referencia a la tabla se invalidarán.

Consulte la cláusula `ACTIVATE ROW ACCESS CONTROL` de la sentencia `ALTER TABLE` para obtener más información acerca de cómo activar el control de acceso a filas y cómo se aplican los permisos de fila.

Notas

- **Permisos de fila que se han creado antes de activar el control de acceso a filas para una tabla:** la sentencia `CREATE PERMISSION` es una sentencia independiente que se puede utilizar para crear un permiso de fila antes de activar el control de acceso a filas para una tabla. El único requisito será que la tabla y las columnas existan antes de crear el permiso. Se pueden crear varios permisos de fila para una tabla.

La definición del permiso de fila se almacena en el catálogo de DB2. Se registrará la dependencia de la tabla para la cual se está creando el permiso y las dependencias de otros objetos a los que se hace referencia en la definición. No se invalidará ningún paquete o sentencia en memoria caché dinámicas. Se puede crear un permiso de fila como habilitado o inhabilitado para el control de acceso a filas. Un permiso de fila habilitado no entra en vigor hasta que se utiliza la sentencia `ALTER TABLE` para activar el control de acceso a filas para la tabla. Un permiso de fila inhabilitado sigue sin ser efectivo incluso cuando el control de acceso a filas está activado para la tabla. Se puede utilizar la sentencia `ALTER PERMISSION` para alternar entre `ENABLE` y `DISABLE`.

Después de activar el control de acceso a filas para una tabla, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, la base de datos DB2 aplicará de forma implícita todos los permisos de fila habilitados que se hayan definido para la tabla a fin de controlar el acceso a esta.

La creación de permisos de fila antes de activar el control de acceso a filas para una tabla es la secuencia recomendada para evitar invalidaciones múltiples de paquetes y de sentencias en memoria caché dinámicas que hacen referencia a la tabla.

- **Permisos de fila que se han creado después de activar el control de acceso a filas para una tabla:** un permiso de fila habilitado entra en vigor cuando se confirma. Todos los paquetes y las sentencias en memoria caché dinámicas que hacen referencia a la tabla se invalidan. En adelante, cuando se haga referencia a la tabla en una sentencia de manipulación de datos, se aplicarán de forma implícita a la sentencia todos los permisos de fila habilitados. Los permisos de fila inhabilitados siguen sin ser efectivos incluso cuando el control de acceso a filas está activado para la tabla.
- **No hay efecto cascada cuando se hace referencia a tablas aplicadas de control de acceso a filas o columnas en las definiciones de permiso de fila:** una definición de permiso de fila podría hacer referencia a tablas y columnas actualmente aplicadas por el control de acceso a filas o columnas. El control de acceso de estas tablas se ignora cuando se hace referencia a la tabla para la cual se está creando el permiso de fila en una sentencia de manipulación de datos.
- **Consideración sobre los límites de DB2:** Si en la sentencia de manipulación de datos ya se consideran algunos de los límites de DB2 en la sentencia, cuantos

CREATE PERMISSION

más permisos de fila y máscaras de columna se hayan habilitado más probable será que éstos afecten a algunos límites. Esto ocurre porque las definiciones de máscara de columna y los permisos de fila habilitados están fusionados de forma implícita en la sentencia cuando se hace referencia a la tabla en la sentencia de manipulación de datos. Consulte "Límites de SQL y XML" para obtener información sobre los límites de una sentencia.

- **Permisos que están habilitados pero en estado no válido:** Si un permiso está habilitado para el control de acceso a filas pero tiene el estado como no válido, el acceso a la tabla en la que se ha definido el permiso permanece bloqueado hasta que se resuelve esta situación (SQLSTATE 560D0).

Ejemplo

Los responsables de caja en una entidad financiera sólo pueden acceder a los clientes de su oficina. Todos los responsables de caja son miembros del rol TELLER. Los representantes del servicio de atención al cliente pueden acceder a todos los clientes de la entidad financiera. Todos los representantes del servicio de atención al cliente son miembros del rol CSR. Un usuario con la autorización SECADM crea un permiso de fila de acuerdo a ella para cada grupo de personal de la entidad financiera. Después de activar el acceso de control de acceso de nivel de fila para una tabla CUSTOMER, en la sentencia SELECT las condiciones de búsqueda de ambos permisos de fila se fusionarán en la sentencia y se combinarán con el operador lógico OR para controlar el conjunto de filas al que puede acceder cada grupo.

```
CREATE PERMISSION TELLER_ROW_ACCESS ON CUSTOMER
FOR ROWS WHERE VERIFY_ROLE_FOR_USER
  (SESSION_USER, 'TELLER') = 1 AND
  BRANCH = (SELECT HOME_BRANCH FROM INTERNAL_INFO
            WHERE EMP_ID = SESSION_USER)
ENFORCED FOR ALL ACCESS
ENABLE;

CREATE PERMISSION CSR_ROW_ACCESS ON CUSTOMER
FOR ROWS WHERE VERIFY_ROLE_FOR_USER(SESSION_USER, 'CSR') = 1
ENFORCED FOR ALL ACCESS
ENABLE;
```

CREATE PROCEDURE

La sentencia CREATE PROCEDURE define un procedimiento en el servidor actual.

Mediante esta sentencia, pueden crearse tres tipos distintos de procedimientos. Cada uno de los tipos se describe por separado.

- Externo. El cuerpo del procedimiento se escribe en un lenguaje de programación. Al ejecutable externo hace referencia un procedimiento que se ha definido en el servidor actual, junto con diversos atributos del procedimiento.
- Derivado. El cuerpo del procedimiento forma parte del procedimiento derivado, al que hace referencia el procedimiento derivado definido en el servidor actual, junto con distintos atributos del procedimiento. Un procedimiento derivado cuyo procedimiento fuente se encuentra en una fuente de datos también se denomina *procedimiento federado*.
- SQL. El cuerpo del procedimiento se escribe en SQL y se define en el servidor actual, junto con diversos atributos del procedimiento.

La sentencia CREATE PROCEDURE se puede enviar en formato ofuscado. En una sentencia ofuscada, solamente son legibles el nombre del procedimiento y sus parámetros. El resto de la sentencia se codifica de tal manera que no es legible, pero que el servidor de bases de datos puede descodificar. Las sentencias ofuscadas se pueden generar mediante una llamada a la función DBMS_DDL.WRAP.

CREATE PROCEDURE (externo)

La sentencia CREATE PROCEDURE (externo) define un procedimiento externo en el servidor actual.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, una de las autorizaciones siguientes:
 - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del procedimiento no hace referencia a un esquema existente
 - Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente
- Autorización DBADM

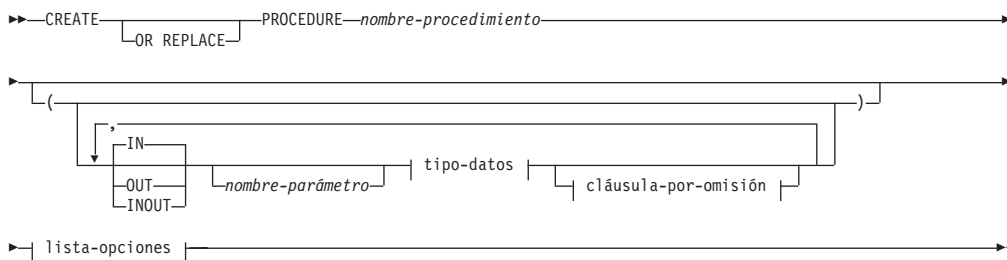
Para crear un procedimiento no delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización DBADM

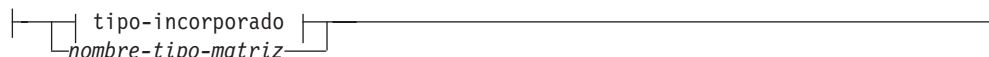
Para crear un procedimiento delimitado, no necesita autorizaciones ni privilegios adicionales.

Para sustituir un procedimiento existente, el ID de autorización de la sentencia debe ser el propietario del procedimiento existente (SQLSTATE 42501).

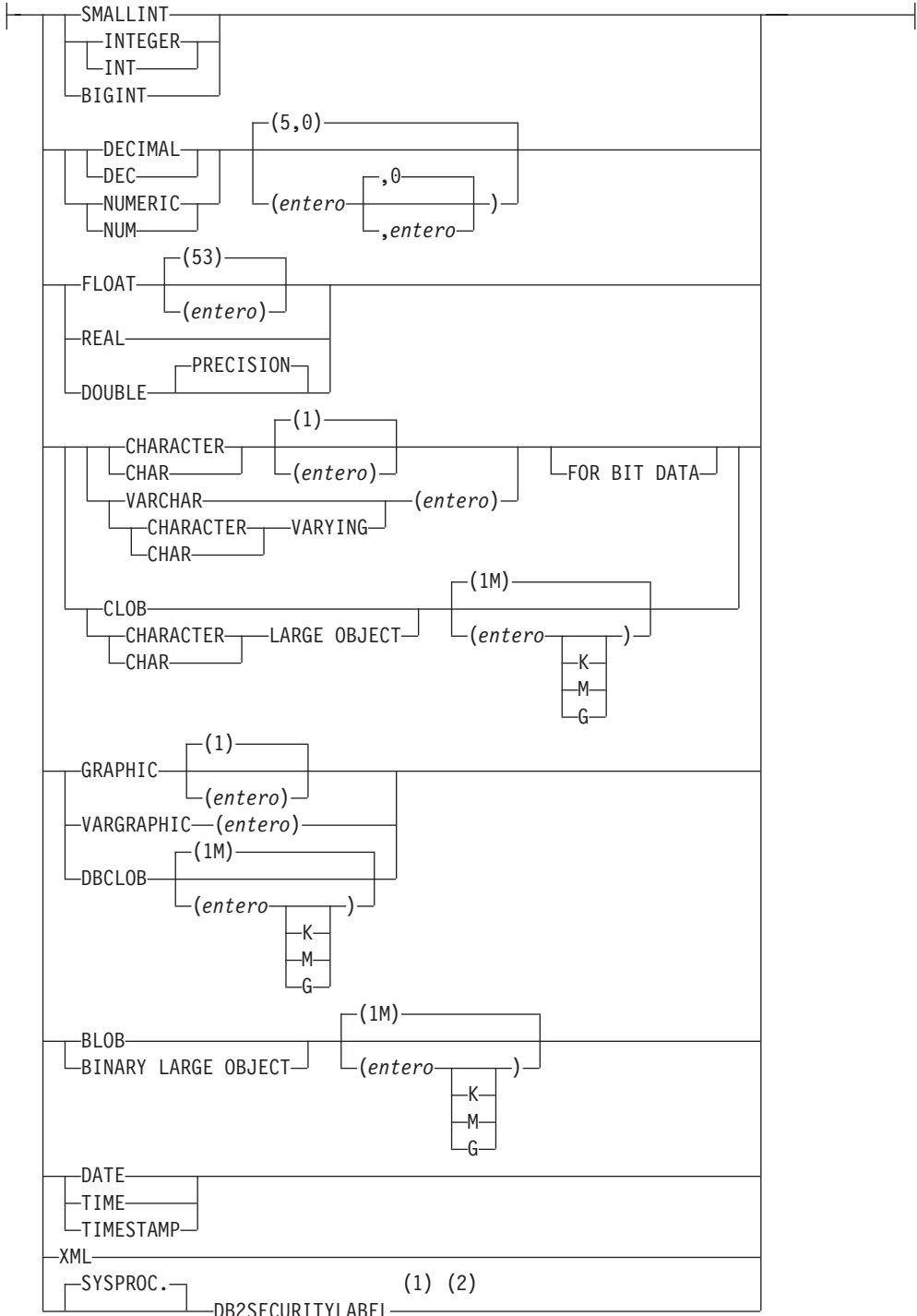
Sintaxis



tipo-datos:



tipo-incorporado:

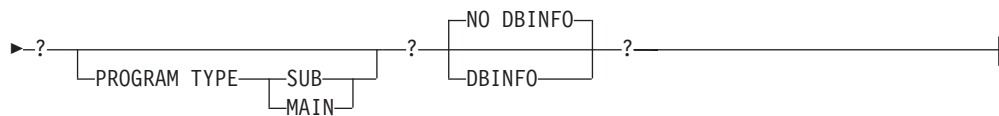
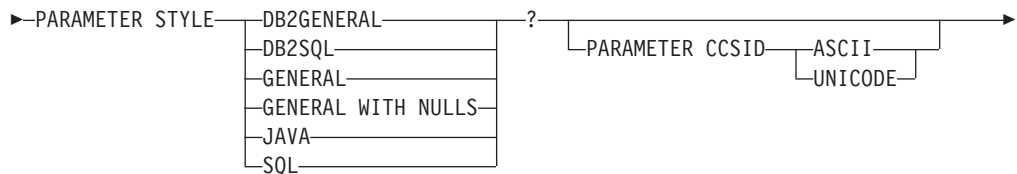
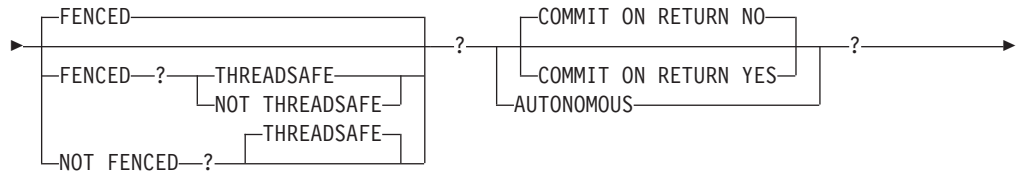
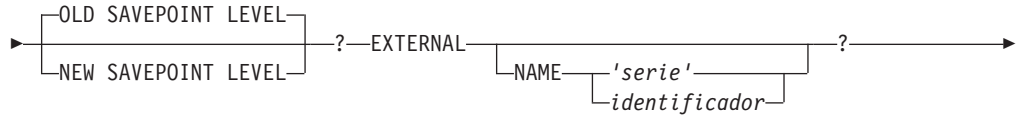
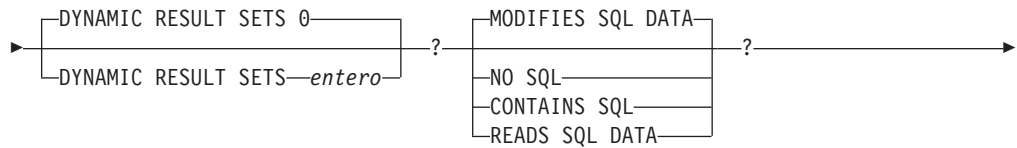
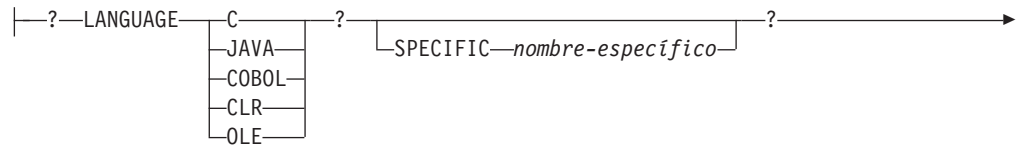


CREATE PROCEDURE (externo)

cláusula-por-omisión:



lista-opciones:



Notas:

- 1 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 2 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

Descripción**OR REPLACE**

Especifica que se debe sustituir la definición del procedimiento si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre el procedimiento no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para el procedimiento en el servidor actual. Para sustituir un procedimiento ya existente, el nombre específico y el nombre de procedimiento de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de procedimiento de la antigua definición, o la signatura de la nueva definición debe coincidir con la signatura de la antigua definición. De lo contrario, se creará un nuevo procedimiento.

nombre-procedimiento

Indica el nombre del procedimiento que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. El formato no calificado de un *nombre-procedimiento* es un identificador SQL. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de 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). No es necesario que el nombre no calificado, junto con el número de parámetros, sea exclusivo en los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

(IN | OUT | INOUT nombre-parámetro tipo-datos cláusula-por-omisión,...)

Identifica los parámetros del procedimiento y especifica el modo, nombre de parámetro opcional, tipo de datos y valor por omisión opcional de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el procedimiento va a esperar.

En un esquema, no se permite que dos procedimientos que se denominen igual tengan exactamente el mismo número de parámetros. Una signatura duplicada devuelve un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

CREATE PROCEDURE (externo)

la segunda sentencia no se ejecutará satisfactoriamente porque el número de parámetros del procedimiento es el mismo, aunque los tipos de datos no lo sean.

Si el procedimiento devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT

Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT

Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro

Opcionalmente, especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos

Especifica el tipo de datos del parámetro. No se puede especificar un tipo estructurado (SQLSTATE 429BB).

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Sólo se pueden especificar tipos de datos incorporados que tienen una correspondencia en el lenguaje que se está utilizando para escribir el procedimiento.

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de un procedimiento es una versión serializada del valor XML que se pasa como parámetro en la llamada de procedimiento, los parámetros del tipo XML deben declararse mediante la sintaxis XML AS CLOB(*n*).
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- La coma flotante decimal no se soporta con los lenguajes C, Java COBOL, CLR y OLE (SQLSTATE 42613).

nombre-tipo-matriz

Especifica el nombre de un tipo de matriz definido por el usuario. Si se especifica el *nombre-tipo-matriz* sin un nombre de esquema, el tipo de matriz se resuelve buscando en los esquemas de la vía de acceso de SQL. La matriz debe ser una matriz común y el procedimiento debe ser un procedimiento Java definido con la cláusula PARAMETER STYLE JAVA (SQLSTATE 428H2).

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que pueden especificarse como valor por omisión son los mismos que pueden

especificarse para un valor por omisión de columna (consulte el apartado "*cláusula-por-omisión*" en la sentencia "CREATE TABLE"). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La *expresión* puede ser cualquier expresión del tipo descrito en "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión en las situaciones siguientes:

- Para parámetros INOUT o OUT (SQLSTATE 42601)
- Para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB)

SPECIFIC nombre-específico

Proporciona un nombre exclusivo para la instancia del procedimiento que se está definiendo. Este nombre específico puede utilizarse al modificar, descartar o realizar un comentario en el procedimiento. No puede utilizarse nunca para invocar el procedimiento. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar otra instancia de rutina 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.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, este 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 será 'SQL' seguido de una indicación de fecha y hora expresada en forma de caracteres: 'SQLaammddhmmssxx'.

DYNAMIC RESULT SETS entero

Indica el límite superior estimado de los conjuntos de resultados devueltos para el procedimiento.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

Indica si el procedimiento emite sentencias de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que el procedimiento no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL

Indica que el procedimiento puede ejecutar sentencias de SQL que no lean ni modifiquen datos de SQL (SQLSTATE 38004). Las sentencias que no están soportadas en ningún procedimiento devuelven un error diferente (SQLSTATE 38003).

CREATE PROCEDURE (externo)

READS SQL DATA

Indica que en el procedimiento pueden incluirse algunas sentencias de SQL que no modifiquen datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no están soportadas en ningún procedimiento devuelven un error diferente (SQLSTATE 38003).

MODIFIES SQL DATA

Indica que el procedimiento puede ejecutar cualquier sentencia de SQL excepto las que no están soportadas en ningún procedimiento (SQLSTATE 38003).

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.

CALLED ON NULL INPUT

CALLED ON NULL INPUT siempre se aplica a los procedimientos. Esto significa que el procedimiento recibe llamada independientemente de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento comprobar si hay valores de argumento nulos.

OLD SAVEPOINT LEVEL o NEW SAVEPOINT LEVEL

Especifica si este procedimiento establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información acerca de los niveles de punto de salvaguarda, consulte la sección "Normas" de la descripción de la sentencia SAVEPOINT.

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje para la que está escrito el cuerpo del procedimiento.

C Esto significa que el gestor de bases de datos llamará al procedimiento como si fuera un procedimiento C. El procedimiento debe ajustarse al convenio de llamada y enlace del lenguaje C, definido por el prototipo C de ANSI estándar.

JAVA

Esto significa que el gestor de bases de datos llamará al procedimiento como 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.

CLR

Esto significa que el gestor de bases de datos llamará al procedimiento como método de una clase .NET. En este momento, sólo se soporta LANGUAGE CLR para los procedimientos que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE

Esto significa que el gestor de bases de datos llamará al procedimiento

CREATE PROCEDURE (externo)

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 la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para los procedimientos almacenados en DB2 para los sistemas operativos Windows.

THREADSAFE no puede especificarse para los procedimientos que se han definido con LANGUAGE OLE (SQLSTATE 42613).

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". Si la cláusula NAME no tiene un formato correcto, se devuelve un error (SQLSTATE 42878).

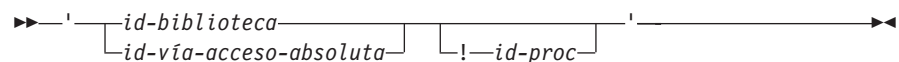
NAME 'serie'

Esta cláusula identifica el nombre del código escrito por el usuario que implanta el procedimiento que se está definiendo.

La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes. 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 que ha de crearse (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.



El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

id-biblioteca

Identifica el nombre de la biblioteca donde reside el procedimiento. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

- En los sistemas UNIX, si se ha especificado 'myfunc' como *library_id*, y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará el procedimiento en la biblioteca /u/production/sqllib/function/myproc si se ha especificado FENCED o en /u/production/sqllib/function/unfenced/myproc si se ha especificado NOT FENCED.

CREATE PROCEDURE (externo)

- En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

Los procedimientos almacenados en cualquiera de estos directorios no utilizan ninguno de los atributos registrados.

id-vía-acceso-absoluta

Identifica el nombre completo de vía de acceso del procedimiento.

En sistemas UNIX, por ejemplo, /u/jchui/mylib.systems haría que el gestor de bases de datos buscara el procedimiento myproc en /u/jchui/mylib.

En los sistemas operativos Windows, 'd:\mylib\myproc.dll' haría que el gestor de bases de datos cargara el archivo myproc.dll desde el directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id-proc

Identifica el nombre del punto de entrada del procedimiento que se debe invocar. El carácter de exclamación (!) sirve como delimitador entre el ID de biblioteca y el ID de procedimiento. '!proc8' indicaría al gestor de bases de datos que buscara la biblioteca en la ubicación especificada por el *id-vía-acceso-absoluta* y que utilizara el punto de entrada proc8 dentro de esa biblioteca.

Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

El cuerpo de cada procedimiento debe encontrarse en un directorio que se haya montado y que esté disponible en todas las particiones de base de datos.

- Para LANGUAGE JAVA:

La *serie* especificada contiene el identificador del archivo jar opcional, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar el procedimiento que ha de crearse (CREATE). 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*, este deberá existir cuando se ejecute la sentencia CREATE PROCEDURE. Sin embargo, cuando se llama al procedimiento, el identificador de clase y el identificador de método deben existir y poderse acceder desde la máquina del servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42884).

► ' [*id-jar* :] *id-clase* [.] *id-método* ' ◀

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

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 de paquete completo, por ejemplo, 'misPaqs.StoredProcs'. La máquina virtual Java buscará en el directorio '..\myPacks\StoredProcs\' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '..\myPacks\StoredProcs\'.

id-método

Identifica el nombre de método de la clase Java que se invoca.

- Para LANGUAGE CLR:

La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método dentro de la clase que el gestor de bases de datos invoca para ejecutar el procedimiento que se está creando. No es necesario que existan el módulo, la clase y el método cuando se ejecute la sentencia CREATE PROCEDURE. Sin embargo, cuando se llama al procedimiento, el módulo, la clase y el método deben existir y poderse acceder desde la máquina de servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42284).

Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe saber que se utiliza la infraestructura .NET en un procedimiento, para tomar las decisiones necesarias en tiempo de ejecución. Todos los procedimientos que utilicen la infraestructura .NET deben estar catalogados como 'LANGUAGE CLR'.

►— '*ensamblaje*—:—*id-clase*—!—*id-método*—' —————►

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

ensamblaje

Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de la instancia de DB2 (por ejemplo, c:\DB2\function). Si el archivo reside en un subdirectorio del directorio function de la instancia, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de la instancia es c:\DB2 y el archivo de ensamblaje es c:\DB2\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

id-clase

Especifica el nombre de la clase del ensamblaje proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la

CREATE PROCEDURE (externo)

clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método

Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

La serie 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 que la sentencia crea. 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* | *idcls* | *id-método* ' ◀◀

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

idprog

Identifica el identificador de programa del objeto OLE.

El gestor de bases de datos no interpreta un *idprog*; sólo lo reenvía al controlador de automatización de OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y debe dar soporte al enlace tardío (llamado también enlace basado en IDispatch). Según especifica el convenio, los *idprog* tienen el formato siguiente:

<nombre_programa>.<nombre_componente>.<versión>

Puesto que sólo se trata de un convenio y no de una norma, los *idprog* pueden tener, de hecho, un formato distinto.

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 el formato:

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnn}

donde 'n' es un carácter alfanumérico. El gestor de bases de datos no interpreta un *idcls*; sólo lo reenvía a las API de OLE en tiempo de ejecución.

id-método

Identifica el nombre del método del objeto OLE que se debe invocar.

NAME *identificador*

Este *identificador* especificado es un identificador de 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 del esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

FENCED o NOT FENCED

Esta cláusula especifica si el procedimiento 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 se registra como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) contra el acceso del procedimiento. Todos los procedimientos tienen la opción de ejecutarse como FENCED o como NOT FENCED. En general, un procedimiento que se ejecute como FENCED no funcionará tan bien como otro de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:

La utilización de NOT FENCED para los procedimientos que no se han comprobado de forma adecuada puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 toman algunas precauciones ante varios de los tipos comunes de anomalías inadvertidas que podrían producirse, pero no pueden garantizar la integridad completa si se utilizan métodos NOT FENCED.

Necesita la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED) para registrar un procedimiento como NOT FENCED. Sólo puede especificarse FENCED para un procedimiento con LANGUAGE OLE o NOT THREADSAFE.

No podrán crearse procedimientos LANGUAGE CLR cuando se especifique la cláusula NOT FENCED (SQLSTATE 42601).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que el procedimiento es seguro para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si el procedimiento se ha definido con un LANGUAGE distinto de OLE:

- Si el procedimiento se ha definido como THREADSAFE, el gestor de bases de datos puede invocar el procedimiento en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, un procedimiento no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Los procedimientos FENCED y NOT FENCED pueden ser THREADSAFE.
- Si el procedimiento se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará el procedimiento en el mismo proceso que otra rutina.

Para los procedimientos FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si el procedimiento se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para los procedimientos NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

COMMIT ON RETURN

Indica si debe efectuarse una confirmación cuando el procedimiento vuelve. El valor por omisión es NO.

CREATE PROCEDURE (externo)

NO No se puede realizar una confirmación cuando vuelve el procedimiento.

YES

Se efectúa una confirmación cuando el procedimiento vuelve si la sentencia CALL devuelve un SQLCODE positivo

La operación de confirmación incluye el trabajo que efectúa el proceso de aplicación de llamada y el procedimiento.

Si el procedimiento devuelve conjuntos de resultados, los cursores asociados a dichos conjuntos deben haberse definido como WITH HOLD para poderlos utilizar después de la confirmación.

AUTONOMOUS

Indica que el procedimiento debe ejecutarse en su propio ámbito de transacción autónomo.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si el procedimiento realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). El valor por omisión es EXTERNAL ACTION. Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones que suponen que el procedimiento no tiene ningún impacto externo.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación.

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

PARAMETER STYLE

Esta cláusula sirve para especificar los convenios utilizados para pasar parámetros a los procedimientos y devolver el valor de los mismos.

DB2GENERAL

Significa que el procedimiento utilizará un convenio de pase de parámetros definido para su uso con métodos Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

DB2SQL

Además de los parámetros existentes en la sentencia CALL, se pasan los argumentos siguientes al procedimiento:

- Un vector que contiene un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento
- El nombre específico del procedimiento
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR o OLE.

GENERAL

Significa que el procedimiento utilizará un mecanismo de pase de parámetros por medio del cual el procedimiento recibirá los parámetros especificados en la sentencia CALL. Los parámetros se envían directamente, tal como espera el lenguaje; la estructura de la SQLDA no se utiliza. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

Los indicadores nulos *no* se envían directamente al programa.

GENERAL WITH NULLS

Además de los parámetros de la sentencia CALL que ha especificado bajo GENERAL, se pasa otro argumento al procedimiento. Este argumento adicional es un vector de indicadores nulos, uno para cada parámetro de la sentencia CALL. En C, sería una matriz de enteros cortos. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

JAVA

Significa que el procedimiento utilizará un convenio para el envío de parámetros que se ajusta a la especificación para el lenguaje Java y las 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.

SQL

Además de los parámetros existentes en la sentencia CALL, se pasan los argumentos siguientes al procedimiento:

- Un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento
- El nombre específico del procedimiento
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR o OLE.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es la página de códigos de la base de datos.

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de

CREATE PROCEDURE (externo)

caracteres están en UTF-8. En los dos casos, cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es 1208.

Si la base de datos no es una base de datos Unicode y se crea un procedimiento con `PARAMETER CCSID UNICODE`, el procedimiento no puede tener ningún tipo gráfico, tipo XML ni tipos definidos por el usuario (`SQLSTATE 560C1`). Los procedimientos `PARAMETER CCSID UNICODE` sólo se pueden llamar desde un cliente de DB2 Versión 8.1 o posterior (`SQLSTATE 42997`).

Si la base de datos no es Unicode y se ha especificado el orden de clasificación alternativo en la configuración de la base de datos, se pueden crear procedimientos con `PARAMETER CCSID ASCII` o `PARAMETER CCSID UNICODE`. Todos los datos que se pasan al procedimiento o desde él se convertirán en la página de códigos adecuada.

Esta cláusula no se puede especificar con `LANGUAGE OLE`, `LANGUAGE JAVA` ni `LANGUAGE CLR` (`SQLSTATE 42613`).

PROGRAM TYPE

Especifica si el procedimiento espera parámetros del estilo de una rutina principal o una subrutina. El valor por omisión es `SUB`.

SUB

El procedimiento espera que los parámetros se pasen como argumentos por separado.

MAIN

El procedimiento espera que los parámetros se pasen como un contador de argumentos y como un vector de argumentos (`argc`, `argv`). El nombre del procedimiento que debe invocarse también debe ser "main". Los procedimientos almacenados de este tipo deben seguir incorporándose de la misma forma que para una biblioteca compartida, en lugar de como se haría para un ejecutable autónomo. `PROGRAM TYPE MAIN` sólo es válido cuando la cláusula `LANGUAGE` especifica `C`, `COBOL` o `CLR`.

DBINFO o NO DBINFO

Indica si se pasa información específica conocida por DB2 al procedimiento cuando se invoca como argumento en tiempo de invocación adicional (`DBINFO`) o no (`NO DBINFO`). `NO DBINFO` es el valor por omisión. `DBINFO` no puede utilizarse para `LANGUAGE OLE` (`SQLSTATE 42613`). Tampoco recibe soporte para `PARAMETER STYLE JAVA` o `DB2GENERAL`.

Si se especifica `DBINFO`, se pasa al procedimiento una estructura 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 la aplicación - ID de autorización del usuario que se ha conectado con la base de datos (registro especial `SYSTEM_USER`).
- Página de códigos - identifica la página de códigos de la base de datos.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca el procedimiento.
- Plataforma - contiene el tipo de plataforma del servidor.

La estructura DBINFO es común para todas las rutinas externas y contiene campos adicionales que no están relacionados con los procedimientos.

Si cambia el ID de autorización de sesión (registro especial SESSION_USER) con la sentencia SET SESSION AUTHORIZATION, el ID de autorización de la aplicación devolverá el valor del registro especial SYSTEM_USER.

Normas

- **Restricciones de la rutina autónoma:** las rutinas autónomas no pueden devolver conjuntos de resultados y no admiten los siguientes tipos de datos de parámetros (SQLSTATE 428H2):

- Tipos de cursor
- Tipos estructurados
- XML

No se puede hacer referencia a variables globales de tipos de cursor dentro del ámbito autónomo.

Notas

- La creación de un procedimiento 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- Un procedimiento que se llama desde una sentencia de SQL compuesto (en línea) se ejecutará como si se hubiera creado especificando NEW SAVEPOINT LEVEL, aunque se haya especificado OLD SAVEPOINT LEVEL o se haya tomado por omisión al crearse el procedimiento.
- Los parámetros de XML sólo están soportados en los procedimientos externos de LANGUAGE JAVA si se especifica la cláusula PARAMETER STYLE DB2GENERAL.
- **Establecimiento del valor por omisión:** los parámetros de un procedimiento que se definen con un valor por omisión se establecen en su valor por omisión cuando se invoca el procedimiento, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca el procedimiento.
- **Privilegios:** el definidor de un procedimiento siempre recibe el privilegio WITH GRANT OPTION de EXECUTE en el procedimiento, así como el derecho de descartar el procedimiento. Cuando el procedimiento se utiliza en una sentencia de SQL, el usuario que define el procedimiento debe disponer del privilegio EXECUTE para cualquiera de los paquetes que el procedimiento utiliza.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - RESULT SETS puede especificarse en lugar de DYNAMIC RESULT SETS.
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT.
 - DB2GENRL puede especificarse en lugar de DB2GENERAL.
 - SIMPLE CALL puede especificarse en lugar de GENERAL.
 - SIMPLE CALL WITH NULLS puede especificarse en lugar de GENERAL WITH NULLS.

CREATE PROCEDURE (externo)

- PARAMETER STYLE DB2DARI recibe soporte.

La sintaxis siguiente se acepta como comportamiento por omisión:

- ASUTIME NO LIMIT
- NO COLLID
- STAY RESIDENT NO
- CCSID UNICODE en una base de datos Unicode
- CCSID ASCII en una base de datos que no es Unicode si no se especifica
PARAMETER CCSID UNICODE

Ejemplos

- *Ejemplo 1:* Cree la definición de procedimiento para un procedimiento, grabado en Java, al que se pasa un número de pieza y que devuelve el coste de la pieza y la cantidad disponible actualmente.

```
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, escrito en C, al que se pasa un número de ensamblaje y que devuelve el número de piezas que componen el ensamblaje, el coste total de las piezas y un conjunto de resultados en el que se listan los números de pieza, la cantidad y el coste unitario de cada pieza.

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,  
    OUT NUM_PARTS INTEGER,  
    OUT COST DOUBLE)  
EXTERNAL NAME 'parts!assembly'  
DYNAMIC RESULT SETS 1 NOT FENCED  
LANGUAGE C PARAMETER STYLE GENERAL
```


CREATE PROCEDURE (derivado)

La sentencia CREATE PROCEDURE (derivado) define un procedimiento (el *procedimiento derivado*) que está basado en otro procedimiento (el *procedimiento fuente*). En sistemas federados, se entiende por *procedimiento federado* un procedimiento derivado cuyo procedimiento fuente se encuentra en una fuente de datos soportada.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del procedimiento no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente
- Autorización DBADM

Para los orígenes de datos que requieren una correlación de usuarios, los privilegios contenidos en el ID de autorización en la fuente de datos deben incluir el privilegio para seleccionar datos de las tablas del catálogo remoto.

Para sustituir un procedimiento existente, el ID de autorización de la sentencia debe ser el propietario del procedimiento existente (SQLSTATE 42501).

Sintaxis

```

▶▶ CREATE OR REPLACE PROCEDURE nombre-procedimiento
▶ | cláusula-procedimiento-fuente | | lista-opciones |

```

cláusula-procedimiento-fuente:

```

| SOURCE | nombre-objeto-fuente | ( )
| NUMBER OF PARAMETERS | entero
▶ UNIQUE ID | id-exclusivo | FOR SERVER nombre-servidor

```

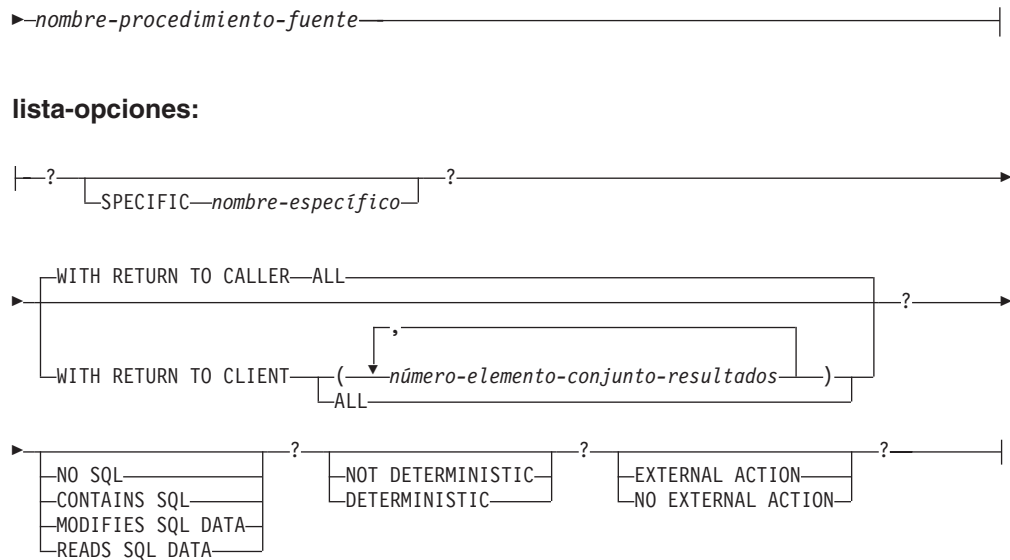
nombre-objeto-fuente:

```

| nombre-esquema-fuente | . | nombre-paquete-fuente | .

```

CREATE PROCEDURE (derivado)



Descripción

OR REPLACE

Especifica que se debe sustituir la definición del procedimiento si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre el procedimiento no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para el procedimiento en el servidor actual. Para sustituir un procedimiento ya existente, el nombre específico y el nombre de procedimiento de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de procedimiento de la antigua definición, o la signature de la nueva definición debe coincidir con la signature de la antigua definición. De lo contrario, se creará un nuevo procedimiento.

nombre-procedimiento

Los nombres del procedimiento derivado que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. El formato no calificado de un *nombre-procedimiento* es un identificador SQL. 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 o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros, no deben identificar un procedimiento que se haya descrito en el catálogo (SQLSTATE 42723). No es necesario que el nombre no calificado, junto con el número de parámetros, sea exclusivo en los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

En un sistema federado, un *nombre-procedimiento* es el nombre del procedimiento del servidor federado.

SOURCE *nombre-objeto-fuente*

Especifica el procedimiento fuente que utiliza el procedimiento que se está

definiendo. En sistemas federados, el procedimiento fuente es un procedimiento que se encuentra en una fuente de datos soportada.

nombre-esquema-fuente

Identifica el nombre de esquema del procedimiento fuente. Si se utiliza un nombre de esquema para identificar el procedimiento fuente, el *nombre-esquema-fuente* debe especificarse en la sentencia CREATE PROCEDURE (derivado). Si el *nombre-esquema-fuente* contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

nombre-paquete-fuente

Identifica el nombre de paquete del procedimiento fuente. El *nombre-paquete-fuente* sólo se aplica a los orígenes de datos Oracle. Si se utiliza un nombre de esquema para identificar el procedimiento fuente, el *nombre-paquete-fuente* debe especificarse en la sentencia CREATE PROCEDURE (derivado). Si el *nombre-paquete-fuente* contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

nombre-procedimiento-fuente

Identifica el nombre de procedimiento del procedimiento fuente. Si el *nombre-procedimiento-fuente* contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

()

Indica que el número de parámetros es cero.

NUMBER OF PARAMETERS *entero*

Especifica el número de parámetros para el procedimiento fuente. El valor mínimo para *entero* es 0 y el valor máximo es 32.767.

UNIQUE ID *constante-serie*

Proporciona un modo exclusivo de identificar de modo exclusivo el procedimiento fuente cuando hay varios procedimientos en la fuente de datos con el mismo nombre, esquema y número de parámetros. El valor de la *constante-serie* que tiene una longitud máxima de 128, se interpreta exclusivamente mediante cada fuente de datos.

FOR SERVER *nombre-servidor*

Especifica una definición de servidor que se ha registrado utilizando la sentencia CREATE SERVER.

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia del procedimiento derivado que se está definiendo. Este nombre específico puede utilizarse al modificar, descartar o realizar un comentario en el procedimiento almacenado. Este nombre no puede utilizarse nunca para invocar el procedimiento derivado. El formato no calificado de un *nombre-específico* es un identificador SQL. El formato calificado del *nombre-específico* es un *nombre-esquema* seguido de un punto y un identificador de SQL. El valor del *nombre-específico*, 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 igual a un *nombre-procedimiento* existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, este deberá ser el mismo que el calificador explícito o implícito del *nombre-procedimiento* o se generará un error (SQLSTATE 42882).

CREATE PROCEDURE (derivado)

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo será 'SQL' seguido de una indicación de fecha y hora expresada en forma de caracteres: 'SQLaammddhhmmssxxx'.

WITH RETURN TO CALLER o WITH RETURN TO CLIENT

Indica si se han manejado los conjuntos de resultados del procedimiento fuente. Si el procedimiento fuente no procede de una fuente de datos Oracle, el único conjunto de resultados se devuelve al cliente o al que llama; si el procedimiento fuente se ha codificado para que devuelva más de un conjunto de resultados, sólo se devuelve el primer conjunto de resultados al cliente o al que llama. El valor por omisión es WITH RETURN TO CALLER.

WITH RETURN TO CALLER ALL

Indica que todos los conjuntos de resultados del procedimiento fuente se han devuelto al que llama.

WITH RETURN TO CLIENT

Especifica los conjuntos de resultados del procedimiento fuente que se devuelven directamente a la aplicación de cliente. El valor de los conjuntos de resultados dinámicos en la fuente de datos debe ser mayor que 0 para que se devuelva un conjunto de resultados.

(número-elemento-conjunto-resultados, ...)

Especifica una lista no vacía de conjuntos de resultados para devolver a la aplicación de cliente (SQLSTATE 42601). Un *número-elemento-conjunto-resultados* identifica un conjunto de resultados en función del orden de devolución de los conjuntos de resultados, donde 1 identifica el primer conjunto de resultados, 2 el segundo, y así sucesivamente. Un *número-elemento-conjunto-resultados* superior al número total de conjuntos de resultados devueltos se pasará por alto. Cada *número-elemento-conjunto-resultados* debe ser un valor entero mayor que cero (SQLSTATE 42815) y no puede sobrepasar el valor de una constante de enteros pequeños (SQLSTATE 42820). La lista de conjuntos de resultados para devolver a la aplicación de cliente no debe contener valores duplicados y debe especificarse en orden ascendente (SQLSTATE 42815). Los conjuntos de resultados siempre se procesan en el orden en el que se devuelven del procedimiento fuente.

Los conjuntos de resultados que no se identifican en la lista para devolver a la aplicación de cliente se devuelven al emisor.

Nota: Esta lista de conjuntos de resultados para devolver a la aplicación de cliente solamente debe utilizarse con procedimientos fuente que se sabe que de forma coherente devuelven conjuntos de resultados destinados al cliente en la misma posición de la lista de conjuntos de resultados cada vez que se ejecutan. Es posible que un procedimiento fuente devuelva conjuntos diferentes de conjuntos de resultados cada vez que se ejecuta, en función de la lógica interna del procedimiento. Si este fuera el caso, especifique WITH RETURN TO CALLER ALL o bien WITH RETURN TO CLIENT ALL en su lugar, y codifique la aplicación para manejar este caso.

ALL

Indica que todos los conjuntos de resultados del procedimiento fuente se han devuelto al cliente.

NO SQL, CONTAINS SQL, MODIFIES SQL DATA, READS SQL DATA

Indica el nivel de acceso a datos para las sentencias de SQL que se han incluido en el procedimiento derivado. Debido a que el procedimiento fuente

del procedimiento derivado no está situado en el servidor federado, el nivel especificado no se aplica durante la ejecución del procedimiento fuente en la fuente de datos. Si hay alguna discrepancia entre lo que se especifica para el procedimiento derivado y lo que hace realmente el procedimiento derivado en la fuente de datos, es posible que se produzcan incoherencias de datos. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es MODIFIES SQL DATA. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

DETERMINISTIC o NOT DETERMINISTIC

Especifica si el procedimiento derivado devuelve siempre los mismos resultados para los valores de argumentos determinados (DETERMINISTIC) o si el procedimiento derivado depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Un procedimiento derivado DETERMINISTIC debe siempre devolver el mismo resultado si se invoca de forma sucesiva con entradas de datos idénticas. Esta cláusula no afecta actualmente el proceso del procedimiento. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es NOT DETERMINISTIC. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si el procedimiento derivado realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). Si se especifica la cláusula NO EXTERNAL ACTION, la base de datos federada utiliza la optimización que presupone que el procedimiento derivado no tiene un impacto externo. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es EXTERNAL ACTION. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

Normas

- Si *nombre-objeto-fuente*, junto con las cláusulas NUMBER OF PARAMETERS y UNIQUE ID no identifican un procedimiento en una fuente de datos, se devuelve un error (SQLSTATE 42883). Si se identifica más de un procedimiento, se devuelve un error (SQLSTATE 42725).
- Si se especifica la cláusula UNIQUE ID y la fuente de datos no da soporte a los ID exclusivos, se devuelve un error (SQLSTATE 42883).

Notas

- Para poder registrar un procedimiento federado para una fuente de datos, el servidor federado debe estar configurado para acceder a esa fuente de datos. Esta configuración incluye la creación de un derivador para la fuente de datos, la definición del servidor para el servidor para la fuente de datos y la creación de las correlaciones de usuarios entre el servidor federado y el servidor fuente de datos para los orígenes de datos que requieren correlaciones de usuarios.
- *Creación de procedimientos que no son válidos inicialmente*: si un objeto al que se hace referencia en el cuerpo del procedimiento no existe o se ha marcado como no válido o el definidor no tiene temporalmente los privilegios para acceder al objeto y, si el parámetro de configuración de la base de datos

CREATE PROCEDURE (derivado)

auto_reval no se ha establecido en DISABLED, el procedimiento seguirá creándose satisfactoriamente. El procedimiento se marcará como no válido y se volverá a validar la próxima vez que se invoque.

- A diferencia de los procedimientos de SQL y externos definidos en el servidor federado, los procedimientos federados no heredan los registros especiales del que llama, incluso aquellos cuyos *nombre-objeto-remoto* hace referencia a un procedimiento de una fuente de datos DB2.
- Si se cambia la definición del procedimiento fuente (por ejemplo, se modifica un tipo de datos de parámetro), se debe descartar y volver a crear el procedimiento federado; de lo contrario, se pueden producir errores cuando se invoque el procedimiento federado.
- Si la longitud del parámetro de procedimiento fuente es mayor de 128, el nombre de parámetro del procedimiento federado se trunca en 128 bytes.
- **Compatibilidades:** no se admite la sintaxis de DataJoiner para Crear apodo de procedimiento almacenado. En la sintaxis de la nueva versión 9, la correlación de tipos de parámetros se maneja de modo similar a los apodos, ya que una búsqueda de catálogo determina el tipo de datos remoto. El tipo de parámetro local se determina mediante la correlación de tipos hacia adelante.

Ejemplos

- *Ejemplo 1:* Crear un procedimiento federado cuyo nombre es FEDEMPLOYEE para un procedimiento Oracle cuyo nombre es EMPLOYEE, utilizando el nombre de esquema remoto USER1, el nombre de paquete remoto P1 en el servidor federado S1, y devolviendo el conjunto de resultados al cliente.

```
CREATE PROCEDURE FEDEMPLOYEE SOURCE USER1.P1.EMPLOYEE  
FOR SERVER S1 WITH RETURN TO CLIENT ALL
```

- *Ejemplo 2:* Crear un procedimiento federado denominado FEDSALARYSTAT para un procedimiento Oracle cuyo nombre es SALARYSTAT, utilizando el nombre de esquema remoto USER1, el nombre de paquete remoto P1 en el servidor federado S1, y devolviendo el primer y el tercer conjunto de resultados al cliente, y el resto de conjuntos de resultados al emisor.

```
CREATE OR REPLACE PROCEDURE FEDSALARYSTAT SOURCE USER1.P1.SALARYSTAT  
FOR SERVER S1 WITH RETURN TO CLIENT(1,3)
```

CREATE PROCEDURE (SQL)

La sentencia CREATE PROCEDURE (SQL) define un procedimiento de SQL en el servidor actual.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

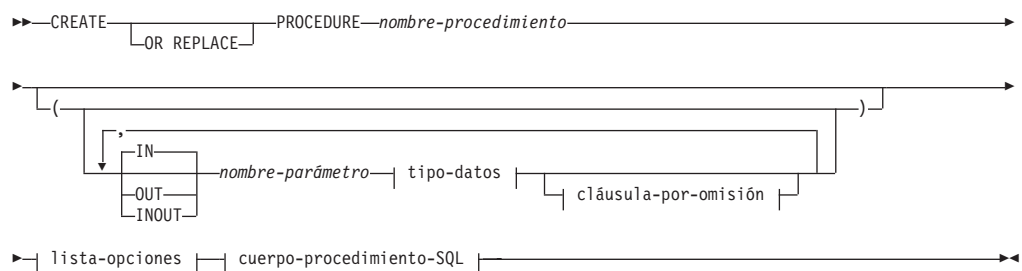
- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito del procedimiento no existe.
- Privilegio CREATEIN en el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente.
- Autorización DBADM

Los privilegios que mantiene el ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL especificadas en el cuerpo del procedimiento.

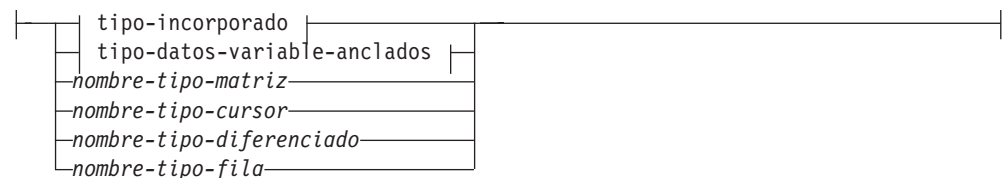
Para sustituir un procedimiento existente, el ID de autorización de la sentencia debe ser el propietario del procedimiento existente (SQLSTATE 42501).

Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE PROCEDURE (SQL) no se tienen en cuenta.

Sintaxis

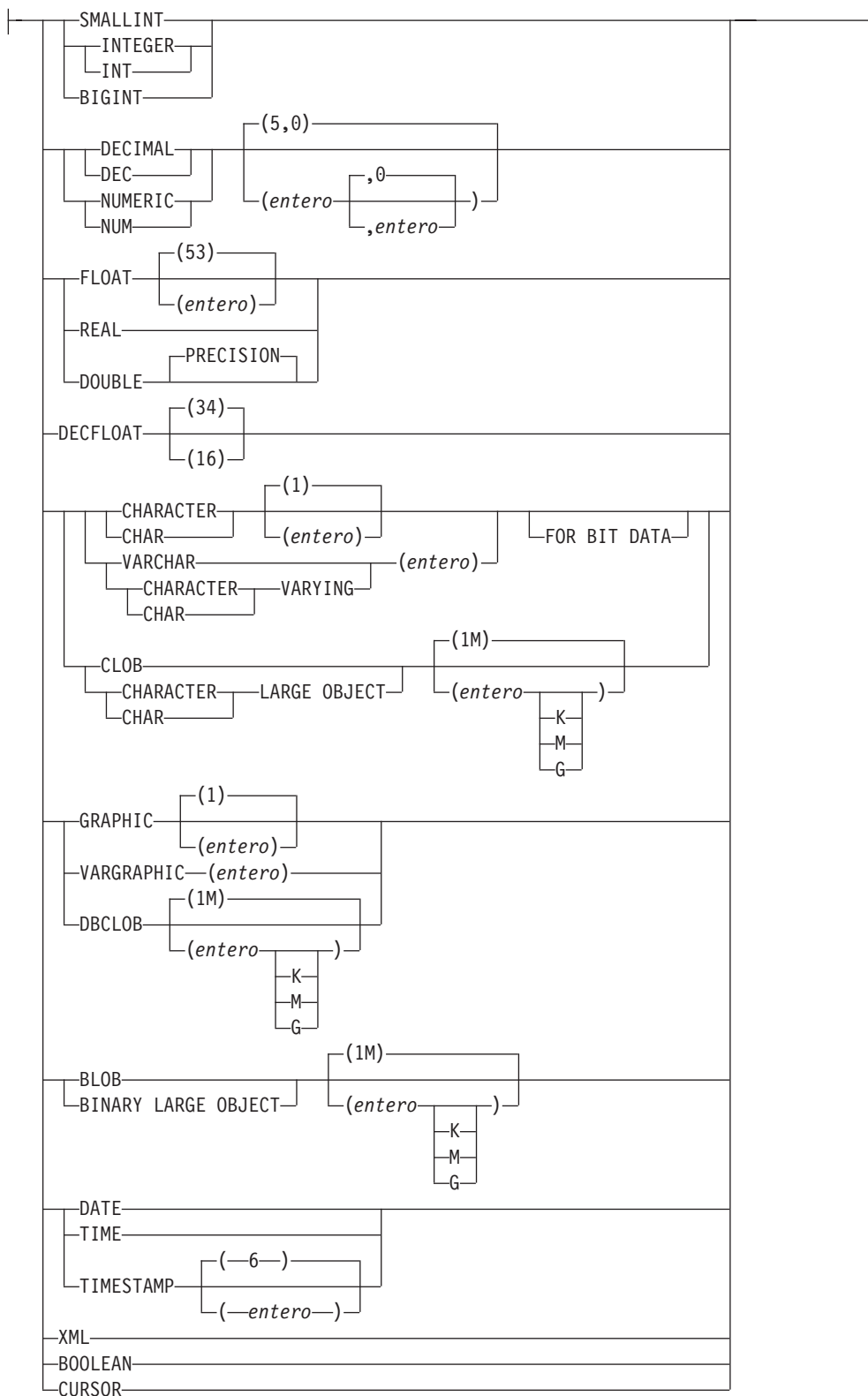


tipo-datos:

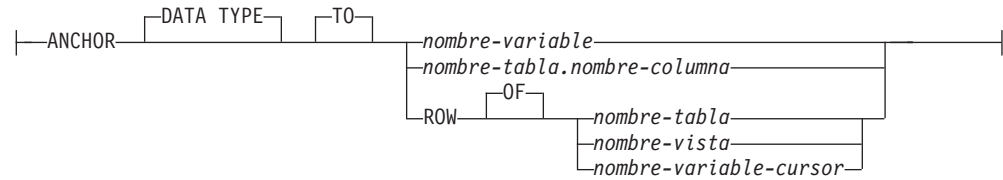


CREATE PROCEDURE (SQL)

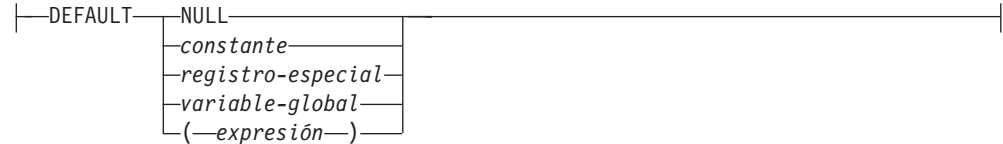
tipo-incorporado:



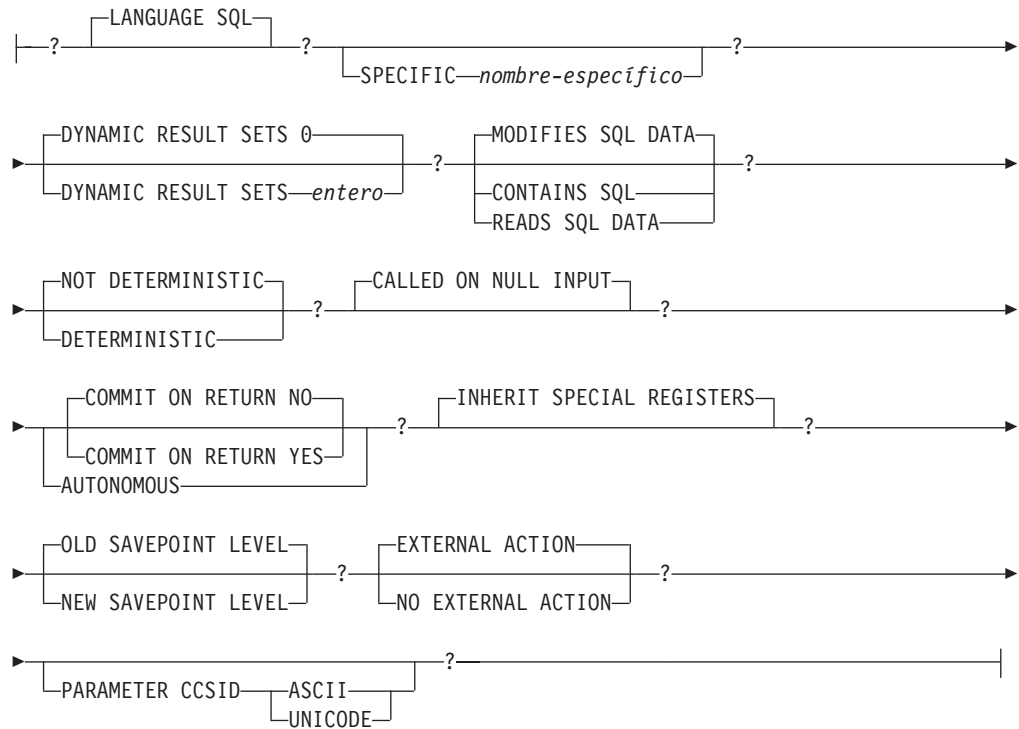
tipo-datos-anclados:



cláusula-por-omisión:



lista-opciones:



cuerpo-procedimiento-SQL:



Descripción

OR REPLACE

Especifica que se debe sustituir la definición del procedimiento si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los

CREATE PROCEDURE (SQL)

privilegios que se han otorgado sobre el procedimiento no se ven afectados por ello. Esta opción sólo puede especificarla el propietario del objeto. Esta opción se ignora si no existe una definición para el procedimiento en el servidor actual. Para sustituir un procedimiento ya existente, el nombre específico y el nombre de procedimiento de la nueva definición tienen que ser los mismos que el nombre específico y el nombre de procedimiento de la antigua definición, o la signatura de la nueva definición debe coincidir con la signatura de la antigua definición. De lo contrario, se creará un nuevo procedimiento.

nombre-procedimiento

Indica el nombre del procedimiento que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. El formato no calificado de un *nombre-procedimiento* es un identificador SQL. 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 de SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros, no deben identificar un procedimiento que se haya descrito en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número de parámetros, es exclusivo dentro de su esquema, pero no es necesario que sea exclusivo 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 devuelve un error (SQLSTATE 42939).

(IN | OUT | INOUT nombre-parámetro tipo-datos cláusula-por-omisión,...)

Identifica los parámetros del procedimiento, y especifica la modalidad, el nombre, el tipo de datos y el valor por omisión opcional 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 este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. 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 se emiten estas 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 no se ejecutará satisfactoriamente porque el número de parámetros del procedimiento es el mismo, aunque los tipos de datos no lo sean.

IN | OUT | INOUT

Especifica la modalidad del parámetro.

Si el procedimiento devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro

del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro

Especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos

Especifica el tipo de datos del parámetro. No se puede especificar tipos estructurados ni tipos de referencia (SQLSTATE 429BB).

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado excepto BOOLEAN y CURSOR, que no pueden especificarse para una tabla, consulte el apartado "CREATE TABLE".

BOOLEAN

Para un booleano.

CURSOR

Para una referencia a un cursor subyacente.

tipo-datos-anclados

Identifica otro objeto que se utiliza para definir el tipo de datos. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican a la especificación del tipo de datos directamente o, en el caso de una fila, a la creación de un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable global se utiliza como tipo de datos para el *nombre-parámetro*.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el *nombre-parámetro*.

ROW OF *nombre-tabla* **o** *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. El tipo de datos del *nombre-parámetro* es un tipo de fila sin nombre.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por

CREATE PROCEDURE (SQL)

nombre-variable-cursor. La variable de cursor especificada debe ser uno de los elementos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula *CONSTANT* especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable de cursor no es de un tipo firme que utiliza un tipo de fila con nombre, el tipo de datos de *nombre-parámetro* es un tipo de fila sin nombre.

nombre-tipo-matriz

Especifica el nombre de un tipo de matriz definido por el usuario. Si se especifica el *nombre-tipo-matriz* sin un nombre de esquema, el tipo de matriz se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-cursor

Especifica el nombre de un tipo de cursor. Si se especifica el *nombre-tipo-cursor* sin un nombre de esquema, el tipo de cursor se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. La longitud, precisión y la escala del parámetro son, respectivamente, la longitud, la precisión y la escala del tipo de fuente del tipo diferenciado. Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. Los campos de cada parámetro son los campos del tipo de fila. Si se especifica el *nombre-tipo-fila* sin un nombre de esquema, el tipo de fila se resuelve buscando en los esquemas de la vía de acceso de SQL.

DEFAULT

Especifica un valor por omisión para el parámetro. El valor por omisión puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. Los registros especiales que se pueden especificar como el valor por omisión son los mismos que se pueden especificar para un valor por omisión de columna (véase *cláusula-por-omisión* en la sentencia CREATE TABLE). Se pueden especificar otros registros especiales como valor por omisión utilizando una expresión.

La *expresión* puede ser cualquier expresión del tipo descrito en "Expresiones". Si no se especifica un valor por omisión, el parámetro no tendrá ningún valor por omisión y no se podrá omitir el argumento correspondiente al invocar el procedimiento. El tamaño máximo de la *expresión* es de 64 K bytes.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL o SQLSTATE 429BL). La expresión debe ser compatible con asignaciones con el tipo de datos del parámetro (SQLSTATE 42821).

No se puede especificar un valor por omisión en las situaciones siguientes:

- Para parámetros INOUT o OUT (SQLSTATE 42601)

- Para un parámetro del tipo ARRAY, ROW o CURSOR (SQLSTATE 429BB)

SPECIFIC *nombre-específico*

Proporciona un nombre exclusivo para la instancia del procedimiento que se está definiendo. Este nombre específico puede utilizarse al modificar, descartar o realizar un comentario en el procedimiento. No puede utilizarse nunca para invocar el procedimiento. El formato no calificado de un *nombre-específico* es un identificador SQL. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador de SQL. El nombre (incluido el calificador implícito o explícito) no debe designar 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 igual a un *nombre-procedimiento* existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, éste deberá ser el mismo que el calificador explícito o implícito del *nombre-procedimiento* o se generará un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo será 'SQL' seguido de una indicación de fecha y hora expresada en forma de caracteres: 'SQLaammdhmmssxxx'.

DYNAMIC RESULT SETS *entero*

Indica el límite superior estimado de los conjuntos de resultados devueltos para el procedimiento.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

Indica el nivel de acceso a datos para las sentencias de SQL que se han incluido en el procedimiento.

CONTAINS SQL

Indica que el procedimiento puede ejecutar sentencias de SQL que no lean ni modifiquen datos de SQL (SQLSTATE 38004 ó 42985). Es posible que las sentencias que no reciban soporte en los procedimientos devuelvan otro error (SQLSTATE 38003 o 42985).

READS SQL DATA

Indica que en el procedimiento pueden incluirse algunas sentencias de SQL que no modifiquen datos de SQL (SQLSTATE 38002 ó 42985). Es posible que las sentencias que no reciban soporte en los procedimientos devuelvan otro error (SQLSTATE 38003 o 42985).

MODIFIES SQL DATA

Indica que el procedimiento puede ejecutar cualquier sentencia de SQL excepto las que no están soportadas en ningún procedimiento (SQLSTATE 38003 ó 42985).

Si se utiliza la cláusula BEGIN ATOMIC en un procedimiento de SQL compuesto, sólo se podrá crear el procedimiento si está definido como MODIFIES SQL DATA.

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.

CREATE PROCEDURE (SQL)

Actualmente, esta cláusula no afecta al proceso del procedimiento.

CALLED ON NULL INPUT

CALLED ON NULL INPUT siempre se aplica a los procedimientos. Esto significa que el procedimiento recibe llamada independientemente de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento comprobar si hay valores de argumento nulos.

COMMIT ON RETURN

Indica si debe efectuarse una confirmación cuando el procedimiento vuelve. El valor por omisión es NO.

NO No se puede realizar una confirmación cuando vuelve el procedimiento.

YES

Se efectúa una confirmación cuando el procedimiento vuelve si la sentencia CALL devuelve un SQLCODE positivo

La operación de confirmación incluye el trabajo que efectúa el proceso de aplicación de llamada y el procedimiento.

Si el procedimiento devuelve conjuntos de resultados, los cursores asociados a dichos conjuntos deben haberse definido como WITH HOLD para poderlos utilizar después de la confirmación.

AUTONOMOUS

Indica que el procedimiento debe ejecutarse en su propio ámbito de transacción autónomo.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

OLD SAVEPOINT LEVEL o NEW SAVEPOINT LEVEL

Especifica si este procedimiento establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información sobre los niveles de punto de salvaguarda, consulte "Normas" en "SAVEPOINT".

LANGUAGE SQL

Esta cláusula se utiliza para especificar que el cuerpo de procedimiento se ha escrito en el lenguaje SQL.

EXTERNAL ACTION o NO EXTERNAL ACTION

Especifica si el procedimiento realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). El valor por omisión es EXTERNAL ACTION. Si

se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones que suponen que el procedimiento no tiene ningún impacto externo.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

cuerpo-procedimiento-SQL

Especifica la sentencia de SQL que forma el cuerpo del procedimiento de SQL.

Consulte *sentencia-procedimiento-SQL* en la sentencia de “SQL compuesto (compilado)”.

Normas

- **Restricciones de la rutina autónoma:** Las rutinas autónomas no pueden devolver conjuntos de resultados y no dan soporte a los tipos de datos siguientes (SQLSTATE 428H2):
 - Tipos de cursor definidos por el usuario
 - Tipos estructurados definidos por el usuario
 - XML como parámetros IN, OUT e INOUT

No se puede hacer referencia a variables de sesión de tipos de cursor dentro del ámbito autónomo.

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.
- **Uso de tipos de fila y cursor:** un procedimiento que utiliza un tipo de cursor o un tipo de fila para un parámetro solamente se puede invocar desde dentro de una sentencia de SQL compuesto (compilado) (SQLSTATE 428H2), excepto JDBC, que puede invocar un procedimiento con parámetros OUT que tienen un tipo de cursor.

Notas

- La creación de un procedimiento 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- Un procedimiento que se llama desde una sentencia de SQL compuesto (en línea) se ejecutará como si se hubiera creado especificando NEW SAVEPOINT

CREATE PROCEDURE (SQL)

LEVEL, aunque se haya especificado OLD SAVEPOINT LEVEL o se haya tomado por omisión al crearse el procedimiento.

- **Creación de procedimientos que no son válidos inicialmente:** si un objeto al que se hace referencia en el cuerpo del procedimiento no existe o se ha marcado como no válido o el definidor no tiene temporalmente los privilegios para acceder al objeto y, si el parámetro de configuración de la base de datos **auto_reval** no se ha establecido en DISABLED, el procedimiento seguirá creándose satisfactoriamente. El procedimiento se marcará como no válido y se volverá a validar la próxima vez que se invoque.
- **Establecimiento del valor por omisión:** los parámetros de un procedimiento que se definen con un valor por omisión se establecen en su valor por omisión cuando se invoca el procedimiento, aunque sólo si no se suministra un valor para el argumento correspondiente o se especifica como DEFAULT, cuando se invoca el procedimiento.
- **Privilegios:** el definidor de un procedimiento siempre recibe el privilegio WITH GRANT OPTION de EXECUTE en el procedimiento, así como el derecho de descartar el procedimiento.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - RESULT SETS puede especificarse en lugar de DYNAMIC RESULT SETS.
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT.

La sintaxis siguiente se acepta como comportamiento por omisión:

- ASUTIME NO LIMIT
- NO COLLID
- STAY RESIDENT NO

Ejemplo

Crear un procedimiento de SQL que devuelva el salario medio de la plantilla de trabajadores. Se obtendrá un conjunto de resultados 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;
```


CREATE PROCEDURE (SQL)

```
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 ROLE

La sentencia CREATE ROLE define un rol en el servidor actual.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

►► CREATE ROLE *nombre-rol* ◀◀

Descripción

nombre-rol

Da nombre al rol. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar un rol existente en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' y no debe ser 'ACCESSCTRL', 'DATAACCESS', 'DBADM', 'NONE', 'NULL', 'PUBLIC', 'SECADM', 'SQLADM' o 'WLMADM' (SQLSTATE 42939).

Ejemplo

Cree un rol denominado DOCTOR.

```
CREATE ROLE DOCTOR
```

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 incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

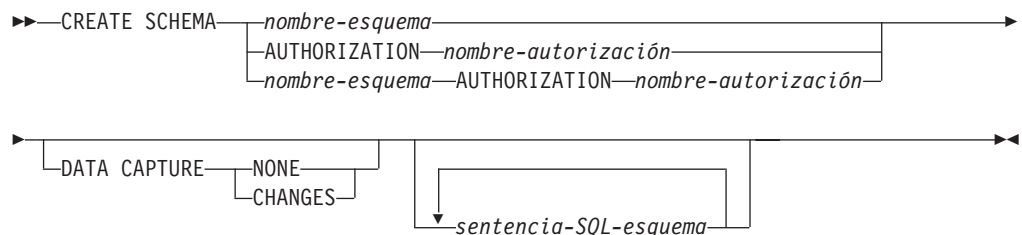
Un ID de autorización que tenga la autorización DBADM puede crear un esquema con cualquier *nombre-esquema* o *nombre-autorización* válido.

Un ID de autorización que no tenga la autorización DBADM sólo puede crear un esquema con un *nombre-esquema* o un *nombre-autorización* que coincida con el ID de autorización de la sentencia.

Si la sentencia incluye una *sentencia-SQL-esquema*, los privilegios del *nombre-autorización* (que, si no se especifica, toma por omisión el ID de autorización de la sentencia) deben incluir, como mínimo, una de las autorizaciones siguientes:

- Los privilegios necesarios para ejecutar cada *sentencia-SQL-esquema*
- Autorización 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 de *nombre-autorización* también se utiliza para denominar 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 denominado *nombre-esquema*, cuyo propietario es

CREATE SCHEMA

nombre-autorización. El *nombre-esquema* no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710). El *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

DATA CAPTURE

Indica si debe grabarse información adicional para la duplicación de datos en las anotaciones cronológicas. El valor por omisión se determina en función del valor del parámetro de configuración de la base de datos **dft_schemas_dcc**. Si el valor es "Sí", el valor por omisión es CHANGES; de lo contrario, el valor por omisión es NONE.

NONE

Indica que no se va a anotar cronológicamente ninguna información adicional de duplicación de datos.

CHANGES

Indica que en el archivo de anotaciones cronológicas se registrará información adicional referente a los cambios de SQL efectuados en este esquema. Esta opción es necesaria para duplicar este esquema y cuando se utiliza el programa de captura de duplicación para capturar los cambios contenidos en el archivo de anotaciones cronológicas para este esquema.

sentencia-SQL-esquema

Las sentencias de SQL que se pueden incluir como parte de la sentencia CREATE SCHEMA son:

- Sentencia CREATE TABLE, excluyendo las tablas con tipo y las tablas de consulta materializada
- Sentencia CREATE VIEW, excluyendo las vistas con tipo
- Sentencia CREATE INDEX
- Sentencia COMMENT
- Sentencia GRANT

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 otorgan al propietario del esquema los privilegios CREATEIN, DROPIN y ALTERIN para 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 (SQLSTATE 42875). Cualquier otro objeto que se haga referencia en las sentencias pueden calificarse con un nombre de esquema válido.

- 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 de SQL incorporado 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 o alias.
- Establecer el atributo DATA CAPTURE en el nivel de esquema hace que las tablas recién creadas hereden el atributo DATA CAPTURE del esquema si no se especifica uno en el nivel de tabla.

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 SECURITY LABEL COMPONENT

La sentencia CREATE SECURITY LABEL COMPONENT define un componente que se utilizará como parte de una política de seguridad.

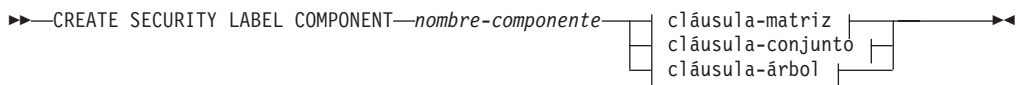
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



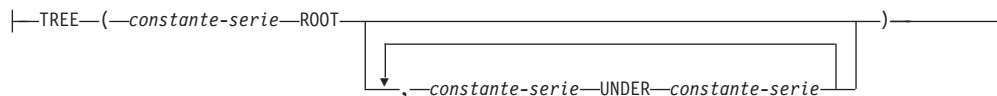
cláusula-matriz:



cláusula-conjunto:



cláusula-árbol:



Descripción

nombre-componente

Asigna un nombre al componente de etiqueta de seguridad. Este nombre consta de una sola parte. El nombre no debe identificar un componente de etiqueta de seguridad existente en el servidor actual (SQLSTATE 42710).

ARRAY

Especifica un conjunto ordenado de elementos.

constante-serie,...

Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad. Es importante el orden en que aparecen los elementos de la matriz. El primer

CREATE SECURITY LABEL COMPONENT

elemento tiene una clasificación superior al segundo. El segundo elemento tiene una clasificación superior al tercero, y así sucesivamente.

SET

Especifica un conjunto no ordenado de elementos.

constante-serie,...

Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad. El orden de los elementos no es importante.

TREE

Especifica una estructura de árbol de elementos de nodos.

constante-serie

Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad.

ROOT

Especifica que la *constante-serie* que sigue a la palabra clave es el elemento del nodo raíz del árbol.

UNDER

Especifica que la *constante-serie* antes de la palabra clave **UNDER** es hija de la *constante-serie* que sigue a la palabra clave **UNDER**. Se debe definir un elemento como elemento raíz o como el hijo de otro elemento antes de que se pueda utilizar como padre; de lo contrario, se devuelve un error (SQLSTATE 42704).

Normas

Estas normas se aplican a los tres tipos de componentes (ARRAY, SET y TREE):

- Los nombres de elementos no pueden contener ninguno de estos caracteres:
 - Paréntesis de apertura - (
 - Paréntesis de cierre -)
 - Coma - ,
 - Dos puntos - :
- Un nombre de elemento no puede tener más de 32 bytes (SQLSTATE 42622).
- Si un componente de etiqueta de seguridad es un conjunto o un árbol, no podrán formar parte de este componente más de 64 elementos.
- Una sentencia CREATE SECURITY LABEL COMPONENT puede especificar como máximo 65.535 para un componente de etiqueta de seguridad de tipo de matriz.
- Ningún nombre de elemento se puede utilizar más de una vez en el mismo componente (SQLSTATE 42713).

Ejemplos

- *Ejemplo 1:* Crear un componente de etiqueta de seguridad del tipo ARRAY llamado LEVEL. El componente tiene los cuatro elementos siguientes, enumerados en orden de clasificación decreciente: Top Secret, Secret, Classified y Unclassified.

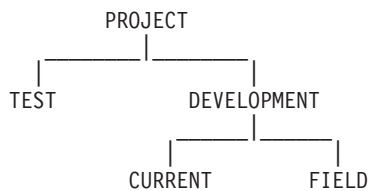
```
CREATE SECURITY LABEL COMPONENT LEVEL
  ARRAY ['Top Secret', 'Secret', 'Classified', 'Unclassified']
```

- *Ejemplo 2:* Crear un componente de etiqueta de seguridad del tipo SET llamado COMPARTMENTS. El componente tiene los tres elementos siguientes: Research, Analysis y Collection.

CREATE SECURITY LABEL COMPONENT

```
CREATE SECURITY LABEL COMPONENT COMPARTMENTS  
SET {'Collection', 'Research', 'Analysis'}
```

- *Ejemplo 3:* Crear un componente de etiqueta de seguridad del tipo TREE llamado GROUPS. GROUPS tiene cinco elementos: PROJECT, TEST, DEVELOPMENT, CURRENT y FIELD. El diagrama siguiente muestra la relación de estos elementos entre sí:



```
CREATE SECURITY LABEL COMPONENT GROUPS  
TREE (  
  'PROJECT' ROOT,  
  'TEST' UNDER 'PROJECT',  
  'DEVELOPMENT' UNDER 'PROJECT',  
  'CURRENT' UNDER 'DEVELOPMENT',  
  'FIELD' UNDER 'DEVELOPMENT'  
)
```


CREATE SECURITY LABEL

La sentencia CREATE SECURITY LABEL define una etiqueta de seguridad.

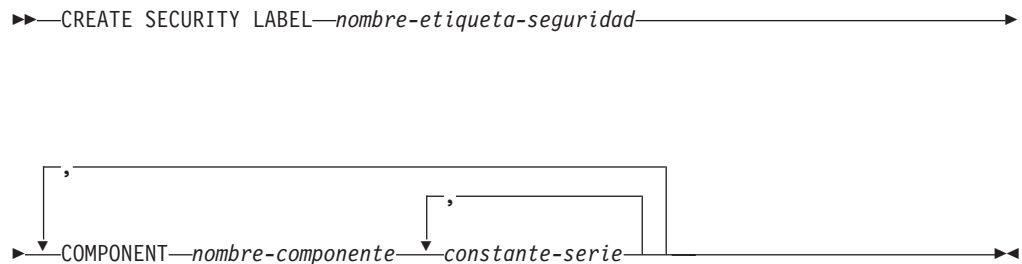
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Descripción

nombre-etiqueta-seguridad

Asigna un nombre a la etiqueta de seguridad. El nombre debe calificarse con una política de seguridad (SQLSTATE 42704) y no debe identificar una etiqueta de seguridad existente para esta política de seguridad (SQLSTATE 42710).

COMPONENT *nombre-componente*

Especifica el nombre de un componente de etiqueta de seguridad. Si el componente no forma parte de la política de seguridad *nombre-política-seguridad*, se devuelve un error (SQLSTATE 4274G). Si un componente se especifica dos veces en la misma sentencia, se devuelve un error (SQLSTATE 42713).

constante-serie,...

Especifica un elemento válido para el componente de seguridad. Un elemento válido es el que se especificó cuando se creó el componente de seguridad. Si el elemento no es válido, se devuelve un error (SQLSTATE 4274F).

Ejemplos

- *Ejemplo 1:* Crear una etiqueta de seguridad llamada EMPLOYEESECLABEL que forme parte de la política de seguridad DATA_ACCESS y que tenga el elemento Top Secret para el componente LEVEL, así como los elementos Research y Analysis para el componente COMPARTMENTS.

```

CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research', 'Analysis'
  
```

CREATE SECURITY LABEL

- *Ejemplo 2:* Crear una etiqueta de seguridad llamada EMPLOYEESECLABELREAD que tenga el elemento Top Secret para el componente LEVEL y el elemento Research para el componente COMPARTMENTS.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD  
COMPONENT LEVEL 'Top Secret',  
COMPONENT COMPARTMENTS 'Research'
```

- *Ejemplo 3:* Crear una etiqueta de seguridad llamada EMPLOYEESECLABELWRITE que tenga el elemento Analysis para el componente COMPARTMENTS y un valor nulo para el componente LEVEL. Suponer que la política de seguridad llamada DATA_ACCESS es la misma política de seguridad que la que se utiliza en los ejemplos 1 y 2.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE  
COMPONENT COMPARTMENTS 'Analysis'
```

- *Ejemplo 4:* Crear una etiqueta de seguridad llamada BEGINNER que forme parte de una política de seguridad CLASSPOLICY existente, y que tenga el elemento Trainee para el componente TRUST y el elemento Morning para el componente SECTIONS.

```
CREATE SECURITY LABEL CLASSPOLICY.BEGINNER  
COMPONENT TRUST 'Trainee',  
COMPONENT SECTIONS 'Morning'
```

CREATE SECURITY POLICY

La sentencia CREATE SECURITY POLICY define una política de seguridad.

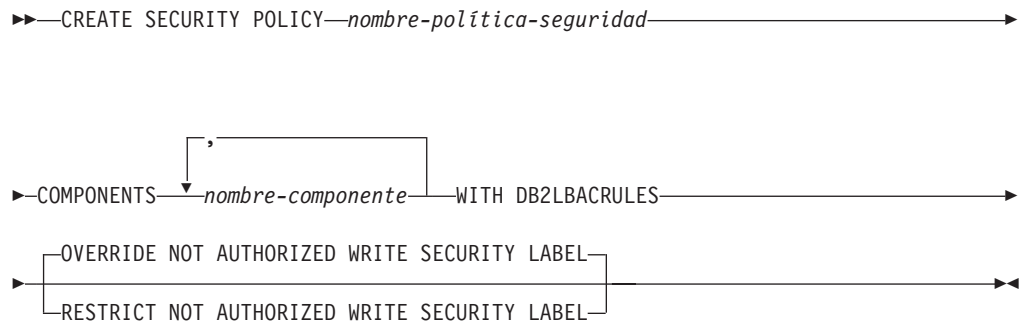
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Descripción

nombre-política-seguridad

Asigna un nombre a la política de seguridad. Este nombre consta de una sola parte. El nombre no debe identificar una política de seguridad existente en el servidor actual (SQLSTATE 42710).

COMPONENTS *nombre-componente*,...

Identifica un componente de etiqueta de seguridad. El nombre debe identificar un componente de etiqueta de seguridad que ya existe en el servidor actual (SQLSTATE 42704). No se debe especificar el mismo componente de seguridad más de una vez para la política de seguridad (SQLSTATE 42713). No se pueden especificar más de 16 componentes de etiqueta de seguridad para una política de seguridad (SQLSTATE 54062).

WITH DB2LBACRULES

Indica qué conjunto de normas se utilizará al comparar las etiquetas de seguridad que forman parte de esta política de seguridad. Actualmente sólo hay un conjunto de normas: DB2LBACRULES.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL o RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

Especifica la acción que se llevará a cabo cuando un usuario no tenga autorización para grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia INSERT o UPDATE emitida sobre una tabla que está protegida con esta política de seguridad. Una etiqueta de seguridad del usuario y las credenciales de exención determinan la autorización del

CREATE SECURITY POLICY

usuario para grabar una etiqueta de seguridad proporcionada explícitamente. El valor por omisión es `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

Indica que el valor de la etiqueta de seguridad del usuario se usará para el acceso de grabación durante una operación de inserción o de actualización, en lugar de la etiqueta de seguridad especificada explícitamente.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

Indica que la operación de inserción o actualización no será satisfactoria si el usuario no está autorizado a grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia `INSERT` o `UPDATE` (`SQLSTATE 42519`).

Notas

- *Conjunto de normas DB2LBACRULES*: `DB2LBACRULES` es un conjunto de normas predefinido que incluye las siguientes normas: `DB2LBACREADARRAY`, `DB2LBACREADSET`, `DB2LBACREADTREE`, `DB2LBACWRITEARRAY`, `DB2LBACWRITESET`, `DB2LBACWRITETREE`.
- Las autorizaciones de rol y de grupo no se tienen en cuenta por omisión cuando se crea una política de seguridad. Utilice la sentencia `ALTER SECURITY POLICY` para cambiar este comportamiento y tenerlas en cuenta.

Ejemplos

- *Ejemplo 1*: Crear una política de seguridad llamada `DATA_ACCESS` que utilice el conjunto de normas `DB2LBACRULES` y que tenga dos componentes: `LEVEL` y `COMPARTMENTS`, en este orden. Suponer que los dos componentes ya existen.

```
CREATE SECURITY POLICY DATA_ACCESS  
  COMPONENTS LEVEL, COMPARTMENTS  
  WITH DB2LBACRULES
```

- *Ejemplo 2*: Crear una política de seguridad llamada `CONTRIBUTIONS` que tenga los componentes `MEMBER` y `BADGE`, que se supone que ya existen.

```
CREATE SECURITY POLICY CONTRIBUTIONS  
  COMPONENTS MEMBER, BADGE  
  WITH DB2LBACRULES
```

CREATE SEQUENCE

La sentencia CREATE SEQUENCE define una secuencia en el servidor de aplicaciones.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

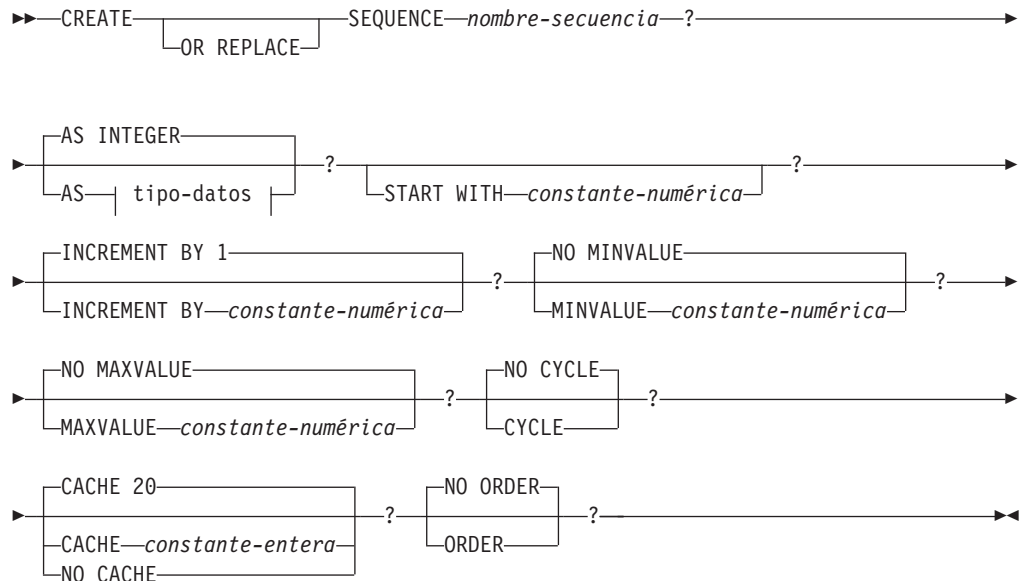
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la secuencia no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la secuencia hace referencia a un esquema existente
- Autorización DBADM

Para sustituir una secuencia existente, el ID de autorización de la sentencia debe ser el propietario de la secuencia existente (SQLSTATE 42501).

Sintaxis

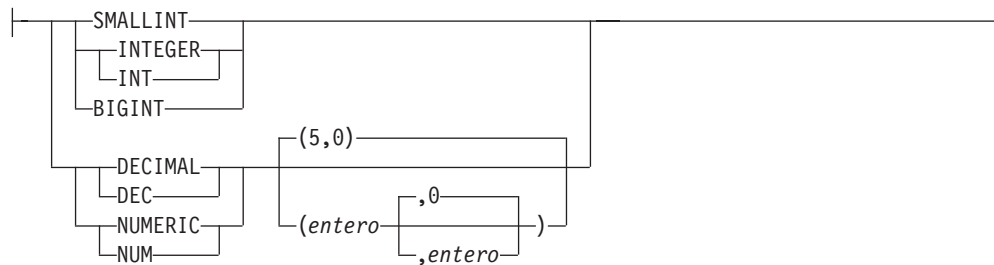


tipo-datos:



CREATE SEQUENCE

tipo-incorporado:



Notas:

- 1 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados.

Descripción

OR REPLACE

Especifica que se debe sustituir la definición por la secuencia si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre la secuencia no se ven afectados por ello. No se tiene en cuenta esta opción si no existe una definición para la secuencia en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

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 de fuente sea un tipo numérico exacto con una escala de cero (SQLSTATE 42815). El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2). 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 a la secuencia (SQLSTATE 42815), sin 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 a la secuencia (SQLSTATE 42815). El valor no debe superar el valor de una constante de enteros grande (SQLSTATE 42820) ni contener dígitos que no sean cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente. Si este valor es 0 o positivo, es 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 una columna del tipo de datos asociado a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser inferior o igual al valor máximo (SQLSTATE 42815).

NO MINVALUE

Para una secuencia ascendente, el valor es el valor **START WITH** o bien 1 si no se ha especificado **START WITH**. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos asociado con la secuencia. 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 a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser 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 **START WITH** o bien -1 si no se ha especificado **START WITH**.

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,

CREATE SEQUENCE

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. 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 reducen 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.

En un entorno DB2 pureScale o de varias particiones, utilice las opciones CACHE y NO ORDER para que varios miembros de DB2 puedan poner en antememoria valores de secuencia de forma simultánea.

En un entorno DB2 pureScale, si se especifican tanto CACHE como ORDER, la especificación de ORDER altera temporalmente la especificación de CACHE y, en su lugar, NO CACHE entrará en vigor.

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. Es el valor por omisión.

Notas

- Es posible definir una secuencia constante, es decir, una que devuelva siempre un valor constante. Esto puede hacerse especificando un valor cero para INCREMENT y un valor para START WITH que no exceda el valor de MAXVALUE o bien especificando el mismo valor para START WITH, MINVALUE y MAXVALUE. Para una secuencia constante, cada vez que se invoque NEXT VALUE para la secuencia, se devolverá el mismo valor. Una secuencia constante puede utilizarse como variable global numérica. ALTER SEQUENCE puede utilizarse para ajustar los valores que se generarán para una secuencia constante.
- El ciclo de una secuencia puede especificarse 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.
- Una secuencia puede definirse explícitamente para que ejecute un ciclo especificando la palabra clave CYCLE. Al definir una secuencia, utilice la opción CYCLE para indicar que los valores generados deben ejecutar un ciclo cuando se haya alcanzado el límite. Cuando se ha definido una secuencia para la ejecución automática de un ciclo (es decir, CYCLE se ha especificado explícitamente), el valor máximo o mínimo generado para una secuencia puede que no sea el valor MAXVALUE o MINVALUE real que se ha especificado, si el incremento es un valor distinto de 1 ó -1. Por ejemplo, la secuencia que se ha definido con START WITH=1, INCREMENT=2, MAXVALUE=10 generará un valor máximo de 9 y no generará el valor 10. Al definir una secuencia con CYCLE, considere detenidamente el efecto que pueden tener los valores para MINVALUE, MAXVALUE y START WITH.
- 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.
- Al usuario que define una secuencia se le otorgan los privilegios ALTER y USAGE y el privilegio WITH GRANT OPTION. El propietario de la secuencia puede descartar la secuencia.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede utilizarse una coma para separar varias opciones en la secuencia.
 - Se puede especificar NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER en lugar de NO MINVALUE, NO MAXVALUE, NO CYCLE, NO CACHE y NO ORDER, respectivamente
- **Consideraciones sobre un entorno DB2 pureScale** o de varias particiones:
 - En un entorno DB2 pureScale o de varias particiones, si las opciones CACHE y NO ORDER están en vigor, se pueden activar varias antememorias a la vez y puede que las solicitudes de asignaciones de valor siguiente de los distintos miembros no asignen valores en un orden numérico estricto. Supongamos,

CREATE SEQUENCE

por ejemplo, que los miembros DB2A y DB2B están utilizando la misma secuencia y que DB2A obtiene los valores de antememoria del 1 al 20 y DB2B obtiene los valores de antememoria del 21 al 40. En este caso, si DB2A ha solicitado primero el siguiente valor y, a continuación, lo ha solicitado DB2B y, entonces, DB2A lo ha vuelto a solicitar, el orden real de valores asignado sería 1,21,2. Por lo tanto, para garantizar que los números de secuencia se generen en un orden numérico estricto entre varios miembros que utilicen la misma secuencia de forma simultánea, especifique la opción ORDER.

- En un entorno DB2 pureScale, el uso de la opción ORDER o NO CACHE garantiza que los valores asignados a una secuencia que comparten aplicaciones en varios miembros sigan un orden numérico estricto. Si se especifica ORDER, la opción NO CACHE queda implícita aunque se especifique CACHE *n*.

Ejemplo

Crear una secuencia denominada ORG_SEQ que empiece por el 1, que se ejecute en incrementos de 1, que no ejecute un ciclo y que coloque en antememoria 24 valores al mismo tiempo:

```
CREATE SEQUENCE ORG_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

CREATE SERVICE CLASS

La sentencia CREATE SERVICE CLASS define una clase de servicio.

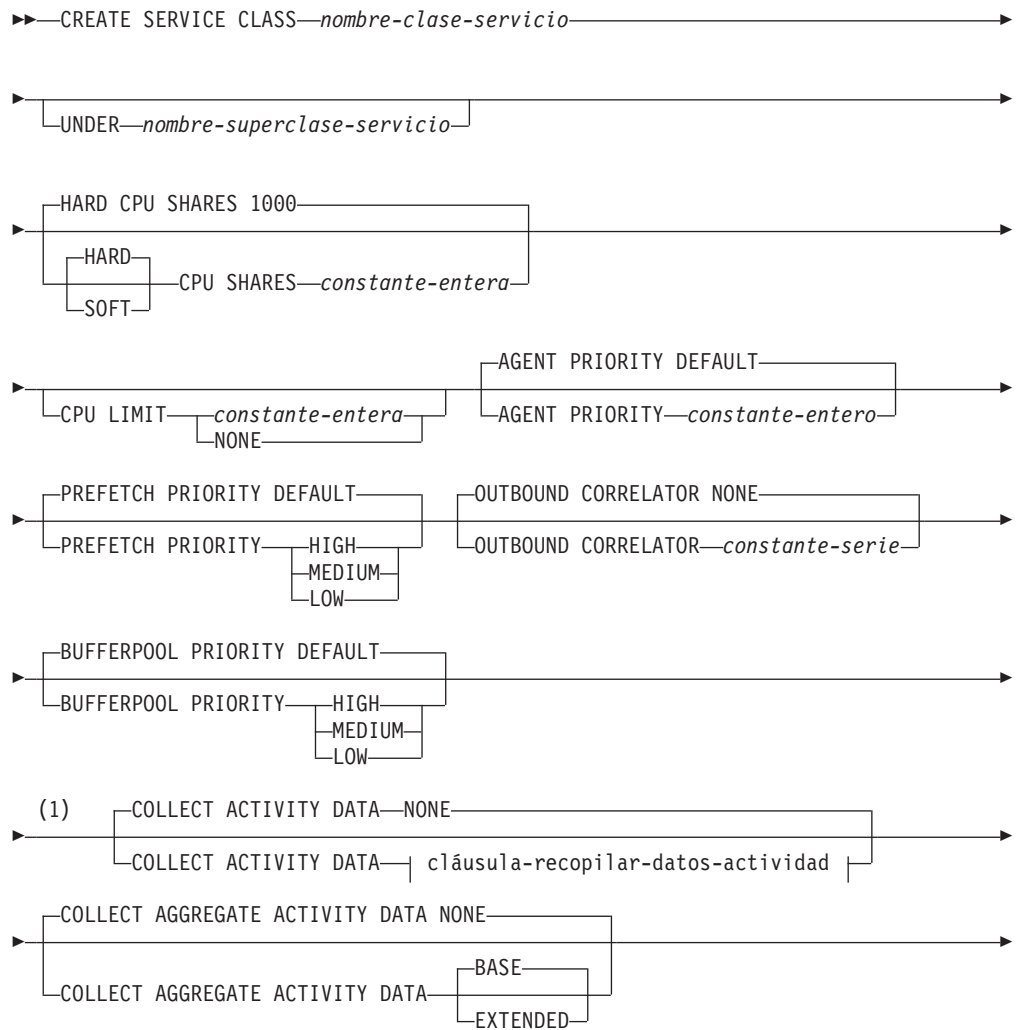
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

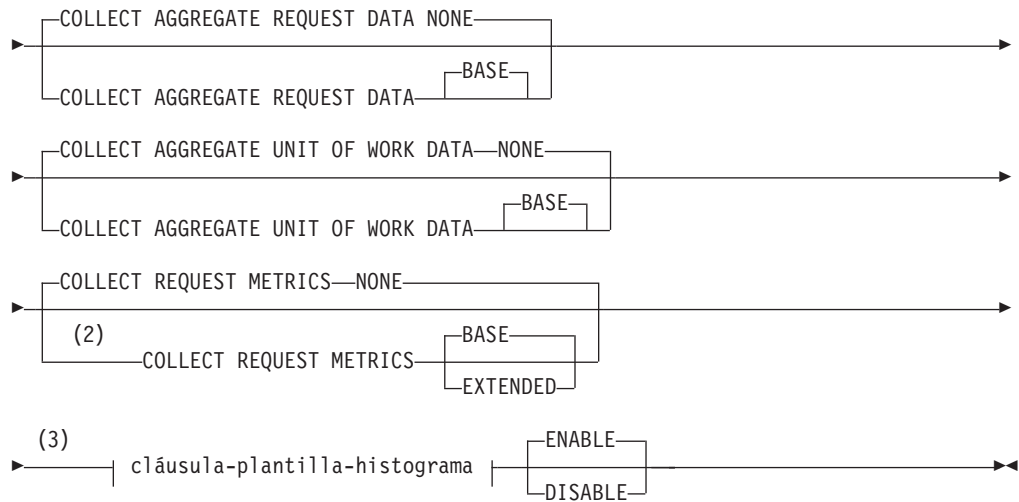
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

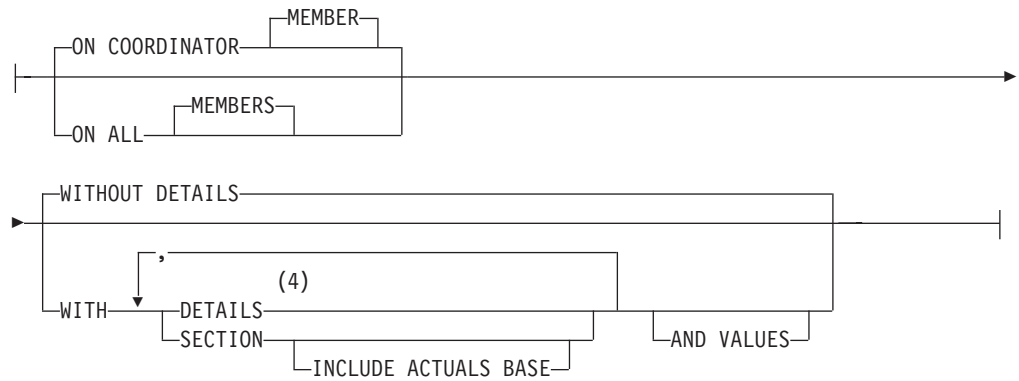
Sintaxis



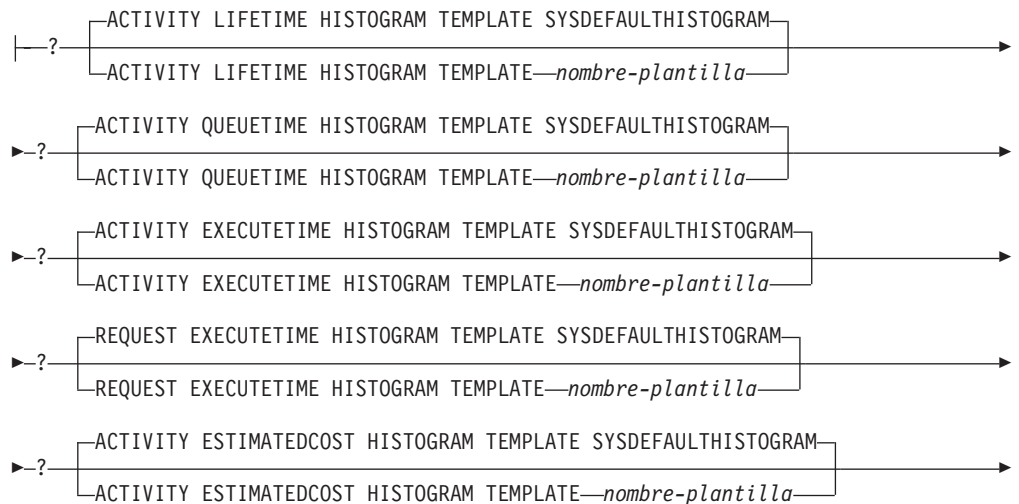
CREATE SERVICE CLASS

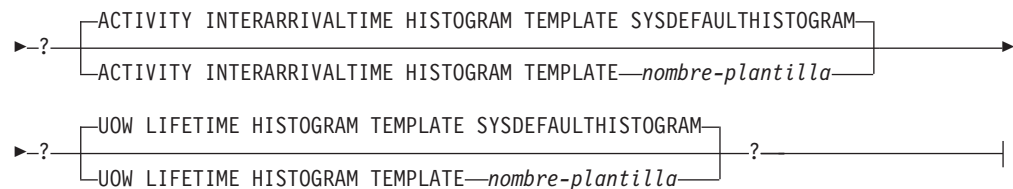


cláusula-recopilar-datos-actividad:



cláusula-plantilla-histograma:





Notas:

- 1 Todas las cláusulas COLLECT excepto COLLECT REQUEST METRICS sólo son válidas para una subclase de servicio.
- 2 La cláusula COLLECT REQUEST METRICS sólo es válida para una superclase de servicio.
- 3 Las cláusulas HISTOGRAM TEMPLATE sólo son válidas para una subclase de servicio.
- 4 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-clase-servicio

Indica el nombre de la clase de servicio. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). Si la clase de servicio es una superclase de servicio, el *nombre-clase-servicio* no debe identificar ninguna superclase de servicio que ya exista en el catálogo (SQLSTATE 42710). Si la clase de servicio es una subclase de servicio, el *nombre-clase-servicio* no debe identificar ninguna subclase de servicio que ya exista bajo la superclase de servicio (SQLSTATE 42710). Si la clase de servicio es una subclase de servicio, el *nombre-clase-servicio* no debe ser el mismo que su superclase de servicio (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

UNDER *nombre-superclase-servicio*

Especifica que la clase de servicio es una subclase de la superclase de servicio *nombre-superclase-servicio*. Si no se especifica UNDER, la clase de servicio es una superclase de servicio. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista para la base de datos (SQLSTATE 42704). La superclase de servicio no puede ser una clase de servicio por omisión (SQLSTATE 5U029).

HARD CPU SHARES 1000 o **HARD CPU SHARES** *constante-entera* o **SOFT CPU SHARES** *constante-entera*

Especifica el número de cuotas de recursos de CPU que el asignador de WLM asigna a esta clase de servicio cuando el trabajo se ejecuta dentro de esta clase de servicio. Los valores válidos de la *constante-entera* son enteros entre 1 y 65535. El valor por omisión es **HARD CPU SHARES 1000**, que asigna 1000 cuotas fijas a esta clase de servicio. Calificar **CPU SHARES** con la palabra clave **HARD** o especificar **CPU SHARES** sin calificarlo con la palabra clave **HARD** o **SOFT** indica que se asignarán cuotas de CPU fijas a esta clase de servicio. Especificar la palabra clave **SOFT** indica que se asignarán cuotas de CPU flexibles a esta clase de servicio. Para utilizar cuotas de CPU fijas y flexibles con el asignador del gestor de carga de trabajo de DB2, debe habilitar el parámetro de configuración del gestor de bases de datos **wlm_disp_cpu_shares**.

CPU LIMIT *constante-entero* o **CPU LIMIT NONE**

Especifica el porcentaje máximo de los recursos de CPU que el asignador de WLM puede asignar a esta clase de servicio. Los valores válidos de la

CREATE SERVICE CLASS

constante-entera son enteros entre 1 y 100. Puede especificar también **CPU LIMIT NONE** para indicar que no hay ningún límite de CPU.

AGENT PRIORITY DEFAULT o **AGENT PRIORITY** *constante-entero*

Especifica la prioridad del sistema operativo (delta) relativa de los agentes que se ejecutan en la clase de servicio o la prioridad normal de las hebras que se ejecutan en DB2. El valor por omisión es DEFAULT.

Importante: El parámetro de configuración del gestor de bases de datos **agentpri** está en desuso desde la Versión 9.5. Todavía se puede utilizar en los clientes y servidores de datos anteriores a la Versión 9.5. Además, la prioridad de agente para la clase de servicio de WLM ha quedado en desuso en la Versión 10.1 y podría eliminarse en un release futuro. Comience a utilizar la función de asignador de WLM en lugar de la prioridad de agente. Para obtener más información, consulte "Prioridad de agente de clases de servicio ha quedado en desuso" en *Novedades en DB2 Versión 10.1*.

Cuando se establece en DEFAULT, no se adopta ninguna acción especial y los agentes de la clase de servicio se planifican con arreglo a la prioridad normal con la que el sistema operativo planifica todas las hebras de DB2. Cuando se establece este parámetro en un valor que no sea DEFAULT, los agentes se establecen en una prioridad que sea igual a la prioridad normal más AGENT PRIORITY cuando comienza la siguiente actividad. Por ejemplo, si la prioridad normal es 20 y AGENT PRIORITY se establece en -10, la prioridad de los agentes de la clase de servicio se establece en $20 - 10 = 10$.

Nota: No se pueden utilizar conjuntamente comparticiones de prioridad de agente y asignador de WLM. Cuando el asignador se habilita estableciendo el valor del parámetro de configuración de gestor de bases de datos **wlm_dispatcher** en ON, la prioridad de agente especificada se ignora y la prioridad de agente se establece en el valor por omisión hasta que el asignador está inhabilitado.

El gestor de carga de trabajo de DB2 (WLM) no asigna prioridades de agente de clase de servicio a tareas que se realicen dentro de un proceso en protegido (FMP). Los procedimientos delimitados no ejecutan la lógica dentro de una clase de servicio. Los procedimientos delimitados se ejecutan dentro de DB2 FMP y los agentes de DB2 no realizan estas tareas. A modo de recordatorio, DB2 WLM controla los agentes de DB2.

En sistemas operativos UNIX y Linux, los valores válidos son DEFAULT y de -20 a 20 (SQLSTATE 42615). Los valores negativos denotan una prioridad relativa más alta. Los valores positivos denotan una prioridad relativa más baja.

En sistemas operativos Windows, los valores válidos son DEFAULT y de -6 a 6 (SQLSTATE 42615). Los valores negativos denotan una prioridad relativa más baja. Los valores positivos denotan una prioridad relativa más alta.

Si AGENT PRIORITY es DEFAULT para una subclase de servicio, hereda el valor de AGENT PRIORITY de su superclase padre. AGENT PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032). AGENT PRIORITY debe establecerse en DEFAULT si se establece OUTBOUND CORRELATOR (SQLSTATE 42613).

Nota: En AIX, el propietario de instancia debe tener posibilidades CAP_NUMA_ATTACH y CAP_PROPAGATE para establecer una prioridad

relativa más alta para los agentes en una clase de servicio utilizando AGENT PRIORITY. Para otorgar estas posibilidades, inicie la sesión como root y ejecute el mandato siguiente:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

En Solaris 10 o posterior, el propietario de instancia debe tener el privilegio proc_priocntl para establecer una prioridad relativa más alta para los agentes en una clase de servicio con AGENT PRIORITY. Para otorgar este privilegio, inicie la sesión como root y ejecute el siguiente mandato:

```
usermod -K defaultpriv=basic,proc_priocntl db2user
```

En este ejemplo, se añade proc_priocntl al conjunto de privilegios por omisión del usuario db2user.

Asimismo, cuando DB2 se está ejecutando en una zona de Solaris no global, el privilegio proc_priocntl debe añadirse al conjunto de privilegios de límite de la zona. Para otorgar este privilegio a la zona, inicie la sesión como root y ejecute el siguiente mandato:

```
global# zonecfg -z db2zone
zonecfg:db2zone> set limitpriv="default,proc_priocntl"
```

En este ejemplo, proc_priocntl se añade al conjunto de privilegios de límite de la zona db2zone.

En Solaris 9, no existe ningún recurso para DB2 para incrementar la prioridad relativa de los agentes. Actualice a Solaris 10 o posterior para utilizar la prioridad de agente de la clase de servicio.

PREFETCH PRIORITY DEFAULT | HIGH | MEDIUM | LOW

Este parámetro controla la prioridad con la que los agentes de la clase de servicio pueden someter sus peticiones de captación previa. Los valores válidos son HIGH, MEDIUM, LOW o DEFAULT (SQLSTATE 42615). HIGH, MEDIUM y LOW significan que las peticiones de captación previa se someterán a las colas de prioridad alta, media y baja, respectivamente. Los captadores previos vacían la cola de prioridad de la alta a la baja. Los agentes de la clase de servicio envían sus peticiones de captación previa al nivel de PREFETCH PRIORITY cuando comienza la siguiente actividad. Si se altera PREFETCH PRIORITY después de someter una petición de captación previa, no se cambiará la prioridad de petición. El valor por omisión es DEFAULT, que se correlaciona internamente con MEDIUM para las superclases de servicio. Si se especifica DEFAULT para una subclase de servicio, hereda el valor de PREFETCH PRIORITY o de su superclase padre.

PREFETCH PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032).

OUTBOUND CORRELATOR NONE o OUTBOUND CORRELATOR *constante-serie*

Especifica si asociar o no hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa.

Si se establece OUTBOUND CORRELATOR en *constante-serie* para la superclase de servicio y se establece OUTBOUND CORRELATOR NONE para una subclase de servicio, la subclase de servicio hereda el OUTBOUND CORRELATOR de su padre. OUTBOUND CORRELATOR debe establecerse en NONE si AGENT PRIORITY no se establece en DEFAULT (SQLSTATE 42613). El valor por omisión es OUTBOUND CORRELATOR NONE.

OUTBOUND CORRELATOR NONE

Para una superclase de servicio, especifica que no hay ninguna asociación de clase de servicio de gestor de carga de trabajo externa con esta clase de

CREATE SERVICE CLASS

servicio y, para una subclase de servicio, especifica que la asociación de clase de servicio de gestor de carga de trabajo externa es la misma que la de su padre.

OUTBOUND CORRELATOR *constante-serie*

Especifica la *constante-serie* que ha de utilizarse como correlacionador para asociar hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa. El gestor de carga de trabajo externa debe estar activa (SQLSTATE 5U030). El gestor de carga de trabajo externa debería configurarse para reconocer el valor de *constante-serie*.

BUFFERPOOL PRIORITY DEFAULT | HIGH | MEDIUM | LOW

Este parámetro controla la prioridad de agrupación de almacenamientos intermedios de páginas captadas por actividades de esta clase de servicio. Los valores válidos son HIGH, MEDIUM, LOW o DEFAULT (SQLSTATE 42615). La probabilidad de intercambio de las páginas captadas por actividades de una clase de servicio con una prioridad de agrupación de almacenamientos intermedios más alta es menor que la captura de páginas mediante actividades en una clase de servicio con una prioridad de agrupación de almacenamientos intermedios más baja. El valor por omisión es DEFAULT, que se correlaciona internamente con LOW para la superclase de servicio. Si se especifica DEFAULT para una subclase de servicio, hereda el valor de BUFFERPOOL PRIORITY de su superclase padre.

BUFFERPOOL PRIORITY no puede alterarse para una subclase por omisión (SQLSTATE 5U032).

COLLECT ACTIVITY DATA

Especifica que la información relacionada con cada actividad que se ejecuta en esta clase de servicio ha de enviarse a cualquier supervisor de sucesos de actividades activo cuando se haya completado la actividad. El valor por omisión es COLLECT ACTIVITY DATA NONE. La cláusula COLLECT ACTIVITY DATA sólo es válida para una subclase de servicio.

NONE

Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que se ejecutan en esta clase de servicio.

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad que se ejecuta en la clase de servicio deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la ejecución completa la actividad. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación

deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar `DETAILS` si se especifica `SECTION`. Los datos reales de sección sólo se recopilarán en cualquier miembro donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula `INCLUDE ACTUALS` o se debe establecer el parámetro de configuración de base de datos `section_actuals`.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula `INCLUDE ACTUALS`, el parámetro de configuración de base de datos `section_actuals` y el valor `<collectsectionactuals>` especificado en la rutina `WLM_SET_CONN_ENV`. Por ejemplo, si se especifica `INCLUDE ACTUALS BASE`, pero el valor del parámetro de configuración de la base de datos `section_actuals` es `NONE` y `<collectsectionactuals>` se establece en `NONE`, el valor efectivo de la colección de datos reales de la sección es `BASE`.

`BASE` especifica que lo siguiente debe habilitarse y recopilarse durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividades agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos `wlm_collect_int`. Cuando no se especifica `COLLECT AGGREGATE ACTIVITY DATA`, el valor por omisión es `COLLECT AGGREGATE ACTIVITY DATA NONE`. Cuando se especifica `COLLECT AGGREGATE ACTIVITY DATA`, el valor por omisión es `COLLECT AGGREGATE ACTIVITY DATA BASE`. La cláusula `COLLECT AGGREGATE ACTIVITY DATA` sólo es válida para una subclase de servicio.

BASE

Especifica que los datos de actividades agregados básicos deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal

CREATE SERVICE CLASS

Nota: Sólo las actividades a las que se ha aplicado un umbral SQLTEMPSPACE participan en esta marca de límite superior.

- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

NONE

Especifica que no debe capturarse ningún dato de actividad agregados para esta clase de servicio.

COLLECT AGGREGATE REQUEST DATA

Especifica que los datos de peticiones agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo especificado por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE REQUEST DATA NONE. La cláusula COLLECT AGGREGATE REQUEST DATA sólo es válida para una subclase de servicio.

BASE

Especifica que los datos de peticiones agregados básicos deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo.

NONE

Especifica que no debe capturarse ningún dato de petición agregado para esta clase de servicio.

COLLECT AGGREGATE UNIT OF WORK DATA

Especifica que deben capturarse los datos agregados de unidad de trabajo para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo especificado por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE UNIT OF WORK DATA NONE. La cláusula COLLECT AGGREGATE UNIT OF WORK DATA sólo es válida para una subclase de servicio.

BASE

Especifica que deben capturarse los datos agregados básicos de unidad de trabajo para esta clase de servicio y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos agregados básicos de unidad de trabajo incluyen:

- Unidad de histograma de ciclo de vida de trabajo

NONE

Especifica que no debe recopilarse ningún dato agregado de unidad de trabajo para esta clase de servicio.

COLLECT REQUEST METRICS

Especifica que la métrica del supervisor debe recopilarse para cualquier

petición enviada por una conexión asociada con la superclase de servicio especificada y enviarse a los supervisores de sucesos estadísticos y de unidad de trabajo, si están activos. El valor por omisión es COLLECT REQUEST METRICS NONE. La cláusula COLLECT REQUEST METRICS sólo es válida para una superclase de servicio (SQLSTATE 50U44).

Nota: La configuración de la recopilación de métricas de petición efectiva es la combinación del atributo que se especifica mediante la cláusula COLLECT REQUEST METRICS en la superclase de servicio asociada a la conexión que envía la petición y del parámetro de configuración de base de datos **mon_req_metrics**. Si el atributo de superclase de servicio o bien el parámetro de configuración tiene un valor distinto de NONE, se recopilarán métricas para la petición.

NONE

Especifica que no se recopilará ninguna métrica para las peticiones enviadas por una conexión asociada con la superclase de servicio.

BASE

Especifica que la métrica básica se recopilará para cualquier petición enviada por una conexión asociada con la superclase de servicio.

EXTENDED

Especifica que deben capturarse datos de petición agregados básicos para esta clase de servicio y enviar éstos al supervisor de sucesos estadísticos, si existe uno activo. Asimismo, especifica que los valores de los siguientes elementos de supervisor deben determinarse con granularidad adicional:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

cláusula-plantilla-histograma

Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades que se ejecutan en la clase de servicio. La cláusula HISTOGRAM TEMPLATE sólo es válida para una subclase de servicio.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos de actividades de DB2 en ejecución en la clase de servicio durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la clase de servicio están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

CREATE SERVICE CLASS

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en el miembro coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las peticiones de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la petición se recopila en este histograma en cada miembro donde se ejecuta la petición. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE REQUEST DATA con la opción BASE.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML en ejecución en la clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

UOW LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma que se utiliza para recopilar datos estadísticos sobre la duración, en milisegundos, de unidades de trabajo que se ejecutan en la clase de servicio durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE UNIT OF WORK DATA con la opción BASE.

ENABLE o DISABLE

Especifica si pueden correlacionarse o no conexiones y actividades con la clase de servicio. El valor por omisión es ENABLE.

ENABLE

Las conexiones y actividades pueden correlacionarse con la clase de servicio.

DISABLE

Las conexiones y actividades no pueden correlacionarse con la clase de servicio. Las conexiones o actividades correlacionadas a una clase de servicio inhabilitada se rechazarán (SQLSTATE 5U028). Cuando se inhabilita una superclase de servicio, también se inhabilitarán sus subclases de servicio. Cuando vuelve a habilitarse la superclase de servicio, sus subclases de servicio vuelven a los estados definidos en el catálogo del sistema. Una clase de servicio por omisión no puede inhabilitarse (SQLSTATE 5U032).

Normas

- El número máximo de subclases de servicio que pueden crearse bajo una superclase de servicio es 61 (SQLSTATE 5U027).
- El número máximo de superclases de servicio que pueden crearse para una base de datos es 64 (SQLSTATE 5U027).
- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U027). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (plantilla de histograma)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (clase de servicio)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (umbral)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (conjunto de acciones de trabajo)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (conjunto de clases de trabajo)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (carga de trabajo)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Una subclase por omisión, SYSDEFAULTSUBCLASS, se crea automáticamente para cada superclase de servicio.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todos los miembros. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.
- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

CREATE SERVICE CLASS

- Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* Cree una superclase de servicio denominada PETSALLES. La subclase por omisión para PETSALLES se crea automáticamente.

```
CREATE SERVICE CLASS PETSALLES
```

- *Ejemplo 2:* Cree una subclase de servicio denominada DOGSALLES debajo de la superclase de servicio PETSALLES. Establezca la clase de servicio DOGSALLES como inhabilitada.

```
CREATE SERVICE CLASS DOGSALLES UNDER PETSALLES DISABLE
```

- *Ejemplo 3:* Cree una superclase de servicio denominada BARNSALLES con una prioridad de captación previa de LOW. La subclase por omisión para BARNSALLES se crea automáticamente. Las peticiones de captación previa que envíen los agentes de la clase de servicio BARNSALLES irán a la cola de captación previa de prioridad baja.

```
CREATE  
SERVICE CLASS BARNSALLES PREFETCH PRIORITY LOW
```

CREATE SERVER

La sentencia CREATE SERVER define una fuente de datos para una base de datos federada.

En esta sentencia, el término SERVER y los nombres de parámetro que terminan por *-servidor* sólo hacen referencia a fuentes de datos de un sistema federado. No hacen referencia al servidor federado de ese sistema ni a los servidores de aplicaciones DRDA.

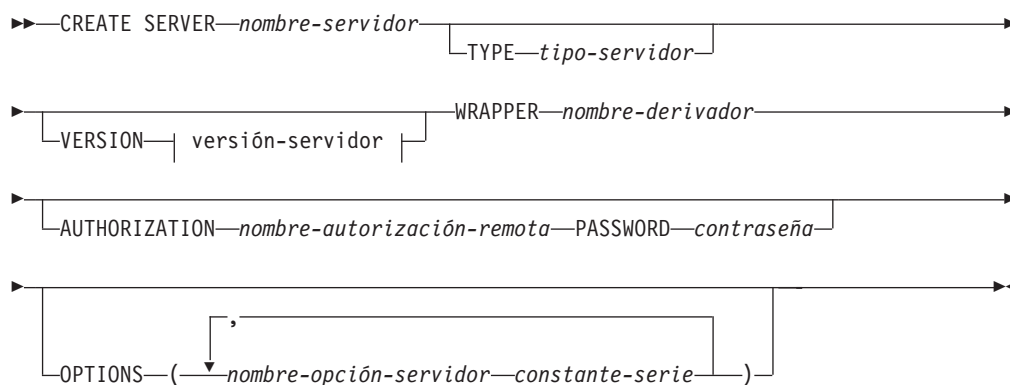
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

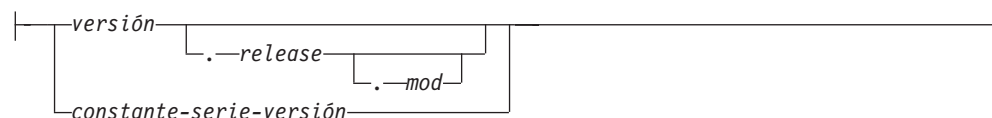
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

Sintaxis



versión-servidor:



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.

Normalmente, una definición de servidor para fuentes de datos relacionales representa una base de datos remota. Algunos sistemas de gestión de base de

CREATE SERVER

datos relacionales como, por ejemplo, Oracle, no admiten múltiples bases de datos en cada instancia. En su lugar, cada instancia representa un servidor en un sistema federado.

Para las fuentes de datos no relacionales, el propósito de una definición de servidor varía según la fuente de datos. Algunas definiciones de servidor se correlacionan con un tipo de búsqueda y daemon, un sitio web o un servidor web. Para otras fuentes de datos no relacionales, se crea una definición de servidor porque la jerarquía de objetos federados requiere que los archivos de fuente de datos (identificados por apodos) se asocien a un objeto de servidor específico.

TYPE *tipo-servidor*

Especifica el tipo de fuente de datos que *nombre-servidor* indica. Este parámetro es necesario para algunos derivadores.

VERSION

Especifica la versión de la fuente de datos indicada por *nombre-servidor*. Este parámetro es necesario para algunos derivadores.

versión

Especifica el número de versión. El valor debe ser un entero.

release

Especifica el número del release de la versión indicada por *versión*. El valor debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. El valor 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 estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-derivador*

Nombra el derivador que el servidor federado utiliza para interactuar con el objeto de servidor especificado por *nombre-servidor*.

AUTHORIZATION *nombre-autorización-remota*

Sólo se requiere para las fuentes de datos de la familia DB2. 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 de autorización no se utiliza al establecer conexiones posteriores con el servidor.

Este ID debe tener la autorización (BINDADD o su equivalente) que las acciones necesarias necesitan. Si se especifica el *nombre-autorización-remota* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota contiene nombres de autorización sensibles a las mayúsculas y minúsculas), el *nombre-autorización-remota* se debe especificar entre comillas dobles.

PASSWORD *contraseña*

Sólo se requiere para las fuentes de datos de la familia DB2. Especifica la contraseña asociada al ID de autorización representado por *nombre-autorización-remota*. Si se especifica la *contraseña* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota tiene contraseñas sensibles a las mayúsculas y minúsculas), la *contraseña* se debe especificar entre comillas dobles.

OPTIONS

Indica las opciones que se habilitan cuando se crea la definición de servidor. Las opciones de servidor se utilizan para configurar la definición de servidor. Algunas opciones de servidor pueden utilizarse a fin de crear la definición de servidor para cualquier fuente de datos. Algunas opciones de servidor son específicas para una fuente de datos en particular.

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

- La *contraseña* se debe especificar cuando la fuente de datos requiere una contraseña. Si las letras de una *contraseña* deben ir en minúsculas, especifique la *contraseña* entre comillas.
- Si se utiliza la sentencia CREATE SERVER para definir una instancia de la familia de DB2 como fuente de datos, puede que DB2 deba vincular determinados paquetes con esa instancia. Si es necesario una vinculación, el *nombre-autorización-remota* de la sentencia debe tener la autorización BIND. El tiempo necesario para que se complete la operación de vinculación depende de la velocidad de la fuente de datos y de la velocidad de conexión de la red.
- DB2 no comprueba que la versión del servidor especificada coincida con la versión del servidor remoto. Si se especifica una versión de servidor incorrecta se pueden provocar errores de SQL cuando se accede a apodos que pertenecen a la definición del servidor DB2. Esto puede producirse con mucha probabilidad cuando se especifica una versión de servidor posterior a la versión del servidor remoto. En ese caso, al acceder a apodos que pertenecen a la definición del servidor, puede que DB2 envíe SQL que el servidor remoto no reconoce.
- **Alternativas de sintaxis:** se soporta la sintaxis siguiente para mantener la compatibilidad con versiones anteriores de DB2:
 - ADD puede especificarse antes de *nombre-opción-servidor constante-serie*.

Ejemplos

- *Ejemplo 1:* Registre una definición de servidor para acceder a la fuente de datos DB2 para z/OS y OS/390, versión 7.1. CRANDALL es el nombre asignado a la definición del servidor DB2 para z/OS y OS/390. DRDA es el nombre del derivador utilizado para acceder a esta fuente de datos. Además, especifique que:
 - GERALD y drowssap son el ID de autorización y la contraseña bajo los cuales se vinculan los paquetes en CRANDALL cuando se procesa esta sentencia.
 - El alias para la base de datos DB2 para z/OS y OS/390 que se ha especificado con la sentencia CATALOG DATABASE es CLIENTS390.
 - Los ID de autorización y las contraseñas bajo los que se puede acceder a CRANDALL se deben enviar a CRANDALL en mayúsculas.
 - CLIENTS390 y la base de datos federada utilizan el mismo orden de clasificación.

```
CREATE SERVER CRANDALL
  TYPE DB2/ZOS
  VERSION 7.1
  WRAPPER DRDA
```

CREATE SERVER

```
AUTHORIZATION "GERALD"  
PASSWORD drowssap  
OPTIONS  
  (DBNAME 'CLIENTS390',  
   FOLD_ID 'U',  
   FOLD_PW 'U',  
   COLLATING_SEQUENCE 'Y')
```

- *Ejemplo 2:* Registre una definición de servidor para acceder a una fuente de datos Oracle 9. CUSTOMERS es el nombre asignado a la definición de servidor Oracle. NET8 es el nombre del derivador utilizado para acceder a esta fuente de datos. Además, especifique que:
 - ABC es el nombre del nodo en el que reside el servidor de bases de datos Oracle.
 - 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 9  
WRAPPER NET8  
OPTIONS  
  (NODE 'ABC',  
   CPU_RATIO '2.0',  
   IO_RATIO '1.5')
```

- *Ejemplo 3:* Registre una definición de servidor para el derivador Excel. La definición de servidor es necesaria para conservar la jerarquía de objetos federados. BIOCHEM_LAB es el nombre asignado a la definición de servidor Excel. EXCEL_2000_WRAPPER es el nombre del derivador utilizado para acceder a esta fuente de datos.

```
CREATE SERVER BIOCHEM_DATA  
WRAPPER EXCEL_2000_WRAPPER
```

CREATE STOGROUP

La sentencia CREATE STOGROUP define un grupo de almacenamiento nuevo dentro de la base de datos, asigna vías de acceso de almacenamiento al grupo de almacenamiento y registra la definición de grupo de almacenamiento y los atributos en el catálogo.

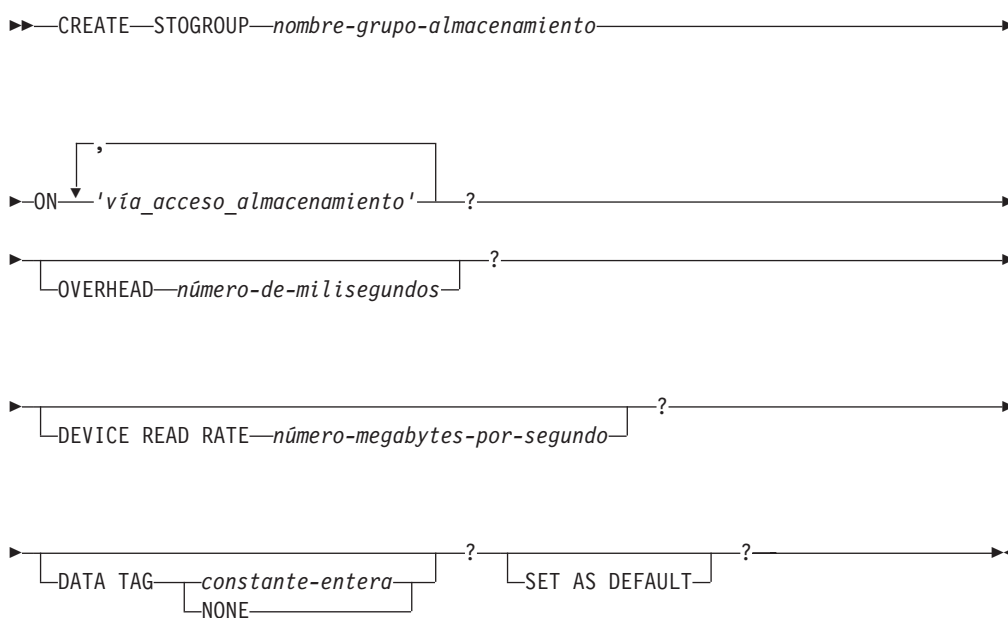
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis



Descripción

nombre-grupo-almacenamiento

Da nombre al grupo de almacenamiento. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo-almacenamiento* no debe identificar un grupo de almacenamiento que ya existe en el servidor actual (SQLSTATE 42710). El *nombre-grupo-almacenamiento* no debe empezar con los caracteres 'SYS' (SQLSTATE 42939).

ON Especifica las vías de acceso de almacenamiento que se deben añadir para el grupo de almacenamiento indicado. En el caso de entornos de bases de datos

CREATE STOGROUP

particionadas, se definirán las mismas vías de acceso de almacenamiento en todas las particiones de base de datos a menos que se utilicen expresiones de particiones de base de datos.

vía-acceso-almacenamiento

Constante de tipo serie que especifica una vía de acceso absoluta o la letra de una unidad (sólo en los sistemas operativos Windows) donde se crearán los contenedores para los espacios de tablas de almacenamiento automático. La serie puede incluir expresiones de partición de base de datos para especificar la información del número de partición de base de datos en la vía de acceso de almacenamiento.

La longitud máxima de una vía de acceso es de 175 caracteres (SQLSTATE 54036). La vía de acceso de almacenamiento que se está añadiendo debe ser válida según las normas de denominación y debe ser accesible (SQLSTATE 57019). De igual modo, en un entorno de base de datos particionada, la vía de acceso de almacenamiento debe existir y ser accesible en cada partición de base de datos (SQLSTATE 57019).

OVERHEAD *número-de-milisegundos*

Especifica el tiempo de latencia, búsqueda de disco y uso de controlador de E/S. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. El valor *número-de-milisegundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todas las vías de acceso de almacenamiento, establezca el valor en un literal numérico que represente el promedio de todas las vías de acceso de almacenamiento que pertenecen al grupo de almacenamiento.

Si no se especifica la cláusula OVERHEAD, OVERHEAD se establecerá en 6,725 milisegundos.

DEVICE READ RATE *número-megabytes-por-segundo*

Especifica la especificación de dispositivo para la velocidad de transferencia de lectura en megabytes por segundo. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. El valor de *número-megabytes-por-segundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todas las vías de acceso de almacenamiento, establezca el valor en un literal numérico que represente el promedio de todas las vías de acceso de almacenamiento que pertenecen al grupo de almacenamiento.

Si no se especifica la cláusula DEVICE READ RATE, DEVICE READ RATE se establecerá en el valor por omisión incorporado de 100 megabytes por segundo.

DATA TAG *constante-entero* o **DATA TAG NONE**

Especifica un código para los datos de los espacios de tablas que utilizan este grupo de almacenamiento a menos que lo altere temporalmente de forma explícita la definición de espacio de tablas. Este valor se puede utilizar como parte de una configuración de WLM en una definición de clase de trabajo o se puede referenciar en una definición de umbral. Para obtener más información, consulte las sentencias CREATE WORK CLASS SET y CREATE THRESHOLD.

constante-entera

Los valores de *constante-entera* son los enteros del 1 al 9.

NONE

Si se especifica NONE, no hay ningún código de datos.

SET AS DEFAULT

Especifica que el grupo de almacenamiento que se crea se designar como

grupo de almacenamiento por omisión. Si no existe ningún grupo de almacenamiento por omisión, el primero que se haya creado se designará como el valor por omisión aunque no se especifique esta cláusula. Puesto que sólo puede haber un grupo de almacenamiento designado como grupo de almacenamiento por omisión, al especificar esta cláusula se elimina el atributo por omisión del grupo de almacenamiento por omisión existente. La especificación de un nuevo grupo de almacenamiento por omisión no afecta al grupo de almacenamiento que los espacios de tablas existentes utilizan.

Normas

- No se puede ejecutar la sentencia CREATE STOGROUP mientras se está añadiendo un servidor de particiones de base de datos (SQLSTATE 55071).
- Un grupo de almacenamiento puede tener un máximo de 128 vías de acceso de almacenamiento definidas (SQLSTATE 5U009).
- Una instancia de base de datos puede tener un máximo de 256 grupos de almacenamiento definidos (SQLSTATE 54035).

Notas

- **Cálculo del espacio libre:** cuando se calcula el espacio libre para una vía de acceso de almacenamiento en una partición de base de datos, el gestor de bases de datos comprueba la existencia de los siguientes directorios o puntos de montaje en la vía de acceso de almacenamiento, y utiliza la primera que se encuentre.

```
<vía acceso almacenamiento>/<nombre instancia>/NODE####/<nombre base datos>
<vía acceso almacenamiento>/<nombre instancia>/NODE####
<vía acceso almacenamiento>/<nombre instancia>
<vía acceso almacenamiento>
```

Donde:

- <vía de acceso almacenamiento> es una vía de acceso de almacenamiento asociada con la base de datos.
- <nombre instancia> es la instancia en la que reside la base de datos.
- NODE#### corresponde al número de partición de base de datos (por ejemplo, NODE0000 o NODE0001).
- <nombre base de datos> es el nombre de la base de datos.
- **Aislamiento de varias particiones de base de datos en una vía de acceso de almacenamiento:** los sistemas de archivos se pueden montar en un punto bajo la vía de acceso de almacenamiento, y el gestor de bases de datos reconocerá que la cantidad real de espacio libre disponible para los contenedores del espacio de tablas puede no ser la misma que la asociada con el directorio de la vía de acceso de almacenamiento.
 Considere un ejemplo donde existan dos particiones de base de datos lógicas en un sistema físico, y haya una única vía de acceso de almacenamiento (/db2data). Cada partición de base de datos utilizará esta vía de acceso de almacenamiento, pero es posible que se desee aislar los datos de cada partición en su propio sistema de archivos. En este caso, se puede crear un sistema de archivos diferente para cada partición y se puede montar en /db2data/<instancia>/NODE####. Al crear contenedores en la vía de acceso de almacenamiento y determinar el espacio libre, el gestor de bases de datos no recuperará la información sobre el espacio libre para /db2data, sino que la recuperará para el directorio /db2data/<instancia>/NODE#### correspondiente.
- **Varias vías de acceso de almacenamiento:** se puede añadir una vía de acceso de almacenamiento a distintos grupos de almacenamiento o al mismo grupo de almacenamiento varias veces.

CREATE STOGROUP

- *Características de soportes similares:* Asegúrese de que las vías de acceso de almacenamiento añadidas a un grupo de almacenamiento tienen características de soportes similares. Si las características de soportes no son similares, especifique un valor que represente el promedio de OVERHEAD y DEVICE READ RATE.

Ejemplos

- *Ejemplo 1:* crear un grupo de almacenamiento denominado HIGHEND con dos vías de acceso en el directorio /db2 (/db2/filesystem1 y /db2/filesystem2) que son discos de estado sólido conectados.

```
CREATE STOGROUP HIGHEND ON '/dbe/filesystem1', '/db2/filesystem2'  
OVERHEAD 0.75 DEVICE READ RATE 500
```

- *Ejemplo 2:* crear un grupo de almacenamiento denominado MIDRANGE con dos unidades D y E y designarlo como grupo de almacenamiento por omisión.

```
CREATE STOGROUP MIDRANGE ON 'D:\', 'E:\' SET AS DEFAULT
```

CREATE SYNONYM

La sentencia CREATE SYNONYM define un sinónimo para un módulo, apodo, secuencia, tabla, vista u otro sinónimo.

Descripción

SYNONYM es sinónimo de ALIAS.

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. La definición puede incluir otros atributos de la tabla, como su clave primaria o restricciones de comprobación.

Para crear una tabla temporal creada, utilice la sentencia `CREATE GLOBAL TEMPORARY TABLE`. Para declarar una tabla temporal declarada, utilice la sentencia `DECLARE GLOBAL TEMPORARY TABLE`.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de `DYNAMICRULES` está en vigor para el paquete (`SQLSTATE 42509`).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización `DBADM` o la autorización `CREATETAB` en combinación con otra autorización, como se describe a continuación.

- Uno de los privilegios y autorizaciones siguientes:
 - Privilegio `USE` en el espacio de tablas
 - `SYSADM`
 - `SYSCTRL`
- Más uno de estos privilegios y autorizaciones:
 - Autorización `IMPLICIT_SCHEMA` para la base de datos, si el nombre de esquema implícito o explícito de la tabla no existe
 - 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 propietario 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:

- Privilegio `REFERENCES` para la tabla
- Privilegio `REFERENCES` para todas las columnas de la clave padre especificada
- Privilegio `CONTROL` sobre la tabla
- Autorización `DBADM`

Para definir una tabla de consulta materializada (utilizando una selección completa), los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes para cada tabla o vista identificada en la selección completa (excluyendo los privilegios de grupo):

- Privilegio `SELECT` para la tabla o vista
- Privilegio `CONTROL` sobre la tabla o vista
- Autorización `DATAACCESS`

Cuando va a definir una tabla de consulta materializada y especifica ciertas cláusulas de la sentencia CREATE TABLE, es probable que se necesite o se utilice autorización adicional en su lugar:

- Si WITH NO DATA está especificado, al menos una de las autorizaciones siguientes es suficiente:
 - DBADM
 - SQLADM
 - EXPLAIN
- Si se especifica REFRESH DEFERRED o REFRESH IMMEDIATE, como mínimo, uno de los siguientes privilegios o autorizaciones es necesario para cada tabla o vista que se identifique en la selección completa:
 - Privilegio ALTER para la tabla o vista
 - Privilegio CONTROL sobre la tabla o vista
 - Autorización DBADM

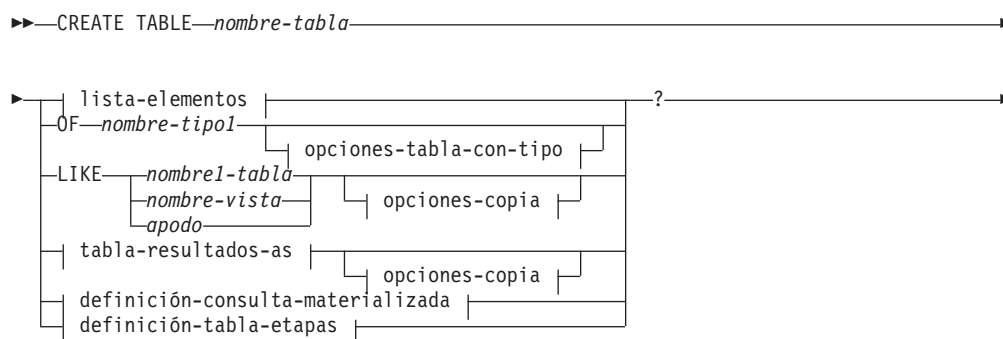
Para definir una tabla de etapas asociada a una tabla de consulta materializada, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes en la tabla de consulta materializada:

- Privilegio ALTER para la tabla de consulta materializada
- Privilegio CONTROL para la tabla de consulta materializada
- Autorización DBADM

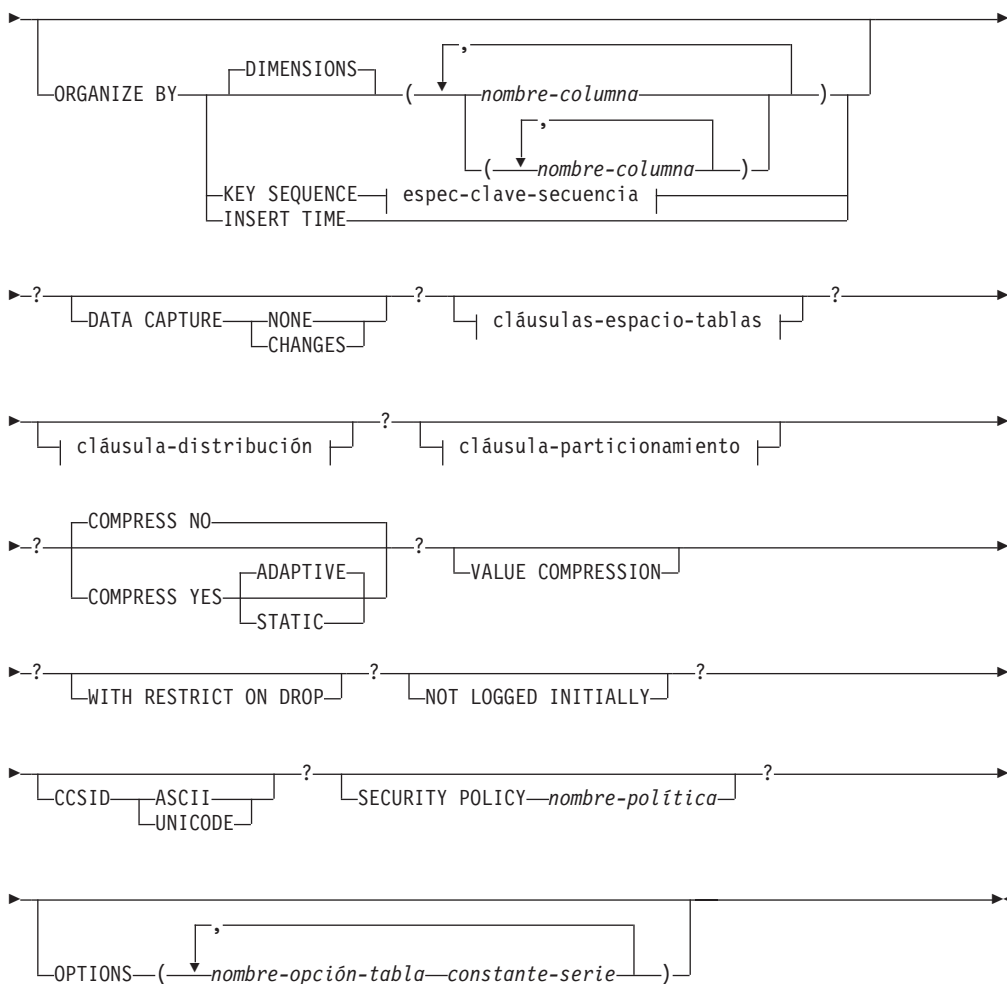
y, como mínimo, uno de los siguientes para cada tabla o vista que se identifique en la selección completa de la tabla de consulta materializada:

- Privilegio SELECT o autorización DATAACCESS para la tabla o vista y, como mínimo, uno de los siguientes:
 - Privilegio ALTER para la tabla o vista
 - Autorización DBADM
- Privilegio CONTROL sobre la tabla o vista

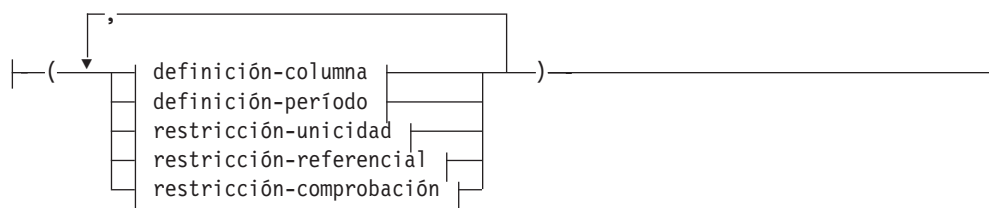
Sintaxis



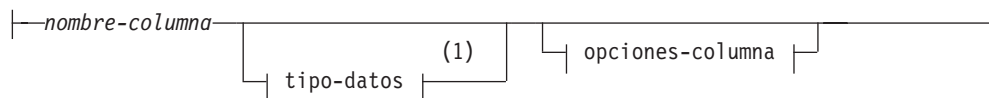
CREATE TABLE



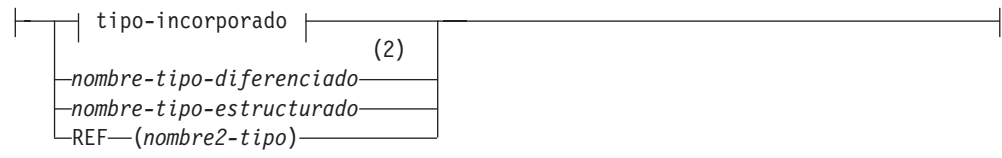
lista-elementos:



definición-columna:

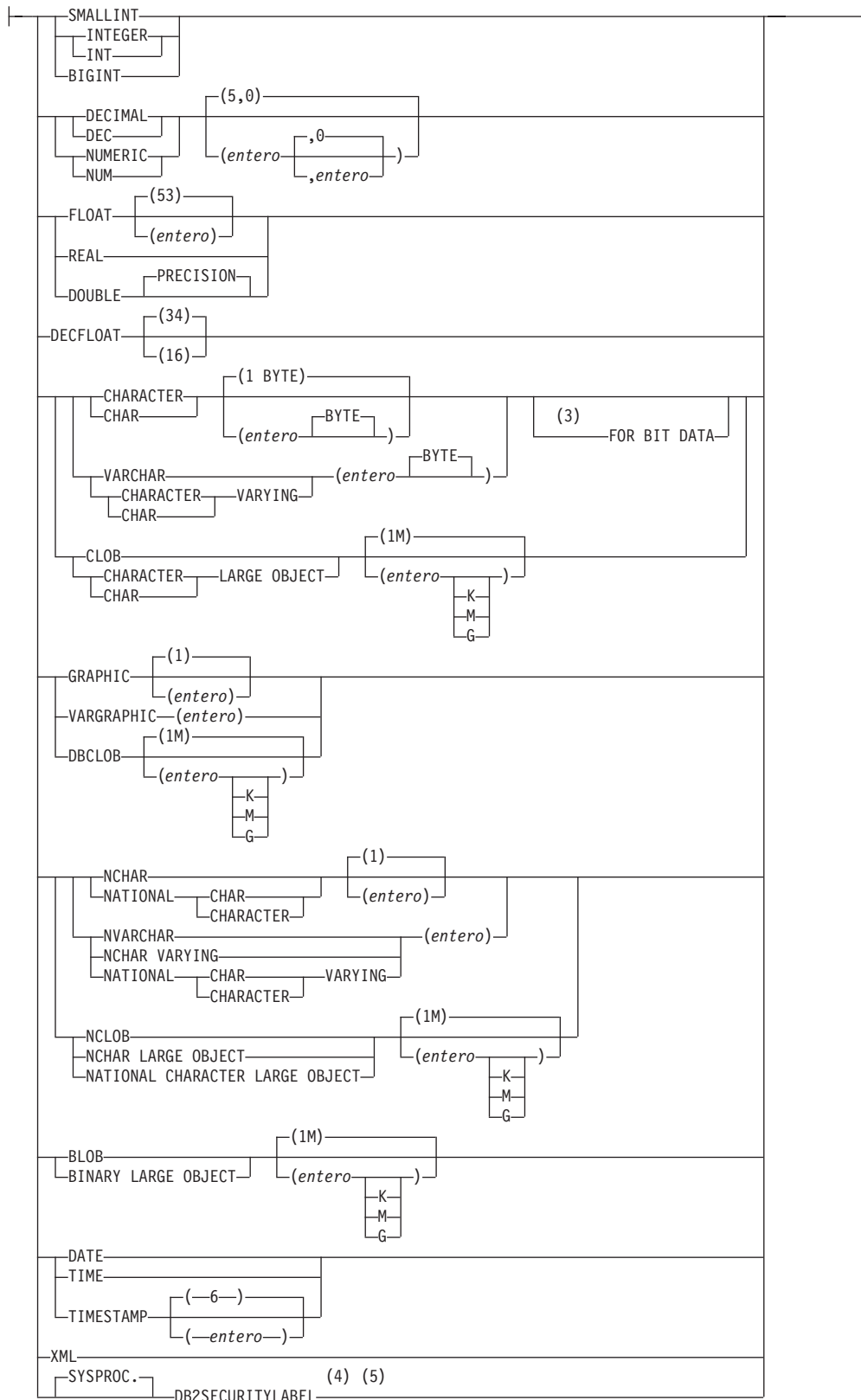


tipo-datos:

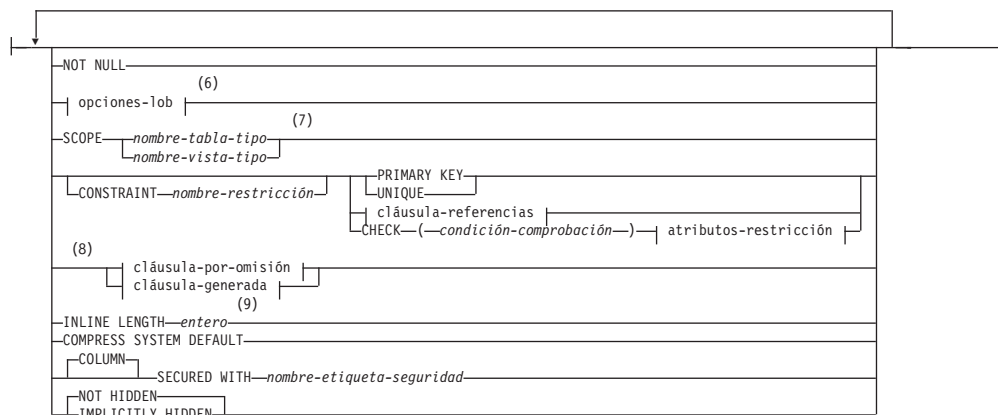


tipo-incorporado:

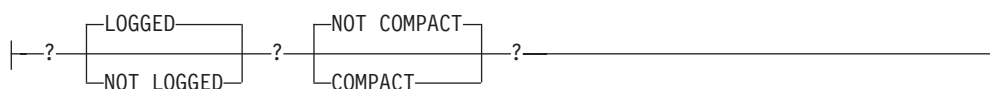
CREATE TABLE



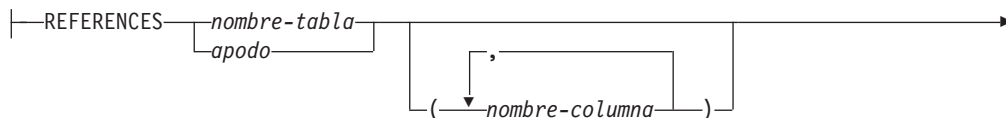
opciones-columna:



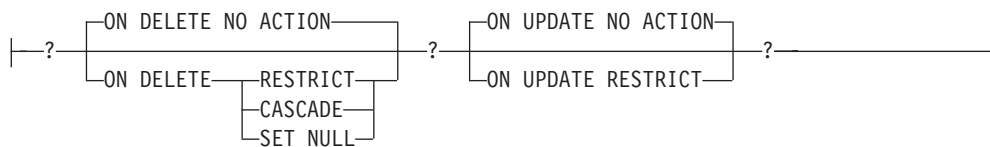
opciones-lob:



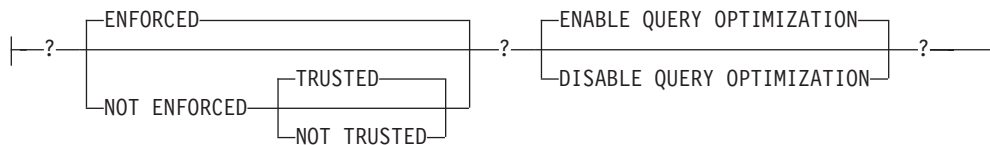
cláusula-referencias:



cláusula-norma:



atributos-restricción:

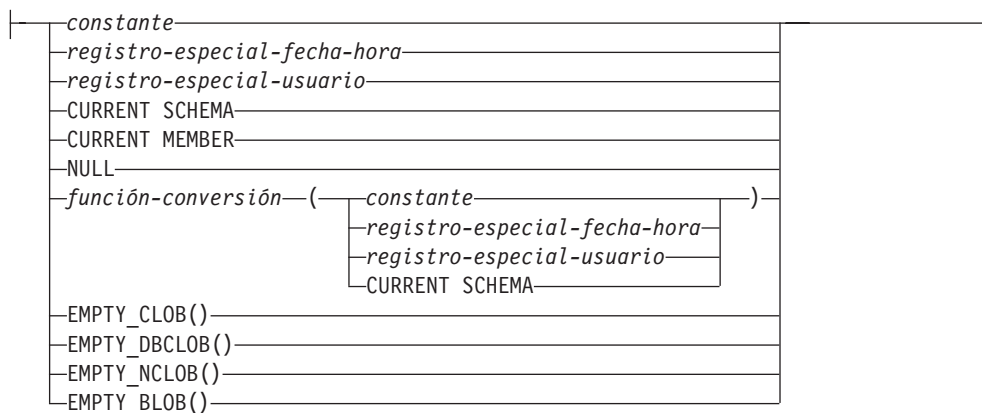


cláusula-por-omisión:

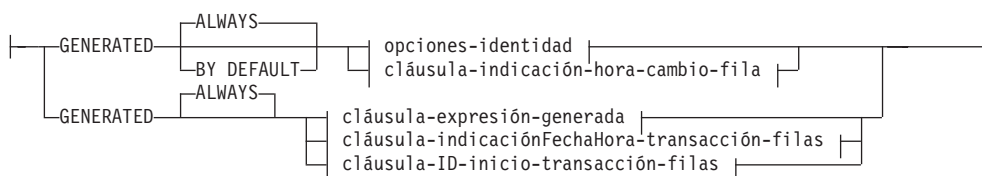


CREATE TABLE

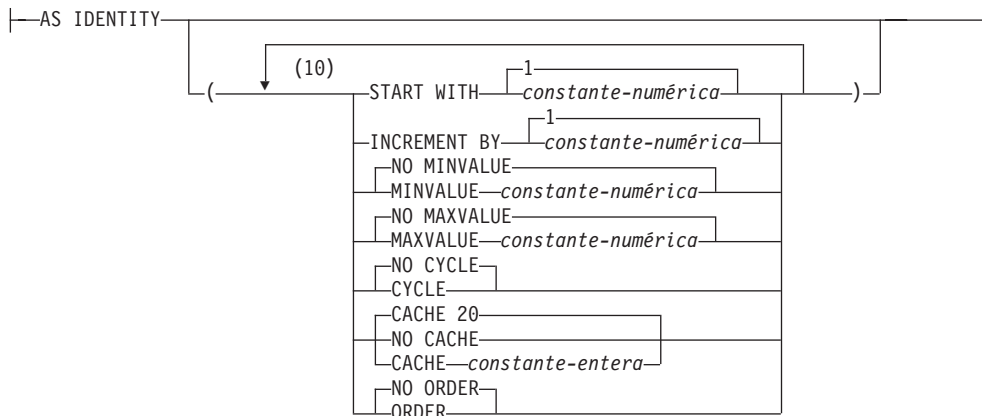
valores-omisión:



cláusula-generada:



opciones-identidad:



cláusula-indicación-hora-cambio-fila:



cláusula-expresión-generada:



cláusula-indicaciónFechaHora-transacción-filas:

(12)

```

|----- AS ROW [ BEGIN ]-----|
|----- [ END ]-----|

```

cláusula-ID-inicio-transacción-filas:

(13)

```

|----- AS TRANSACTION START ID-----|

```

definición-período:

```

|----- PERIOD [ SYSTEM_TIME ]-----|
|----- [ BUSINESS_TIME ]-----|
|----- ( nombre-columna-inicio , nombre-columna-fin )-----|

```

restricción-unicidad:

```

|----- [ CONSTRAINT nombre-restricción ]-----|
|----- [ UNIQUE ]-----|
|----- [ PRIMARY KEY ]-----|

```

```

|----- ( nombre-columna ,-----|
|----- [ BUSINESS_TIME WITHOUT OVERLAPS ]-----|
)-----|

```

restricción-referencia:

```

|----- [ CONSTRAINT nombre-restricción ]-----|
|----- FOREIGN KEY-----|
|----- ( nombre-columna )-----|

```

```

|----- cláusula-referencias-----|

```

restricción-comprobación:

```

|----- [ CONSTRAINT nombre-restricción ]-----|
|----- CHECK-----|
|----- ( nombre-comprobación )-----|

```

```

|----- atributos-restricción-----|

```

condición-error:

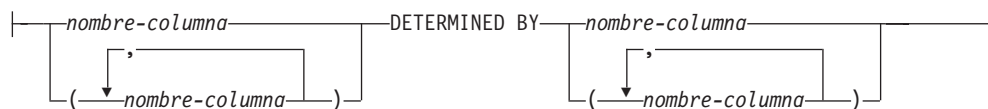
```

|----- condición-búsqueda-----|
|----- [ dependencia-funcional ]-----|

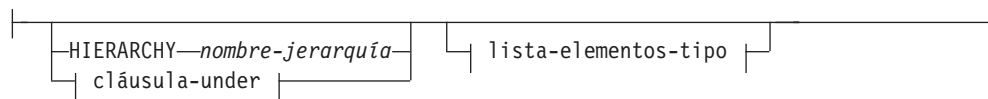
```

CREATE TABLE

dependencia-funcional:



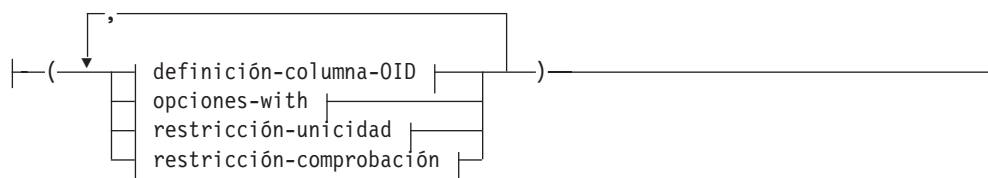
opciones-tabla-con-tipo:



cláusula-under:



lista-elementos-tipo:



definición-columna-OID:



opciones-with:

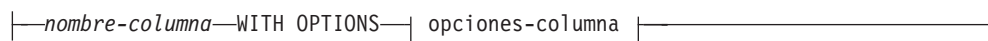
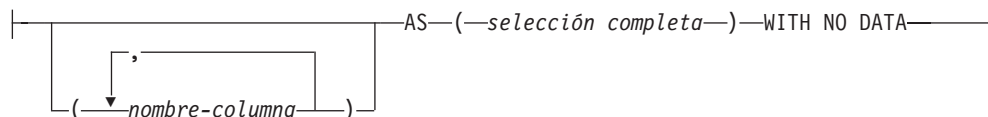
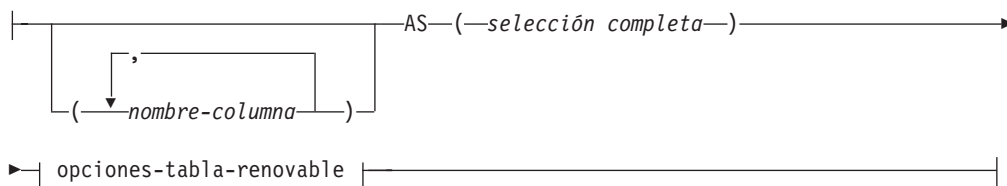


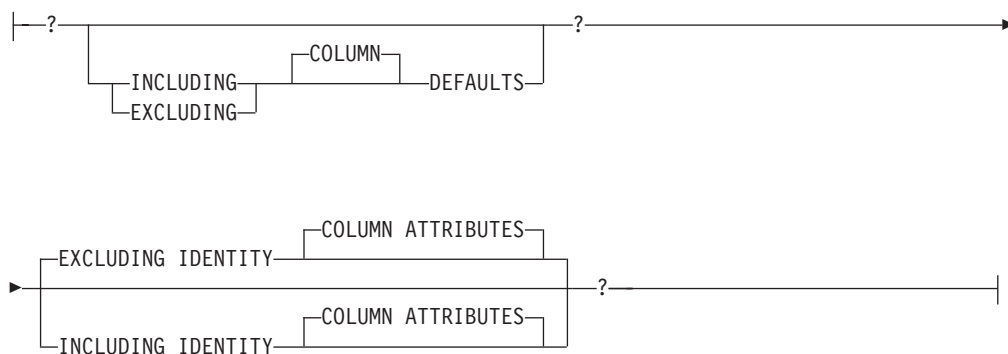
tabla-resultados-as:



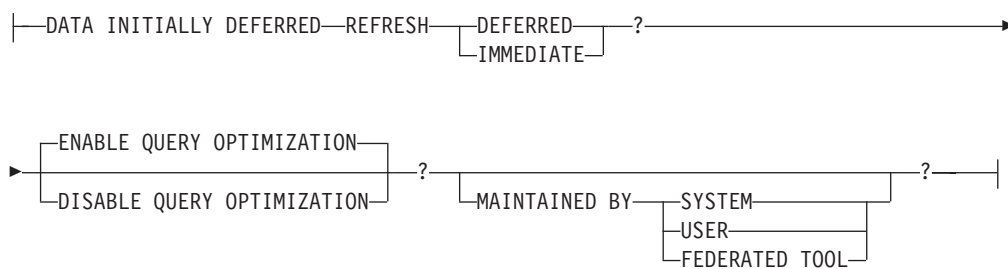
definición-consulta-materializada:



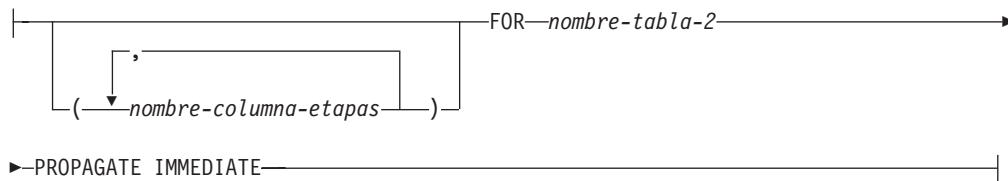
opciones-copia:



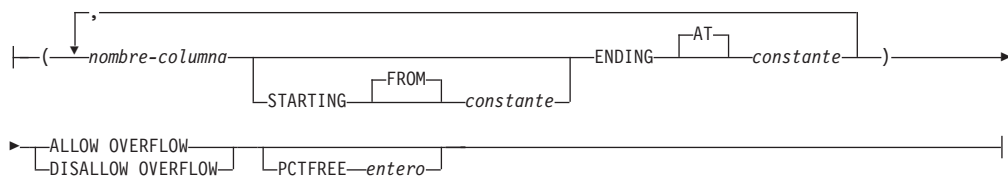
opciones-tabla-renovable:



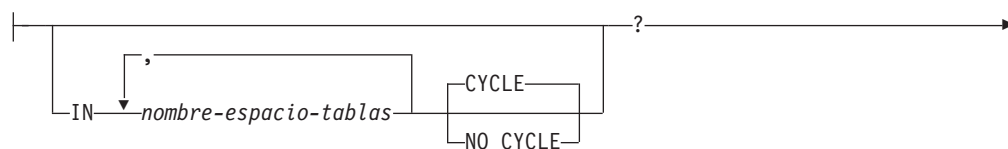
definición-tabla-etapas:



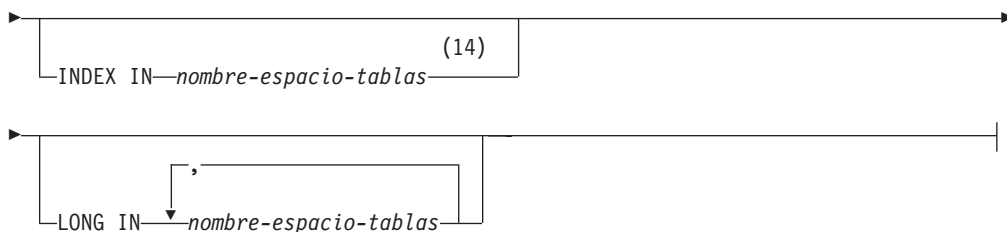
espec-clave-secuencia:



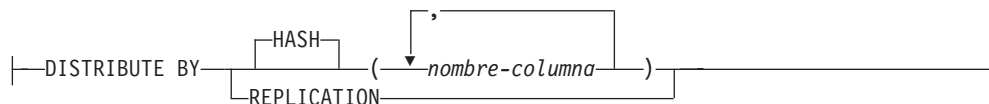
cláusulas-espacio-tablas:



CREATE TABLE



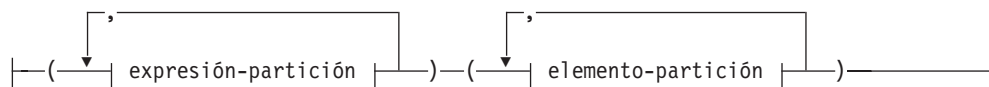
cláusula-distribución:



cláusula-particionamiento:



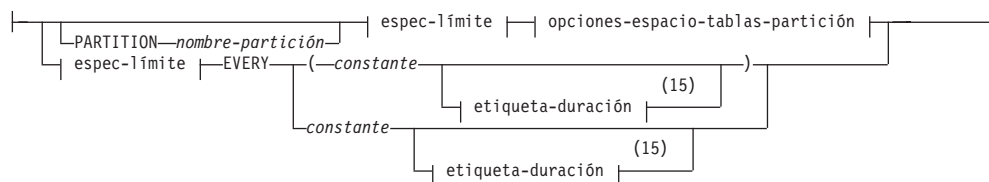
espec-partición-rangos:



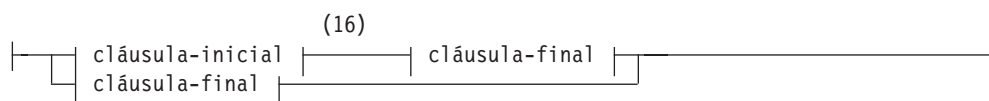
expresión-partición:



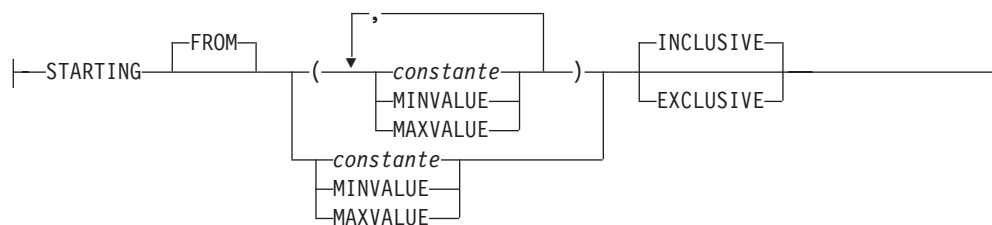
elemento-partición:



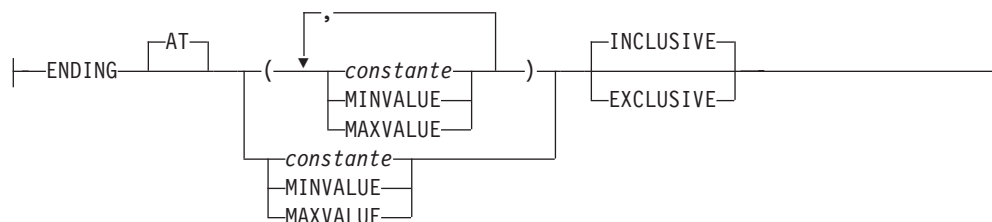
espec-límite:



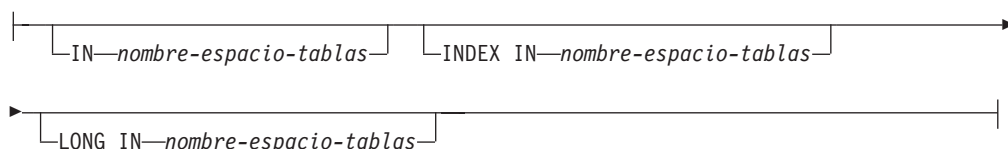
cláusula-inicial:



cláusula-final:



opciones-espacio-tablas-partición:



etiqueta-duración:



Notas:

- 1 Si la primera opción-columna elegida es una cláusula-generada una expresión-generación, el tipo-datos puede omitirse. Éste se determinará a partir del tipo de datos resultante de la expresión-generación.
- 2 El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados.
- 3 La cláusula FOR BIT DATA se puede especificar en cualquier orden con las restricciones de columna siguientes.

CREATE TABLE

- 4 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.
- 5 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.
- 6 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.
- 7 La cláusula SCOPE sólo se aplica al tipo REF.
- 8 La cláusula por omisión y la cláusula generada no se pueden especificar para la misma definición de columna (SQLSTATE 42614).
- 9 INLINE LENGTH sólo se aplica a columnas definidas como tipos estructurados , XML o LOB.
- 10 Una misma cláusula no se debe especificar más de una vez.
- 11 El tipo de datos es opcional para una columna de indicación de fecha y hora de cambio de fila si la primera opción-columna especificada es una cláusula-generada. El valor por omisión de tipo de datos es TIMESTAMP(6).
- 12 El tipo de datos es opcional para columnas de indicación de fecha y hora de inicio de fila y de fin de fila si la primera opción-columna especificada es una cláusula-generada. El valor por omisión de tipo de datos es TIMESTAMP(12).
- 13 El tipo de datos es opcional para una columna de indicación de fecha y hora de ID de inicio de transacción si la primera opción-columna especificada es una cláusula-generada. El valor por omisión de tipo de datos es TIMESTAMP(12).
- 14 Se puede especificar qué espacio de tablas contendrá los índices de una tabla al crearse la tabla. Si la tabla es una tabla particionada, el espacio de tablas de índice para un índice no particionado se puede especificar en la cláusula IN de la sentencia CREATE INDEX.
- 15 Esta sintaxis de un elemento-partición es válida si hay una sola expresión-partición con un tipo de datos numérico o de fecha y hora.
- 16 El primer elemento-partición debe contener una cláusula-inicial y el último elemento-partición debe incluir una cláusula-final.

Descripción

A las tablas de consulta materializada mantenidas por el sistema y a las tablas de consulta materializada mantenidas por el usuario se hace referencia por medio del término común *tabla de consulta materializada*, a menos que sea necesario identificarlas por separado.

nombre-tabla

Indica el nombre de la tabla. El nombre, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo ni alias descrito en el catálogo. El nombre de esquema no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

lista-elementos

Define los elementos de una tabla. Esto incluye la definición de las columnas y las restricciones de la tabla.

definición-columna

Define los atributos de una columna.

nombre-columna

Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no debe ser superior a 4.005.
- Un tamaño de página de 8K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 8.101.
- Un tamaño de página de 16K con un máximo de 1.012 columnas, donde el número total 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, donde el número total de bytes de las columnas no debe ser superior a 32.677.

Para obtener más información, consulte Límite del tamaño de fila.

tipo-datos

Especifica el tipo de datos de la columna.

tipo-incorporado

Para los tipos incorporados, utilice uno de los tipos siguientes.

SMALLINT

Para especificar un entero pequeño.

INTEGER o INT

Para especificar un entero grande.

BIGINT

Para especificar un entero superior.

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; los valores permitidos son del 1 al 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**.

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.

CREATE TABLE

DOUBLE PRECISION

Para especificar coma flotante de precisión doble.

FLOAT

Para especificar coma flotante de precisión doble.

DECFLOAT(*entero-precisión*)

Para especificar un número de coma flotante decimal. El valor de *entero-precisión* es la precisión del número; es decir, el número total de dígitos, los cuales pueden ser 16 ó 34.

Si no se especifica la precisión, se utilizará un valor por omisión de 34.

CHARACTER(*entero*) o CHAR(*entero*) o CHARACTER o CHAR

Para una serie de caracteres de longitud fija de *entero* bytes de longitud, que puede estar en el rango de 1 a 254. Si se omite la especificación de longitud, se supone una longitud de 1.

VARCHAR(*entero*) o CHARACTER VARYING(*entero*) o CHAR VARYING(*entero*)

Para una serie de caracteres de longitud variable de *entero* bytes de longitud máxima, que puede estar en el rango de 1 a 32.672.

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.

CLOB o CHARACTER (CHAR) LARGE OBJECT(*entero* [*K* | *M* | *G*])

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.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 (1 megabyte).

No es posible especificar la cláusula FOR BIT DATA para las columnas CLOB. Sin embargo, una serie CHAR FOR BIT DATA puede asignarse a una columna CLOB y una serie CHAR FOR BIT DATA puede concatenarse con una serie CLOB.

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.

DBCLOB(*entero* [*K* | *M* | *G*])

Para una serie de objeto grande 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. Las diferencias radican en que el número especificado es el número de caracteres de doble byte y que el tamaño máximo es 1.073.741.823 caracteres de doble byte.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 caracteres de doble byte.

NATIONAL CHARACTER (*entero*) o **NATIONAL CHAR** (*entero*) o **NCHAR** (*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.

NATIONAL CHARACTER VARYING (*entero*) o **NATIONAL CHAR VARYING** (*entero*) o **NCHAR VARYING** (*entero*) o **NVARCHAR** (*entero*)

Para una serie gráfica de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 16.336.

NATIONAL CHARACTER LARGE OBJECT (*entero*[*K*|*M*|*G*]) o **NCHAR LARGE OBJECT** (*entero*[*K*|*M*|*G*]) o **NCLOB**(*entero*[*K*|*M*|*G*])

Para una serie de objeto grande 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. Las diferencias radican en que el número especificado es el número de caracteres de doble byte y que el tamaño máximo es 1.073.741.823 caracteres de doble byte.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 caracteres de doble byte.

BLOB o **BINARY LARGE OBJECT**(*entero* [*K* | *M* | *G*])

Para una serie de objeto grande binario de la longitud máxima especificada en bytes.

La longitud puede estar en el rango de 1 byte a 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 *entero* multiplicado por 1.024. El valor máximo para *entero* es 2.097.152.

Si se especifica *entero* M, la longitud máxima es *entero* multiplicado por 1.048.576. El valor máximo para *entero* es 2.048.

Si se especifica *entero* G, la longitud máxima es *entero* multiplicado por 1.073.741.824. El valor máximo de *entero* es 2.

Si se especifica un múltiplo de K, M o G cuyo cálculo da 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.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 (1 megabyte).

Para crear series BLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

Está permitido especificar cualquier número de espacios entre el *entero* y K, M o G; por otra parte, no es obligatorio especificar un espacio. Por ejemplo, todos los valores siguientes son válidos:

BLOB(50K) BLOB(50 K) BLOB (50 K)

DATE

Para la fecha.

CREATE TABLE

TIME

Para la hora.

TIMESTAMP(*entero*) o TIMESTAMP

Para una indicación de fecha y hora. El *entero* debe estar entre 0 y 12 y especifica la precisión en segundos fraccionarios de 0 (segundos) a 12 (picosegundos). El valor por omisión es 6 (microsegundos).

XML

Para un documento XML. Solo se pueden insertar documentos XML con el formato correcto en una columna XML.

Una columna XML tiene las restricciones siguientes:

- La columna no puede formar parte de ningún índice excepto un índice sobre datos XML. 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 42962).
- No se puede especificar un valor por omisión (WITH DEFAULT) para la columna (SQLSTATE 42613). Si la columna puede contener nulos, el valor por omisión de la columna es el valor nulo.
- La columna no se puede utilizar como clave de distribución (SQLSTATE 42997).
- La columna no se puede utilizar como clave de particionamiento de datos (SQLSTATE 42962).
- La columna no se puede utilizar para organizar una tabla de clúster multidimensional (MDC)(SQLSTATE 42962).
- No se puede emplear la columna en una tabla agrupada por clústeres de rangos (SQLSTATE 429BG).
- No se puede hacer referencia a la columna en una restricción de comprobación excepto en un predicado VALIDATED (SQLSTATE 42621).

Cuando se crea una columna de tipo XML, se crea un índice de vías de acceso XML en esa columna. También se crea un índice de regiones XML en el nivel de tabla cuando se crea la primera columna de tipo XML. El nombre de estos índices es 'SQL' seguido de una indicación de fecha y hora de caracteres (*aaammddhhmmssxxx*). El nombre de esquema es SYSIBM.

SYSPROC.DB2SECURITYLABEL

Es un tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida. El tipo de datos subyacente de una columna del tipo diferenciado incorporado DB2SECURITYLABEL es VARCHAR(128) FOR BIT DATA. Una tabla puede tener como máximo una columna de tipo DB2SECURITYLABEL (SQLSTATE 428C1).

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 (definida por la opción de

preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico).

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 de fuente del tipo diferenciado. El tipo diferenciado especificado no puede tener ninguna restricción de tipo de datos y el tipo fuente no puede ser un tipo de datos anclados (SQLSTATE 428H2).

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*.

Una columna definida utilizando un tipo estructurado no se puede utilizar en una clave primaria, restricción de unicidad, clave foránea, clave de índice o clave de distribución (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 eliminación de referencia, es necesario especificar explícitamente un ámbito (SCOPE), utilizando una especificación CAST.

REF (*nombre2-tipo*)

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 FUNCPATH 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 se basa en el tipo de datos de representación que se ha 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 que incluye a *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.

CREATE TABLE

NOT HIDDEN o IMPLICITLY HIDDEN

Especifica si la columna ha de definirse como oculta. El atributo oculto determina si la columna se incluye como referencia implícita a la tabla o si se le puede hacer referencia de manera explícita en sentencias de SQL. El valor por omisión es NOT HIDDEN.

NOT HIDDEN

Especifica que la columna se incluye como referencia implícita a la tabla y que se puede hacer referencia a la columna de manera explícita.

IMPLICITLY HIDDEN

Especifica que la columna no es visible en sentencias de SQL a menos que se haga referencia a la columna de manera explícita por el nombre. Por ejemplo, suponiendo que una tabla incluye una columna definida con la cláusula IMPLICITLY HIDDEN, el resultado de SELECT * no incluye la columna implícitamente oculta. Sin embargo, el resultado de SELECT que haga referencia de manera explícita al nombre de una columna implícitamente oculta incluirá dicha columna en la tala de resultados.

No debe especificarse IMPLICITLY HIDDEN para todas las columnas de la tabla (SQLSTATE 428GU).

opciones-lob

Especifica opciones para los tipos de datos LOB.

LOGGED

Especifica que los cambios efectuados en la columna deben registrarse en el archivo de anotaciones cronológicas. 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.

NOT LOGGED

Especifica que los cambios efectuados en la columna no se van a anotar cronológicamente. Sólo se aplica a datos LOB que no están en línea.

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 anomalía general, todos los cambios confirmados y los cambios retrotraídos reflejarán los resultados esperados.

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 para facilitar los cambios futuros que se efectúen en los valores LOB de la columna. Es el valor por omisión.

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 eliminación de referencia o como argumento de la función Deref. La especificación del ámbito de una columna de tipo de referencia 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, el sistema genera un identificador de 18 bytes que es exclusivo entre los identificadores de las restricciones existentes que se han definido en la tabla. (El identificador se compone de la palabra "SQL" seguida de una secuencia de 15 caracteres numéricos que genera una función basada en la indicación de fecha y hora.)

Si se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* se puede utilizar como nombre de un índice 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.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

CREATE TABLE

No puede utilizarse una columna ROW CHANGE TIMESTAMP como parte de una clave primaria (SQLSTATE 429BV).

Las columnas de inicio de fila, fin de fila e ID de inicio de transacción no se pueden utilizar como parte de una clave primaria (SQLSTATE 429BV).

Consulte PRIMARY KEY en la descripción de *restricción-unicidad*.

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 UNIQUE en la descripción de *restricción-unicidad*.

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 *cláusula-referencias* en la descripción de *restricción-referencial*.

CHECK (*condición-comprobación*)

Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Consulte la descripción de CHECK (*condición-comprobación*).

cláusula-por-omisión

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 "ALTER TABLE". Esta cláusula no debe especificarse con cláusula-generada en una definición de columna (SQLSTATE 42614).

Si una columna se define como XML, no se puede especificar 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-por-omisión* (SQLSTATE 42842).

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 normas de asignación
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- ser una constante numérica o un valor especial de coma flotante decimal si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y, a continuación, se convierten a coma flotante decimal, si la columna de destino es DECFLOAT. Para las columnas DECFLOAT(16), las constantes decimales que tengan una precisión de más de 16 dígitos se redondearán utilizando las modalidades de redondeo especificadas por el registro especial CURRENT DECFLOAT ROUNDING MODE.
- no tener dígitos que no sean cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal (por ejemplo, 1.234 no puede ser el valor por omisión de una columna DECIMAL(5,2)).
- estar expresada con un máximo de 254 bytes, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificados y paréntesis, cuando la constante es el argumento de una *función-conversión*

registro-especial-fecha-hora

Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de 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).

registro-especial-usuario

Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de

CREATE TABLE

caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER.

CURRENT SCHEMA

Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud superior o igual al atributo de longitud del registro especial CURRENT SCHEMA.

CURRENT MEMBER

Especifica el valor que tenía el registro especial CURRENT MEMBER en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT MEMBER, el tipo de datos de la columna debe permitir la asignación desde un entero.

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 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 esta debe coincidir con el nombre del tipo de 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 indicación 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 normas de una constante para el tipo de 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 serie de caracteres.

registro-especial-fecha-hora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario

Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de 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.

CURRENT SCHEMA

Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()

Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

cláusula-generada

Especifica un valor generado para la columna.

GENERATED

Especifica que DB2 generará los valores para la columna. Debe especificarse GENERATED si la columna se tiene que considerar como columna de identidad o columna de indicación de fecha y hora de cambio de fila, columna de inicio de fila, columna de fin de fila, columna de ID de inicio de transacción o columna de expresión generada. No se debe especificar una cláusula por omisión para una columna definida como GENERATED (SQLSTATE 42623).

ALWAYS

Especifica que se generará siempre un valor para la columna cuando se inserte una fila en la tabla o cuando cambie el valor del resultado de *expresión-generación*. El resultado de la expresión se almacena en la tabla. GENERATED ALWAYS es el valor recomendado a menos que se estén realizando operaciones de propagación de datos o de descarga y de recarga. GENERATED ALWAYS es el valor obligatorio para las columnas generadas.

BY DEFAULT

Especifica que DB2 generará un valor para la columna cuando se inserte una fila o se actualice mediante la especificación de la cláusula DEFAULT, a menos que se especifique un valor

CREATE TABLE

explícito. BY DEFAULT es el valor recomendado cuando se utiliza la propagación de datos o se realiza una operación de descarga o recarga.

Aunque no se solicita explícitamente, defina un índice de una sola columna exclusivo en las columnas IDENTITY generadas para garantizar la exclusividad de los valores.

AS IDENTITY

Especifica que la columna va a ser la columna de identidad para esta tabla. Una tabla sólo puede contener una única columna de identidad (SQLSTATE 428C1). La palabra clave IDENTITY sólo puede especificarse si el tipo de datos que se asocia a la columna es un tipo numérico exacto con una escala cero o un tipo diferenciado definido por el usuario para el que el tipo de fuente es un tipo numérico exacto con una escala cero (SQLSTATE 42815). Se consideran tipos numéricos exactos SMALLINT, INTEGER, BIGINT o DECIMAL con una escala cero o un tipo diferenciado basado en uno de estos tipos. En cambio, las comas flotantes de precisión simple o doble se consideran tipos de datos numéricos aproximados. Los tipos de referencia, aunque se hayan representando por medio de un tipo numérico exacto, no pueden definirse como columnas de identidad.

Las columnas de identidad están definidas implícitamente como no nulas (NOT NULL). Una columna de identidad no puede tener una cláusula DEFAULT (SQLSTATE 42623).

START WITH *constante-numérica*

Especifica el primer valor de la columna de identidad. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). El valor por omisión es MINVALUE para las secuencias ascendentes y MAXVALUE para las secuencias descendentes. Este valor no es necesariamente el valor al que se aplicará el ciclo después de alcanzarse el valor mínimo o máximo de la columna de identidad. La cláusula START WITH puede utilizarse para iniciar la generación de valores fuera del rango que se utiliza para los ciclos. El rango utilizado para los ciclos se define mediante MINVALUE y MAXVALUE.

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 pueda asignarse a esta columna (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente. Si este valor es 0 o positivo, es una secuencia ascendente. El valor por omisión es 1.

NO MINVALUE o MINVALUE

Especifica el valor mínimo en el que una columna de identidad descendente ejecuta un ciclo o detiene la generación de valores o en el que una columna de identidad ascendente ejecuta un ciclo tras haberse alcanzado el valor máximo.

NO MINVALUE

Para una secuencia ascendente, el valor es el valor START WITH o bien 1 si no se ha especificado START WITH. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos de la columna. Es el valor por omisión.

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 esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor o igual al valor máximo (SQLSTATE 42815).

NO MAXVALUE o MAXVALUE

Especifica el valor máximo en el que una columna de identidad ascendente ejecuta un ciclo o detiene la generación de valores o en el que una columna de identidad descendente ejecuta un ciclo tras haberse alcanzado el valor mínimo.

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 START WITH o bien -1 si no se ha especificado START WITH. Es el valor por omisión.

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 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser superior o igual al valor mínimo (SQLSTATE 42815).

NO CYCLE o CYCLE

Especifica si esta columna de identidad debe o no seguir generando valores tras la generación de su valor máximo o de su valor mínimo.

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. Es el valor por omisión.

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, después de que una columna de identidad ascendente haya alcanzado el valor máximo, generará su valor mínimo; o después de que una secuencia descendente haya alcanzado el valor mínimo, generará su 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, DB2 puede generar valores duplicados para una columna de identidad. Aunque no se solicita explícitamente, debe definirse un índice de una sola columna, exclusivo, en la columna generada para

garantizar la existencia de valores exclusivos, si se necesitan valores exclusivos. Si existe un índice exclusivo en una columna de identidad de este tipo y se genera un valor que no es exclusivo, se produce un error (SQLSTATE 23505).

NO CACHE o CACHE

Especifica si deben mantenerse en la memoria algunos valores preasignados, para 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. Sin embargo, cuando se necesita un nuevo valor para la columna de identidad y existe un valor no utilizado en la antememoria, la asignación de dicho valor de identidad es más rápida, pues no es necesario realizar ninguna anotación cronológica. Esta opción se utiliza para el rendimiento y el ajuste.

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 petición de un valor de identidad nuevo produce E/S síncrona en las anotaciones cronológicas.

CACHE *constante-entera*

Especifica cuántos valores de la secuencia de identidad deben asignarse previamente y mantenerse en la memoria. Cuando se generan valores para la columna de identidad, la asignación previa y el almacenamiento de los valores en la antememoria reducen la E/S síncrona en el archivo de anotaciones cronológicas.

Si se necesita un nuevo valor para la columna de identidad y no existen valores no utilizados disponibles en la antememoria, la asignación del valor implica tener que esperar a que se produzca la E/S en el archivo de anotaciones cronológicas. 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.

En caso de que se produzca una desactivación de la base de datos, realizada con normalidad o como consecuencia de una anomalía en el sistema, todos los valores de secuencia que se han colocado en la antememoria y que no se han utilizado en las sentencias confirmadas se *pierden*; es decir, nunca se utilizarán. 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. (Si una base de datos no se activa explícitamente, utilizando el mandato ACTIVATE o la API, cuando la última aplicación se desconecte de la base de datos, tendrá lugar una desactivación implícita.)

El valor mínimo es 2 (SQLSTATE 42815). El valor por omisión es CACHE 20.

En un entorno DB2 pureScale o de varias particiones, utilice las opciones CACHE y NO ORDER para que varios miembros de DB2 puedan poner en antememoria valores de secuencia de forma simultánea.

En un entorno DB2 pureScale, si se especifican tanto CACHE como ORDER, la especificación de ORDER altera temporalmente la especificación de CACHE y, en su lugar, NO CACHE entrará en vigor.

NO ORDER o ORDER

Especifica si los valores de identidad deben o no generarse en el orden en que se solicitan.

NO ORDER

Especifica que los valores no deben generarse en el orden en el que se solicitan. Es el valor por omisión.

ORDER

Especifica que los valores deben generarse en el orden en el que se solicitan.

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

Especifica que la columna es una columna de indicación de fecha y hora para la tabla. Se genera un valor para la columna en cada fila que se inserta y para cualquier fila de cualquier columna que se actualiza. El valor que se genera para una columna ROW CHANGE TIMESTAMP es una indicación de fecha y hora que corresponde a la hora de inserción o actualización de la fila. Si se insertan o actualizan varias filas con una sentencia única, el valor de la columna ROW CHANGE TIMESTAMP puede ser diferente para cada fila.

Una tabla sólo puede tener una columna ROW CHANGE TIMESTAMP (SQLSTATE 428C1). Si se especifica *tipo-datos*, debe ser TIMESTAMP o TIMESTAMP(6) (SQLSTATE 42842). Una columna ROW CHANGE TIMESTAMP no puede tener una cláusula DEFAULT (SQLSTATE 42623). Debe especificarse NOT NULL para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42831).

AS (*expresión-generación*)

Especifica que la definición de la columna se basa en una expresión. (Si la expresión para una columna GENERATED ALWAYS incluye una función externa definida por el usuario, el cambio del ejecutable de la función (de forma que los resultados cambien argumentos determinados) puede dar como resultado la existencia de datos no coherentes. Esto puede evitarse utilizando la sentencia SET INTEGRITY para forzar la generación de nuevos valores.) La *expresión-generación* no puede contener ninguno de los elementos siguientes (SQLSTATE 42621):

- Subconsultas
- Expresiones XMLQUERY o XMLEXISTS
- Funciones de columna
- Operaciones de eliminación de referencia o funciones Deref

CREATE TABLE

- Funciones definidas por el usuario o funciones incorporadas que no son determinantes
- Funciones definidas por el usuario mediante la opción EXTERNAL ACTION
- Funciones definidas por el usuario que no estén definidas con NO SQL
- Variables del lenguaje principal o marcadores de parámetros
- Registros especiales y funciones incorporadas que dependen del valor de un registro especial
- Variables globales
- Referencias a columnas definidas más adelante en la lista de columnas
- Referencias a otras columnas generadas
- Referencias a columnas de tipo XML

El tipo de datos de la columna se basa 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 asignarán valores a la columna de acuerdo con las normas de asignación correspondientes. 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 y el tipo de datos de resultado de *expresión-generación* debe tener definida la igualdad definida (consulte "Asignaciones y comparaciones"). Esto excluye las columnas y las expresiones de generación de los tipos de datos LOB, XML, tipos estructurados y tipos diferenciados basados en cualquiera de estos tipos (SQLSTATE 42962).

AS ROW BEGIN

Especifica que el gestor de bases de datos asigna el valor generado siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El valor se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exija que se asigne un valor a la columna de inicio de fila o la columna de ID de inicio de transacción de la tabla, o que se suprima una fila de una tabla temporal de período de sistema.

En el caso de una tabla temporal de período de sistema, el gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de inicio de fila en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas insertadas en una tabla histórica asociada tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio. Ello puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema. El parámetro de configuración de base de datos **system_time_period_adj** debe establecerse en Yes para que tenga lugar este ajuste en el valor de indicación de fecha y hora. Si se insertan o actualizan varias filas en una sola transacción de SQL y no es necesario un ajuste, los valores de la columna de inicio de fila son iguales para todas las filas y son exclusivos entre los valores generados para la columna

en otra transacción. Se necesita una columna de inicio de fila como columna de inicio de un período SYSTEM_TIME, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de inicio de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12) (SQLSTATE 42842). La columna no puede tener una cláusula DEFAULT (SQLSTATE 42623) y debe definirse como NOT NULL (SQLSTATE 42831). Las columnas de inicio de fila no se pueden actualizar.

AS ROW END

Especifica que el gestor de bases de datos asigna el valor máximo del tipo de datos de la columna (9999-12-30-00.00.00.000000000000) siempre que se inserta una fila o se actualiza una columna de la fila.

Se necesita una columna de fin de fila como segunda columna de un período SYSTEM_TIME, lo cual es el uso esperado de este tipo de columna generada.

Una tabla sólo puede tener una columna de fin de fila (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12) (SQLSTATE 42842). La columna no puede tener una cláusula DEFAULT (SQLSTATE 42623) y debe definirse como NOT NULL (SQLSTATE 42831). Las columnas de fin de fila no se pueden actualizar.

AS TRANSACTION START ID

Especifica que el gestor de bases de datos asigna el valor siempre que se inserta una fila en la tabla o que se actualiza una columna en la fila. El gestor de bases de datos asigna un valor de indicación de fecha y hora exclusivo por transacción o el valor nulo. El valor nulo se asigna a la columna de ID de inicio de transacción si la columna puede contener nulos y si existe una columna de inicio de fila en la tabla en la que el valor no tiene que ajustarse. De lo contrario, el valor se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exija que se asigne un valor a la columna de inicio de fila o la columna de ID de inicio de transacción de la tabla, o que se suprima una fila de una tabla temporal de período de sistema. Si se insertan o actualizan varias filas en una sola transacción de SQL, los valores de la columna de ID de inicio de transacción son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.

Se necesita una columna de ID de inicio de transacción para una tabla temporal de período de sistema, lo cual es el uso esperado para este tipo de columna generada.

Una tabla solamente puede tener una columna de ID de inicio de transacción (SQLSTATE 428C1). Si no se especifica *tipo-datos*, la columna se define como TIMESTAMP(12). Si se especifica *tipo-datos*, debe ser TIMESTAMP(12). Una columna de ID de inicio

CREATE TABLE

de transacción no puede tener una cláusula DEFAULT (SQLSTATE 42623). Una columna de ID de inicio de transacción no se puede actualizar.

INLINE LENGTH *entero*

Esta opción sólo es válida para una columna definida mediante un tipo estructurado, XML o un tipo de datos LOB (SQLSTATE 42842).

Para una columna de tipo de datos XML o LOB, *entero* indica el tamaño máximo en bytes de la representación interna de un documento XML o datos LOB que debe almacenarse en la fila de la tabla base. Los documentos de XML que tengan una representación interna más grande se almacenarán de modo independiente de la fila de tabla base en un objeto de almacenamiento auxiliar. Esta operación se realiza automáticamente. No hay longitud en línea por omisión para las columnas del tipo XML. Si el documento XML o los datos LOB se almacenan en línea en la fila de la tabla base, habrá una actividad general adicional. Para los datos LOB, la actividad general es de 4 bytes.

Para una columna de tipo de datos LOB, la longitud en línea por omisión se establece en el tamaño máximo del descriptor de LOB si la cláusula no se especifica. Todo valor INLINE LENGTH explícito debe tener como mínimo el tamaño máximo del descriptor LOB. La tabla siguiente resume los tamaños del descriptor LOB.

Tabla 19. Tamaños del descriptor LOB para varias longitudes de LOB

Longitud máxima de LOB en bytes	INLINE LENGTH explícito mínimo
1.024	68
8.192	92
65.536	116
524.000	140
4.190.000	164
134.000.000	196
536.000.000	220
1.070.000.000	252
1.470.000.000	276
2.147.483.647	312

Para una columna de tipo estructurado, *entero* indica el tamaño máximo en bytes de una instancia de un tipo estructurado que debe almacenarse en línea con el resto de los valores de la fila. Las instancias de tipos estructurados que no se pueden almacenar en línea se almacenan por separado de la fila de la tabla base, de forma similar a como se almacenan 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 longitud en línea de la columna.

Nota: Las longitudes en línea de los subtipos no se cuentan en la longitud en línea por omisión, lo que significa que las instancias de los subtipos pueden no caber en línea 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 no puede superar 32.673. Para un tipo estructurado o de datos XML, debe ser, como mínimo, 292 (SQLSTATE 54010).

COMPRESS SYSTEM DEFAULT

Especifica que los valores por omisión del sistema se deben almacenar utilizando el mínimo espacio. Si no se especifica la cláusula `VALUE COMPRESSION`, se devuelve un aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el mínimo espacio.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

El tipo de datos base no puede ser `DATE`, `TIME`, `TIMESTAMP`, `XML` ni un tipo de datos estructurado (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto. Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con `VALUE COMPRESSION`.

COLUMN SECURED WITH *nombre-etiqueta-seguridad*

Identifica una etiqueta de seguridad existente para la política de seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064). La tabla no debe ser una tabla temporal de período de sistema.

definición-período

PERIOD

Define un período para la tabla.

SYSTEM_TIME (*nombre-columna-inicio, nombre-columna-fin*)

Define un período de sistema con el nombre `SYSTEM_TIME`. No debe haber una columna en la tabla con el nombre `SYSTEM_TIME` (SQLSTATE 42711). Una tabla sólo puede tener un período `SYSTEM_TIME` (SQLSTATE 42711). *nombre-columna-inicio* debe definirse como `ROW BEGIN` y *nombre-columna-fin* debe definirse como `ROW END` (SQLSTATE 428HN).

BUSINESS_TIME (*nombre-columna-inicio, nombre-columna-fin*)

Define un período de aplicación con el nombre `BUSINESS_TIME`. No debe haber una columna en la tabla con el nombre `BUSINESS_TIME` (SQLSTATE 42711). Una tabla sólo puede tener un período `BUSINESS_TIME` (SQLSTATE 42711). *nombre-columna-inicio* y *nombre-columna-fin*, ambos, deben definirse como `DATE` o `TIMESTAMP(p)`, donde *p* es un valor comprendido entre el 0 y el 12 (SQLSTATE 42842), y las columnas deben definirse como `NOT NULL` (SQLSTATE 42831). *nombre-columna-inicio* y *nombre-columna-fin* no deben identificar una columna que se defina con una cláusula `GENERATED` (SQLSTATE 428HZ).

Una restricción de comprobación implícita se genera para garantizar que el valor de *nombre-columna-fin* sea superior que el valor de *nombre-columna-inicio*. El nombre de la restricción de comprobación

CREATE TABLE

creada implícitamente es DB2_GENERATED_CHECK_CONSTRAINT_FOR_BUSINESS_TIME, y no debe ser el nombre de ninguna otra restricción de comprobación especificada en la sentencia (SQLSTATE 42710).

restricción-unicidad

Define una restricción de clave primaria o de unicidad. Si la tabla tiene una clave de distribución, cualquier clave primaria o de unicidad debe ser un superconjunto de la clave de distribución. No puede especificarse una restricción de unicidad o de clave primaria para una tabla que sea una subtabla (SQLSTATE 429B3). Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE). Si la tabla es una tabla raíz, 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.

UNIQUE (*nombre-columna, ...*)

Define una clave exclusiva 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.

Si la tabla tiene definido un período BUSINESS_TIME, se puede especificar BUSINESS_TIME WITHOUT OVERLAPS como último elemento en la lista de expresiones clave. Si se especifica BUSINESS_TIME WITHOUT OVERLAPS, la lista debe incluir, como mínimo, un *nombre-columna*. WITHOUT OVERLAPS significa que para las demás claves especificadas, los valores son exclusivos con respecto del tiempo del período BUSINESS_TIME. Cuando se especifica BUSINESS_TIME WITHOUT OVERLAPS, las columnas del período BUSINESS_TIME no deben especificarse como parte de la restricción (SQLSTATE 428HW). La especificación de BUSINESS_TIME WITHOUT OVERLAPS añade los atributos siguientes a la restricción:

- La columna de fin del periodo de BUSINESS_TIME en orden ascendente
- La columna de inicio del periodo de BUSINESS_TIME en orden ascendente

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar como parte de una clave única ningún tipo diferenciado LOB, XML basado en uno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

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.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave exclusiva y su índice exclusivo. Se creará automáticamente un índice bidireccional exclusivo, que permite las exploraciones directas e inversas, 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 de índice entra en conflicto, el nombre será SQL seguido de una indicación de fecha y hora de caracteres (*aammddhhmmssxx*), con SYSIBM como 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 definirse como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez.

Si la tabla tiene definido un período BUSINESS_TIME, se puede especificar BUSINESS_TIME WITHOUT OVERLAPS como último elemento en la lista de expresiones clave. Si se especifica BUSINESS_TIME WITHOUT OVERLAPS, la lista debe incluir, como mínimo, un *nombre-columna*. WITHOUT OVERLAPS significa que para el resto de claves especificadas, los valores son exclusivos con respecto del tiempo del período BUSINESS_TIME. Cuando se especifica BUSINESS_TIME WITHOUT OVERLAPS, las columnas del período BUSINESS_TIME no deben especificarse como parte de la restricción (SQLSTATE 428HW). La especificación de BUSINESS_TIME WITHOUT OVERLAPS añade los atributos siguientes a la restricción:

- La columna de fin del periodo de BUSINESS_TIME en orden ascendente
- La columna de inicio del periodo de BUSINESS_TIME en orden ascendente

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar como parte de una clave primaria ningún tipo diferenciado LOB, XML basado en uno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

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 bidireccional exclusivo, que permite las exploraciones directas e inversas, 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 de índice entra en

CREATE TABLE

conflicto, el nombre será SQL seguido de una indicación de fecha y hora de caracteres (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

Si la tabla tiene una clave de distribución, las columnas de una *restricción-unicidad* deben ser un superconjunto de las columnas de clave de distribución; 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.

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 ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No puede utilizarse ningún LOB, XML o tipo diferenciado basado en uno de estos tipos, o columna de tipo estructurado, como parte de una clave foránea (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 o el apodo padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla* **o** *apodo*

La tabla o apodo que se especifica en una cláusula REFERENCES debe identificar una tabla base o un apodo que se describa en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción de referencia es un duplicado si su clave foránea, clave padre y tabla padre o apodo padre son los mismos que la clave foránea, clave padre y tabla padre o apodo padre de una restricción de referencia especificada previamente. Las restricciones de referencia duplicadas se pasan por alto y se emite un aviso (SQLSTATE 01543).

En la explicación siguiente, T2 indica la tabla padre identificada y T1 indica la tabla que está creándose (o alterándose). (T1 y T2 pueden ser la misma tabla).

La clave foránea especificada debe tener el mismo número de columnas que la tabla padre de T2 y la descripción de la columna número *n* de la clave foránea debe ser similar a la descripción de la columna número *n* de esa clave padre. Las columnas de indicación de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta norma.

(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-norma

Especifica la acción que debe realizarse en las tablas dependientes.

ON DELETE

Especifica la acción que se debe 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

La norma 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 la cláusula es una especificación implícita de ON DELETE NO ACTION.

Si T1 está conectada para supresión con T2 mediante varias vías de acceso, no se permitirá definir dos normas SET NULL con definiciones de claves foráneas solapadas. Por ejemplo: T1 (i1, i2, i3). No se permite utilizar la Norma1 con la clave foránea (i1, i2) y la Norma2 con la clave foránea (i2, i3).

El orden de aplicación de las normas es el siguiente:

1. RESTRICT
2. SET NULL OR CASCADE
3. NO ACTION

CREATE TABLE

Si dos normas distintas afectan a cualquier fila de T1, se producirá un error y no se suprimirá ninguna fila.

No se puede definir una restricción de referencia si hará que una tabla se suprima-conecte a sí misma mediante un ciclo en el que intervienen dos o más tablas y donde una de las normas de supresión es RESTRICT o SET NULL (SQLSTATE 42915).

Se puede definir una restricción de referencia que haga que una tabla se suprima-conecte a sí misma o a otra tabla mediante varias vías de acceso, excepto en los siguientes casos (SQLSTATE 42915):

- Una tabla no debe ser una tabla dependiente en una relación de tipo CASCADE (referencia a sí misma o referencia a otra tabla) y no debe tener una relación de auto referencia en la que la norma de supresión es RESTRICT o SET NULL.
- Una clave solapa otra clave cuando al menos una columna de una clave está en la misma columna que la otra clave. Cuando una tabla se suprime-conecta a otra tabla a través de varias relaciones con claves foráneas de solapamiento, estas relaciones deben tener la misma norma de supresión y ninguna de las normas de supresión puede ser SET NULL.
- Cuando una tabla está conectada por supresión a otra tabla a través de varias relaciones, y al menos una de estas relaciones se especifica con la norma de supresión SET NULL, las definiciones de clave foránea de estas relaciones no deben contener ninguna clave de distribución ni columna de clave MDC (clúster multidimensional).
- Cuando dos tablas están conectadas por supresión con la misma tabla a través de relaciones de tipo CASCADE, las dos tablas no deben estar conectadas por supresión entre sí si la norma de supresión de la última relación de cada una de las vías de acceso conectadas por supresión es RESTRICT o SET NULL.

Si alguna fila de T1 se ve afectada por dos normas de supresión distintas, el resultado sería el efecto de todas las acciones especificadas por estas normas. Los activadores AFTER y las restricciones CHECK sobre T1 también sufrirán el efecto de todas las acciones. Un ejemplo de esto es una fila que constituye el destino de anulación a través de una vía de acceso de supresión-conexión con una tabla progenitora y que es el destino de una supresión por parte de una segunda vía de acceso de supresión-conexión con la misma tabla progenitora. El resultado sería la supresión de la fila. Los activadores AFTER DELETE sobre esta tabla descendiente se activaría, pero los activadores AFTER UPDATE no.

Cuando se aplican las normas anteriores a las restricciones de referencia, en 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 debe 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 en el apartado “Notas”.

restricción-comprobación

Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o una dependencia funcional que se define entre columnas.

CONSTRAINT *nombre-restricción*

Indica el nombre de la restricción de comprobación.

CHECK (*condición-comprobación*)

Define una restricción de comprobación. La *condición-búsqueda* debe ser verdadera o desconocida para cada fila de la tabla.

condición-búsqueda

La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia de columna debe hacer referencia a una columna de la tabla que está creándose.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- La *condición-búsqueda* no puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
 - Subconsultas
 - Expresiones XMLQUERY o XMLEXISTS
 - Operaciones de eliminación de referencia o funciones Deref donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
 - Especificaciones CAST con una cláusula SCOPE
 - Funciones de columna
 - Funciones que no sean deterministas
 - Funciones definidas para que exista una acción externa
 - Funciones definidas por el usuario definidas con MODIFIES SQL o READS SQL DATA
 - Variables del lenguaje principal
 - Marcadores de parámetro
 - *referencias-secuencia*
 - Especificaciones OLAP
 - Registros especiales y funciones incorporadas que dependen del valor de un registro especial
 - Variables globales
 - Referencias a columnas generadas que no correspondan a la columna de identidad
 - Referencias a columnas de tipo XML (excepto en un predicado VALIDATED)
 - Una *expresión-tabla-anidada* tolerante a errores

dependencia-funcional

Define una dependencia funcional entre columnas.

nombre-columna **DETERMINED BY** *nombre-columna* o
 (*nombre-columna*,...) **DETERMINED BY** (*nombre-columna*,...)

El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que

CREATE TABLE

siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la columna no puede ser un tipo de datos LOB, un tipo diferenciado basado en un tipo de datos LOB, un tipo de datos XML ni un tipo estructurado (SQLSTATE 42962). No puede utilizarse una columna ROW CHANGE TIMESTAMP como parte de una clave primaria (SQLSTATE 429BV). Ninguna columna del conjunto de columnas hijo puede ser una columna con posibilidad de nulos (SQLSTATE 42621).

Si se especifica una restricción de comprobación como parte de una *definición-columna*, sólo puede establecerse una referencia de columna que haga referencia a 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. No se comprueba si hay incoherencias, condiciones duplicadas ni condiciones equivalentes en las restricciones de comprobación. Por lo tanto, se pueden definir restricciones de comprobación contradictorias o redundantes, lo que podría dar lugar a posibles errores en tiempo de ejecución.

Se puede especificar la *condición-búsqueda* "IS NOT NULL"; sin embargo, se recomienda que la anulabilidad se fuerce directamente, utilizando el atributo NOT NULL de una columna. Por ejemplo, CHECK (salario + bonificación > 30000) se acepta si el salario se ha establecido en NULL, ya que las restricciones CHECK deben satisfacerse o bien no conocerse y, en este caso, el salario no se conoce. Sin embargo, se consideraría que CHECK (salario IS NOT NULL) es falso y una violación de la restricción si el salario se ha establecido en NULL.

Las restricciones de comprobación con *condición-búsqueda* se fuerzan cuando se insertan filas en la tabla o se actualizan. Una restricción de comprobación definida en una tabla se aplica automáticamente a todas las subtablas de esta tabla.

El gestor de bases de datos no fuerza una dependencia funcional durante las operaciones normales como, por ejemplo, insertar, actualizar, suprimir o establecer integridad. La dependencia funcional puede utilizarse durante la regrabación de consultas para optimizarlas. Se pueden devolver resultados incorrectos si no se mantiene la integridad de una dependencia funcional.

atributos-restricción

Define los atributos que se asocian a las restricciones de comprobación o de integridad referencial.

ENFORCED o NOT ENFORCED

Especifica si el gestor de bases de datos obliga a aplicar la restricción durante operaciones normales como, por ejemplo, insertar, actualizar o suprimir. El valor por omisión es ENFORCED.

ENFORCED

El gestor de bases de datos obliga a aplicar la restricción. No se puede especificar ENFORCED para una dependencia funcional (SQLSTATE 42621). ENFORCED no se puede especificar cuando una restricción de referencia hace referencia a un apodo (SQLSTATE 428G7).

NOT ENFORCED

El gestor de bases de datos no obliga a aplicar la restricción.

TRUSTED

Se puede confiar en los datos para que se ajusten a la restricción. TRUSTED sólo debe utilizarse si se reconoce de forma independiente que los datos de la tabla se ajustan a la restricción. Los resultados de la consulta pueden ser imprevisibles si los datos no se ajustan realmente a la restricción. Es la opción por omisión.

Las restricciones informativas no deben incumplirse en ningún momento. Las restricciones informativas se utilizan en la optimización de consultas y en el proceso incremental de las MQT REFRESH IMMEDIATE y las tablas de etapas. Estos procesos podrían generar resultados imprevisibles o contenido de tablas de etapas o MQT incorrecto si se incumplen las restricciones. Por ejemplo, el orden en el que se mantienen las tablas padre-hijo es importante. Si se quieren añadir filas a una tabla padre-hijo, primero debe insertar las filas en la tabla padre. Para eliminar filas de una tabla padre-hijo, primero debe suprimir las filas de la tabla hijo. De esta forma se garantiza que no queden filas huérfanas en la tabla hijo en ningún momento. Si se incumplen las restricciones informativas, el mantenimiento incremental de los datos de tablas de etapas y de los datos de MQT dependientes podría optimizarse en función de las restricciones informativas incumplidas, lo que produciría datos incorrectos.

NOT TRUSTED

No se puede confiar en los datos para que se ajusten a la restricción. NOT TRUSTED está pensado para casos en los que los datos se ajustan a la restricción para la mayor parte de las filas, pero no se reconoce de forma independiente que todas las filas o las futuras adiciones se ajustarán a la restricción. Si una restricción es NOT TRUSTED y se ha habilitado para la optimización de consultas, no se utilizará para realizar optimizaciones que dependan de los datos que se ajusten por completo a la restricción. Sólo se puede especificar NOT TRUSTED para las restricciones de integridad referenciales (SQLSTATE 42613).

ENABLE QUERY OPTIMIZATION o DISABLE QUERY OPTIMIZATION

Especifica si se puede utilizar la restricción o la dependencia funcional para la optimización de la consulta bajo circunstancias adecuadas. El valor por omisión es ENABLE QUERY OPTIMIZATION.

ENABLE QUERY OPTIMIZATION

La restricción se supone que es verdadera y se puede utilizar para la optimización de consulta.

DISABLE QUERY OPTIMIZATION

No se puede utilizar la restricción para la optimización de consulta.

OF *nombre1-tipo*

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 dinámico). El nombre de tipo debe ser el nombre de

CREATE TABLE

un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se pueda 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 *definición-columna-OID*). Esta columna de identificador de objeto es la primera columna de la tabla. La columna de 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 tablas. Se crea a la vez que la tabla raíz de la jerarquía. Los datos para todas las subtablas en 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 alias 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. Este nombre está compuesto por el nombre de la tabla que se crea, seguido de un sufijo exclusivo, de tal modo que el identificador será exclusivo entre los identificadores de las tablas, vistas 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).

Las columnas de la tabla incluyen 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, la captura de datos, **NOT LOGGED INITIALLY** y la clave de distribución. 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-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-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).

LIKE *nombre1-tabla* **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 (*nombre-tabla-1*), vista (*nombre-vista*) o apodo (*apodo*) identificado. 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 identificada (incluidas las columnas implícitamente ocultas), vista o apodo. Una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla existente también se definirá como implícitamente oculta. La definición implícita depende de lo que se identifica después de LIKE:

- Si se identifica una tabla, la definición implícita incluye el nombre de columna, tipo de datos, atributo oculto y la característica de poder contener nulos de cada una de las columnas de *nombre1-tabla*. 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

CREATE TABLE

resultante de la selección completa definida en *nombre-vista*. Los tipos de datos de las columnas de la vista deben ser tipos de datos válidos para las columnas de una tabla.

- Si se designa un apodo, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *apodo*.
- Si se identifica una tabla protegida en la cláusula LIKE, la tabla nueva hereda la misma política de seguridad y las mismas columnas protegidas que la tabla identificada.
- Si una tabla se identifica en la cláusula LIKE y la tabla contiene una columna de inicio de fila, columna de fin de fila o columna de ID de inicio de transacción, la columna correspondiente de la nueva tabla sólo hereda el tipo de datos de la columna fuente. La nueva columna no se considera una columna generada.
- Cuando una tabla que incluye un período se identifica en la cláusula LIKE, la nueva tabla no hereda la definición de período.
- Cuando se identifica una tabla temporal de período de sistema en la cláusula LIKE, la nueva tabla no es una tabla temporal de período de sistema.

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, activadores, especificación ORGANIZE BY o especificación PARTITIONING KEY. 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.

Cuando una tabla se identifica en la cláusula LIKE y dicha tabla contiene una columna ROW CHANGE TIMESTAMP, la columna correspondiente de la nueva tabla hereda sólo el tipo de datos de la columna ROW CHANGE TIMESTAMP. La nueva columna no se considera una columna generada.

Si el control de acceso de nivel de fila o de columna está activado para *nombre-tabla-1*, la nueva tabla no lo hereda.

opciones-copia

Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados fuente (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 de resultados fuente. 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* identifica una tabla base, una tabla temporal creada o una tabla temporal declarada, INCLUDING COLUMN DEFAULTS es el valor por omisión. Si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica un apodo, INCLUDING COLUMN DEFAULTS no tendrá ningún efecto y no se copiarán los valores por omisión de la columna.

EXCLUDING COLUMN DEFAULTS

Especifica que no deben copiarse los valores por omisión de las columnas de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica una tabla base, una tabla temporal creada o una tabla temporal declarada, entonces INCLUDING COLUMN DEFAULTS es el valor por omisión.

INCLUDING IDENTITY COLUMN ATTRIBUTES

Los atributos de columna de identidad se copian desde la definición de la tabla de resultados fuente, si es posible. 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 de resultados fuente.

tabla-resultados-as

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 de resultados 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 de resultados de la *selección completa*.

Debe especificarse una lista de nombres de columna si la tabla de resultados 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 Especifica la consulta que se utiliza para la definición de 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. Una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla base a la que se hace referencia en la *selección completa* no se considera oculta en la nueva tabla.

Todos los elementos de la lista de selección deben tener un nombre (utilice la cláusula AS para las expresiones). La *tabla-resultados-as* define atributos de la tabla. Los tipos de datos de las columnas de resultados deben ser tipos de datos válidos para las columnas de una tabla.

CREATE TABLE

La *selección completa* no puede incluir una cláusula *referencia-tabla-cambio-datos* (SQLSTATE 428FL).

Se puede especificar cualquier *selección completa* válida que no haga referencia a una tabla con tipo ni a una vista con tipo.

Si el control de acceso de nivel de columna se activa para las tablas especificadas en *selección_completa*, no se propaga en cascada a la nueva tabla.

WITH NO DATA

La consulta sólo se utiliza para definir la tabla. La tabla no se rellena con los resultados de la consulta.

Las columnas de la tabla se definen basándose en las definiciones de las columnas producidas por 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, el tipo de datos y la posibilidad de contener nulos de la tabla referenciada.

definición-consulta-materializada

nombre-columna

Designa las columnas de la tabla. Si se especifica una lista de nombres de columna, esta debe constar de tantos nombres como columnas haya en la tabla de resultados 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 de resultados de la selección completa.

Se debe especificar una lista de nombres de columna si la tabla de resultados 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 Especifica la consulta que se utiliza para la definición de la tabla y que determina los datos que se deben incluir 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. No se considera oculta en la tabla nueva una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla base a la que se hace referencia en la selección completa.

Todos los elementos de la lista de selección deben tener un nombre (utilice la cláusula AS para las expresiones). La *definición-consulta-materializada* define los atributos de la tabla de consulta materializada. La opción elegida también define el contenido de la selección completa de la manera siguiente.

La selección completa no puede incluir una cláusula *referencia-tabla-cambio-datos* (SQLSTATE 428FL).

Cuando se especifica REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa no puede incluir (SQLSTATE 428EC):

- Las referencias a una tabla de consulta materializada, una tabla temporal creada, una tabla temporal declarada o una tabla con tipo en cualquier cláusula FROM

- Referencias a una vista donde la selección completa de la vista infrinja cualquiera de las restricciones que aparecen en la selección completa de la tabla de consulta materializada
- Expresiones que sean un tipo de referencia (o un tipo diferenciado basado en este tipo)
- Funciones que tengan cualquiera de los atributos siguientes:
 - EXTERNAL ACTION
 - LANGUAGE SQL
 - CONTAINS SQL
 - READS SQL DATA
 - MODIFIES SQL DATA
- Funciones NOT SECURED, si las funciones hacen referencia a una tabla de consulta materializada que a su vez hace referencia a una tabla que tiene activado el control de acceso a columnas o filas.
- Funciones que dependan de características físicas (por ejemplo, DBPARTITIONNUM, HASHEDVALUE, RID_BIT, RID)
- Una expresión ROW CHANGE o referencia a una columna ROW CHANGE TIMESTAMP de la fila
- Referencias de tabla o vista a objetos del sistema (tampoco se deben especificar tablas de Explain)
- Expresiones que sean un tipo estructurado, un tipo LOB (o un tipo diferenciado basado en un tipo LOB) o un tipo XML
- Referencias a una tabla protegida o un apodo protegido

Cuando se especifica DISTRIBUTE BY REPLICATION, se aplican las siguientes restricciones:

- No está permitida la cláusula GROUP BY.
- La tabla de consulta materializada sólo debe hacer referencia a una única tabla; es decir, no puede incluir una unión.

Si se especifica REFRESH IMMEDIATE:

- La consulta debe ser una subselección, con la excepción de que se da soporte a UNION ALL en la expresión de tabla de entrada de GROUP BY.
- La consulta no puede ser recursiva.
- La consulta no puede incluir:
 - Referencias a un apodo
 - Funciones que no sean deterministas
 - Selecciones completas escalares
 - Predicados con selecciones completas
 - Registros especiales y funciones incorporadas que dependen del valor de un registro especial
 - Variables globales
 - SELECT DISTINCT
 - Una *expresión-tabla-anidada* tolerante a errores
- 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.
- Cuando se especifica una cláusula GROUP BY, se aplican las siguientes consideraciones:

CREATE TABLE

- Las funciones de columna que reciben soporte son SUM, COUNT, COUNT_BIG y GROUPING (sin DISTINCT). La lista de selección debe contener una columna COUNT(*) o COUNT_BIG(*). Si la lista de selección de la tabla de consulta materializada contiene SUM(X), donde la X es un argumento que puede ser nulo, la tabla de consulta materializada también debe tener COUNT(X) en su lista de selección. Estas funciones de columna no pueden formar parte de ninguna expresión.
- No está permitida una cláusula HAVING.
- Si se utiliza en un grupo de particiones de base de datos con varias particiones, la clave de distribución debe ser un subconjunto de los elementos de GROUP BY.
- La tabla de consulta materializada no debe contener filas duplicadas y se aplicarán las siguientes restricciones específicas de este requisito de exclusividad, según se especifique la cláusula GROUP BY o no.
 - Cuando se especifica una cláusula GROUP BY, se aplican las siguientes restricciones relacionadas con la exclusividad:
 - Todos los elementos de GROUP BY se deben incluir en la lista de selección.
 - Cuando GROUP BY contiene GROUPING SETS, CUBE o ROLLUP, los elementos de GROUP BY y las funciones de columna GROUPING de la lista de selección deben formar una clave exclusiva del conjunto de resultados. Por lo tanto, deberán satisfacerse las restricciones siguientes:
 - No se pueden repetir conjuntos de agrupación. Por ejemplo, ROLLUP(X, Y), X no está permitido, porque es equivalente a GROUPING SETS((X, Y), (X), (X)).
 - Si X es un elemento de GROUP BY con posibilidad de nulos que aparece en GROUPING SETS, CUBE o ROLLUP, GROUPING(X) deberá aparecer en la lista de selección.
 - Cuando no se especifica la cláusula GROUP BY, se aplican las siguientes restricciones relacionadas con la exclusividad:
 - El requisito de exclusividad de la tabla de consulta materializada se consigue por derivación de una clave exclusiva para la vista materializada a partir de una de las restricciones de clave exclusiva definidas en cada una de las tablas subyacentes. Por lo tanto, las tablas subyacentes deben tener definida como mínimo una restricción de clave exclusiva, y las columnas de estas claves deben aparecer en la lista de selección de la definición de tabla de consulta materializada.

Cuando se especifica REFRESH DEFERRED:

- Si la tabla de consulta materializada se crea con intención de proporcionarle una tabla de etapas asociada en una sentencia posterior, la selección completa de la tabla de consulta materializada debe seguir las mismas restricciones y normas que una selección completa utilizada para crear una tabla de consulta materializada con la opción REFRESH IMMEDIATE.
- Si la consulta es recursiva, la tabla de consulta materializada no se utiliza para optimizar el proceso de las consultas.
- La tabla de consulta materializada no se utiliza para optimizar el proceso de consultas estáticas.

Una tabla de consulta materializada cuya selección completa contiene una cláusula GROUP BY, obtiene el resumen de los datos de las tablas a las que se hace referencia en la selección completa. Este tipo de tabla de consulta materializada también se denomina *tabla de resumen*. Una tabla de resumen en un tipo especializado de tabla de consulta materializada.

Si la *selección_completa* hace referencia a una tabla o una vista que depende de una tabla para la que se ha activado el control de acceso de nivel de fila o columna, esos controles de acceso de nivel de fila o columna se ignorarán al llenar la tabla de consulta materializada. La tabla de consulta materializada se crea de forma automática con el control de acceso de nivel de fila activado. Si los usuarios finales acceden directamente a esta tabla no verán ningún contenido, a menos que se creen los permisos adecuados o un usuario con autorización SECADM decida desactivar el control de acceso de nivel de fila en esta tabla de consulta materializada. Tenga en cuenta que el control de acceso de nivel de fila y columna en la tabla de consulta materializada no afecta al direccionamiento interno hacia la tabla de consultas materializadas por parte del compilador de SQL de DB2.

opciones-tabla-renovable

Defina las opciones que pueden renovarse de los atributos de tabla de consulta materializada.

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 en forma de instantánea en el momento en que se procesa la sentencia REFRESH TABLE. Las tablas de consulta materializada mantenidas por el sistema que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807). Las tablas de consulta materializada mantenidas por el usuario que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE.

IMMEDIATE

Los cambios que se han realizado en las tablas subyacentes como parte de una sentencia DELETE, INSERT o UPDATE se aplican en cascada a la tabla de consulta materializada. En este caso, el contenido de la tabla, en cualquier momento, es igual que cuando se procesa la *subselección* especificada. Las tablas de consulta materializada que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION

La tabla de consulta materializada puede utilizarse para la optimización de la consulta cuando se dan las circunstancias adecuadas.

DISABLE QUERY OPTIMIZATION

La tabla de consulta materializada no se utilizará para la optimización de la consulta. La tabla todavía puede consultarse directamente.

CREATE TABLE

MAINTAINED BY

Especifica quién mantiene los datos de la tabla de consulta materializada: el sistema, el usuario o una herramienta de duplicación. El valor por omisión es SYSTEM.

SYSTEM

Especifica que el sistema mantiene los datos de la tabla de consulta materializada.

USER

Especifica que el usuario mantiene los datos de la tabla de consulta materializada. El usuario puede realizar operaciones de actualización, supresión o inserción en las tablas de consulta materializada mantenidas por el usuario. La sentencia REFRESH TABLE, que se utiliza para las tablas de consulta materializada mantenidas por el sistema, no puede invocarse para las tablas de consulta materializada mantenidas por el usuario. Sólo una tabla de consulta materializada REFRESH DEFERRED puede definirse como MAINTAINED BY USER.

FEDERATED_TOOL

Especifica que la herramienta de duplicación mantiene los datos de la tabla de consulta materializada. La sentencia REFRESH TABLE, que se utiliza para las tablas de consulta materializada mantenidas por el sistema, no se puede invocar para las tablas de consulta materializada mantenidas por herramientas federadas. Sólo se puede definir una única tabla de consulta materializada REFRESH DEFERRED como MAINTAINED BY FEDERATED_TOOL.

Cuando se especifica esta opción, la cláusula select de la sentencia CREATE TABLE no puede contener una referencia a una tabla base (SQLSTATE 428EC).

definición-tabla-etapas

Define la consulta que recibe el soporte de la tabla de etapas indirectamente por medio de una tabla de consulta materializada asociada. Las tablas subyacentes de la tabla de consulta materializada también son las tablas subyacentes de su tabla de etapas asociada. La tabla de etapas recopila los cambios que deben aplicarse a la tabla de consulta materializada para sincronizarla con el contenido de las tablas subyacentes.

Si la *selección_completa* hace referencia a una tabla o una vista que depende de una tabla para la que se ha activado el control de acceso de nivel de fila o columna, esos controles de acceso de nivel de fila o columna se ignorarán al llenar la tabla de etapas. No obstante, la tabla de etapas se crea de forma automática con el control de acceso de nivel de fila activado. Si los usuarios finales acceden directamente a esta tabla de etapas no verán ningún contenido, a menos que se creen los permisos adecuados o un usuario con autorización SECADM decida desactivar el control de acceso de nivel de fila en esta tabla de etapas. Tenga en cuenta que el control de acceso de nivel de fila y columna en la tabla de etapas no afecta al proceso interno de aplicación de los cambios capturados por la tabla de etapas en la tabla de consulta materializada asociada.

nombre-columna-etapas

Indica los nombres de las columnas de la tabla de etapas. Si se especifica una lista de nombres de columna, ésta deberá estar compuesta de *dos* nombres más que columnas hay en la tabla de consulta materializada para la que define la tabla. Si la tabla de consultas materializadas es una tabla de

consulta materializada duplicada, o si la consulta que define la tabla de consulta materializada no contiene una cláusula GROUP BY, la lista de nombres de columna deberá estar compuesta de *tres* nombres más que columnas hay en la tabla de consulta materializada para la que se define la tabla de etapas. Cada nombre de columna debe ser exclusivo y no calificado. Si no se especifica una lista de nombres de columna, las columnas de la tabla heredan los nombres de las columnas de la tabla de consulta materializada asociada. Las columnas adicionales se denominan GLOBALTRANSID y GLOBALTRANSTIME y, si se necesita una tercera columna, ésta se denomina OPERATIONTYPE.

Tabla 20. Columnas adicionales que se añaden a las tablas de etapas

Nombre de columna	Tipo de datos	Descripción de la columna
GLOBALTRANSID	CHAR(8) FOR BIT DATA	El ID de transacción global de cada fila propagada
GLOBALTRANSTIME	CHAR(13) FOR BIT DATA	La indicación de fecha y hora de la transacción
OPERATIONTYPE	INTEGER	Operación para la fila propagada, que puede ser una inserción, una actualización o una supresión.

Deberá especificarse una lista de nombres de columnas si cualquiera de las columnas de la tabla de consulta materializada asociada duplica cualquiera de los nombres de columna generados (SQLSTATE 42711).

FOR *nombre2-tabla*

Especifica la tabla de consulta materializada que se utiliza para la definición de la tabla de etapas. El nombre, incluido el esquema implícito o explícito, debe identificar una tabla de consulta materializada que exista en el servidor actual definido con REFRESH DEFERRED. La selección completa de la tabla de consulta materializada asociada debe seguir las mismas restricciones y normas que una selección completa que se ha utilizado para crear una tabla de consulta materializada con la opción REFRESH IMMEDIATE.

El contenido de la tabla de etapas puede utilizarse para renovar la tabla de consulta materializada, invocando la sentencia REFRESH TABLE, si el contenido de la tabla de etapas es coherente con la tabla de consulta materializada asociada y las tablas fuente subyacentes.

PROPAGATE IMMEDIATE

Los cambios que se han realizado en las tablas subyacentes como parte de una operación de supresión, inserción o actualización, se aplican en cascada a la tabla de etapas en la misma operación de supresión, inserción o actualización. Si la tabla de etapas no se ha marcado como incoherente, su contenido, en cualquier momento, son los cambios delta para la tabla subyacente desde la última renovación de la tabla de consulta materializada.

ORGANIZE BY DIMENSIONS (*nombre-columna,...*)

Especifica una dimensión para cada columna o grupo de columnas que se utiliza para el establecimiento de los datos de tabla en un clúster. La utilización de paréntesis dentro de la lista de dimensiones especifica que un grupo de columnas va a tratarse como una dimensión. La palabra clave DIMENSIONS

CREATE TABLE

es opcional. La tabla cuya definición especifica esta cláusula se conoce como tabla de clúster multidimensional (MDC).

Se mantiene automáticamente un índice de bloques de clúster para cada dimensión especificada y se mantiene un índice de bloques, compuesto de todas las columnas que se utilizan en la cláusula, si ninguno de los índices de bloques de clúster las incluyen. El conjunto de columnas que se utiliza en la cláusula ORGANIZE BY debe respetar las normas de la sentencia CREATE INDEX que especifica CLUSTER.

Cada nombre de columna especificado en la cláusula ORGANIZE BY debe estar definido para la tabla (SQLSTATE 42703). Una dimensión no puede aparecer más de una vez en la lista de dimensiones (SQLSTATE 42709). Las dimensiones no pueden contener una columna ROW CHANGE TIMESTAMP, una columna de inicio de fila, una columna de fin de fila, una columna de ID de inicio de transacción (SQLSTATE 429BV) o una columna XML (SQLSTATE 42962).

Las páginas de la tabla se organizan en bloques de igual tamaño, que es el tamaño de extensión del espacio de tablas, y todas las filas de cada bloque contienen la misma combinación de valores de dimensión.

Una tabla puede ser una tabla de clúster multidimensional (MDC) y una tabla particionada. Las columnas de una tabla de esta clase se pueden emplear tanto en la *espec-partición-rangos* como en la clave MDC. Tenga presente que el particionamiento de tablas es multicolumna, no multidimensional.

Para una tabla MDC particionada creada en DB2 Versión 9.7 Fixpack 1 o releases posteriores, los índices de bloque se particionarán. La ubicación del índice de bloque particionado sigue la norma de ubicación de almacenamiento de índice particionado general. Todas las particiones de índice de una partición de datos determinada, incluidos los índices de bloque MDC, comparten un único objeto de índice. Por omisión, las particiones de índice de cada partición de datos específica residen en el mismo espacio de tablas que la partición de datos. Esto puede alterarse temporalmente con la cláusula INDEX IN de nivel de partición.

Para las tablas MDC creadas en DB2 V9.7 o releases anteriores, los índices de bloque no se particionarán y seguirán sin particionarse si vuelven a crearse. Las tablas MDC con índices de bloque particionados pueden coexistir en la misma base de datos que las tablas MDC con índices de bloque no particionados. Para cambiar los índices de bloque no particionados por índices de bloque particionados, utilice un movimiento de tabla en línea para migrar la tabla MDC.

ORGANIZE BY KEY SEQUENCE *espec-clave-secuencia*

Especifica que la tabla está organizada en secuencia de clave ascendente, con un tamaño fijo basado en el rango especificado de los valores de secuencia de clave. Una tabla organizada de este modo recibe el nombre de *tabla agrupada por clústeres de rangos*. Cada posible valor de clave del rango definido tiene una ubicación por omisión en la tabla física. El almacenamiento necesario para una tabla agrupada por clústeres de rangos debe estar disponible cuando se crea la tabla, y debe ser suficiente para contener el número de filas del rango especificado multiplicado por el tamaño de la fila (para obtener más información sobre cómo determinar los requisitos de espacio, consulte los apartados Límite del tamaño de fila y Número total de bytes).

nombre-columna

Especifica una columna de la tabla que se incluye en la clave exclusiva que determina la secuencia de la tabla agrupada por clústeres de rangos. El

tipo de datos de la columna debe ser SMALLINT, INTEGER o BIGINT (SQLSTATE 42611) y las columnas deben estar definidas como NOT NULL (SQLSTATE 42831). La misma columna no puede estar identificada más de una vez e la clave de secuencia. El número de columnas identificadas no debe ser superior a 64 (SQLSTATE 54008).

Se creará una entrada de índice exclusiva automáticamente en el catálogo para las columnas de la secuencia de clave especificadas con orden ascendente para cada columna. El nombre del índice será SQL, seguido de una indicación de fecha y hora de caracteres (*aaammddhhmmssxxx*), con SYSIBM como nombre de esquema. Un objeto de índice real no se crea en almacenamiento, porque la organización de la tabla se ordena por esta clave. Si se define una clave principal o una restricción exclusiva en las mismas columnas que la clave de secuencia de tabla agrupada por clústeres de rangos, se utiliza esta misma entrada de índice para la restricción.

Para la especificación de secuencia de clave, existe una restricción de comprobación para reflejar las restricciones de columna. Si se especifica la cláusula DISALLOW OVERFLOW, el nombre de la restricción de comprobación será RCT y se obliga el cumplimiento con la restricción de comprobación. Si se especifica la cláusula ALLOW OVERFLOW, el nombre de la restricción de comprobación será RCT_OFLOW y no se aplica la restricción de comprobación.

STARTING FROM *constante*

Especifica el valor constante en el extremo inferior del rango para *nombre-columna*. Los valores menores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821). Si no se especifica ninguna constante inicial, el valor por omisión es 1.

ENDING AT *constante*

Especifica el valor constante en el extremo superior del rango para *nombre-columna*. Los valores mayores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. El valor de la constante final debe ser mayor que la constante inicial. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821).

ALLOW OVERFLOW

Especifica que la tabla agrupada por clústeres de rangos permite filas con valores de clave que quedan fuera del rango de valores definido. Cuando se crea una tabla agrupada por clústeres de rangos que permite desbordamientos, las filas con valores de clave que quedan fuera del rango se colocan al final del rango definido sin ningún orden por omisión. Las operaciones en que intervienen estas filas de desbordamiento son menos eficaces que las operaciones en filas que tienen valores de clave dentro del rango definido.

DISALLOW OVERFLOW

Especifica que la tabla agrupada por clústeres de rangos no permite filas con valores de clave que no quedan dentro del rango de valores definido (SQLSTATE 23513). Las tablas agrupadas por clústeres de rangos que no permiten desbordamientos siempre mantendrán todas las filas en secuencia de clave ascendente.

CREATE TABLE

La cláusula `DISALLOW OVERFLOW` no puede especificarse si la tabla es una tabla de consulta materializada agrupada por clústeres de rangos (SQLSTATE 429BG).

PCTFREE *entero*

Especifica el porcentaje de cada página que debe dejarse como espacio libre. La primera fila de cada página se añade sin restricciones. Cuando se añaden filas adicionales a una página, al menos *entero* por ciento de la página se deja como espacio libre. El valor de *entero* entra en un rango que va de 0 a 99. El valor -1 de `PCTFREE` en el catálogo del sistema (`SYSCAT.TABLES`) se interpreta como el valor por omisión. El valor de `PCTFREE` por omisión para una página de tabla es 0.

ORGANIZE BY INSERT TIME

Especifica que las filas se agrupan en clústeres en la tabla en relación con el tiempo en que se insertan. Las filas se insertan en el extremo lógico del objeto de tabla en lugar de buscar espacio disponible. Una tabla que se organiza por tiempo de inserción recibe el nombre de tabla de agrupación en clústeres de tiempo de inserción (ITC). Este tipo de tabla puede utilizar `REORG TABLE RECLAIM EXTENTS` para reclamar extensiones libres con el fin de que puedan utilizar éstos inmediatamente otros objetos en el espacio de tablas.

Los datos se agrupan en clústeres mediante una dimensión virtual implícitamente creada. Un índice de bloques de agrupación en clústeres se mantiene automáticamente para esta dimensión virtual. La dimensión virtual no puede manipularse y no consume espacio por cada fila que existe en la tabla. Las páginas de la tabla se organizan en bloques de igual tamaño, que es el tamaño de extensión del espacio de tablas.

La cláusula `ORGANIZE BY INSERT TIME` no puede especificarse si la tabla es una tabla con tipo (SQLSTATE 428DH).

DATA CAPTURE

Indica si se debe registrar en el archivo de anotaciones cronológicas información adicional para la duplicación de datos entre bases de datos. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 428DR).

Si la cláusula no se especifica y la tabla no es una tabla con tipo, el valor por omisión se determina mediante el ajuste `DATA CAPTURE` del esquema en el tiempo en que se crea la tabla.

NONE

Indica que no se va a anotar ninguna información adicional.

CHANGES

Indica que en el archivo de anotaciones cronológicas se registrará 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 el nombre de esquema (implícito o explícito) de la tabla tiene más de 18 bytes, esta opción no recibe soporte (SQLSTATE 42997).

Si la tabla es una tabla con tipo que no es adecuada, entonces esta opción no se soporta (SQLSTATE 428DH).

IN *nombre-espacio-tablas,...*

Identifica los espacios de tablas en que se va a crear la tabla. Los espacios de tablas deben existir, deben encontrarse en el mismo grupo de particiones de

base de datos y deben ser todos espacios de tablas DMS normales o todos DMS grandes o todos SMS (SQLSTATE 42838) en los que el ID de autorización de la sentencia posee el privilegio USE.

Se permite como máximo una cláusula IN en el nivel de tabla. Todos los espacios de tablas que utiliza una tabla deben tener el mismo tamaño de página y el mismo tamaño de extensión.

Si tan solo se especifica un espacio de tablas, todas las partes de la tabla se almacenan 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 gestor de bases de datos elige un espacio de tablas (desde el conjunto de espacios de tablas existentes en la base de datos) con el tamaño de página menor posible sobre el que el ID de autorización de la sentencia tenga el privilegio USE.

Si es apto más de un espacio de tablas, elija el espacio de tablas de acuerdo con el orden de preferencia siguiente, según cómo el ID de autorización de la sentencia haya recibido el privilegio USE sobre el espacio de tablas:

1. El ID de autorización
2. Un rol al que se otorga el ID de autorización
3. Un grupo al que pertenezca el ID de autorización
4. Un rol al que se otorga el ID de autorización que pertenece al grupo
5. PUBLIC
6. Un rol al que se otorga 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 si:

- Se descartan o crean espacios de tablas
- Se otorgan o revocan los privilegios USE

Las tablas particionadas pueden tener las particiones de datos repartidas por varios espacios de tablas. Cuando se especifican varios espacios de tablas, todos los espacios de tablas deben existir y todos deben ser espacios de tablas SMS, DMS normales o DMS grandes (SQLSTATE 42838). El ID de autorización de la sentencia debe poseer el privilegio USE para todos los espacios de tablas especificados.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Para obtener más información, consulte el apartado Límite del tamaño de fila.

Si una tabla se coloca en un espacio de tablas grande:

- La tabla puede ser mayor que una tabla de un espacio de tablas normal. Para obtener detalles sobre los límites de espacio de tablas y de tabla, consulte el apartado “Límites de SQL”.
- La tabla puede dar soporte a más de 255 filas por página de datos, lo que puede mejorar la utilización del espacio de las páginas de datos.
- Los índices definidos en la tabla requerirán una entrada de dos bytes por fila adicional, a diferencia de los índices definidos en una tabla que reside en un espacio de tablas normal.

CREATE TABLE

CYCLE o NO CYCLE

Especifica si el número de particiones de datos sin ningún espacio de tablas explícito puede exceder el número de espacios de tablas especificados.

CYCLE

Especifica que, si el número de particiones de datos sin ningún espacio de tablas explícito excede el número de espacios de tablas especificados, los espacios de tablas se asignan a las particiones de datos de modo rotativo.

NO CYCLE

Especifica que el número de particiones de datos sin ningún espacio de tablas explícito no puede exceder el número de espacios de tablas especificados (SQLSTATE 428G1). Esta opción evita la asignación rotativa de los espacios de tablas a las particiones de datos.

opciones-espaciotablas

Especifica el espacio tablas en que se almacenarán los valores de los índices o las columnas largas. Para obtener detalles sobre los tipos de espacios de tablas, consulte "CREATE TABLESPACE".

INDEX IN nombre-espacio-tablas

Identifica el espacio de tablas donde se crean los índices de una tabla no particionada o los índices no particionados de una tabla particionada. El espacio de tablas especificado debe existir; debe ser un espacio de tablas DMS si la tabla tiene datos en espacios de tablas DMS o un espacio de tablas SMS si la tabla particionada tiene datos en espacios de tablas SMS; debe ser un espacio de tablas en que el ID de autorización de la sentencia posea el privilegio USE y debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tablas* (SQLSTATE 42838).

Se puede especificar el espacio de tablas que contendrá los índices cuando se crea una tabla o, en el caso de tablas particionadas, especificando la cláusula IN de la sentencia CREATE INDEX para un índice no particionado. La comprobación del privilegio USE para el espacio de tablas se realiza al crear la tabla, no al crear posteriormente un índice.

En el caso de un índice no particionado en una tabla particionada, el almacenamiento del índice se realiza como se indica a continuación:

- El espacio de tablas mediante la cláusula IN de la sentencia CREATE INDEX
- El espacio de tablas de nivel de tabla especificado para la cláusula INDEX IN de la sentencia CREATE TABLE
- Si no se especifica ninguno de los anteriores, el índice se almacena en el espacio de tablas de la primera partición de datos enlazada o visible

Para obtener información acerca de los índices particionados en tablas particionadas, consulte la descripción de la cláusula INDEX IN de nivel de elemento-partición.

LONG IN nombre-espacio-tablas

Identifica los espacios de tablas donde se almacenarán los valores de todas las columnas largas. Las columnas largas incluyen aquellas con tipos de datos LOB, tipo XML o tipos diferenciados con cualquiera de estos como tipos de fuente, o las columnas que se han definido con

tipos estructurados definidos por el usuario cuyos valores no se pueden almacenar en línea. Esta opción sólo está permitida si la cláusula IN identifica un espacio de tablas DMS.

El espacio de tablas especificado debe existir. Puede ser un espacio de tablas normal si es el mismo espacio de tablas en el que están almacenados los datos; de lo contrario, debe ser un espacio de tablas DMS grande en el que el ID de autorización de la sentencia tenga el privilegio USE. También debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tablas* (SQLSTATE 42838).

Se puede especificar qué espacio de tablas contendrá las columnas largas, LOB o XML al crear una tabla. La comprobación del privilegio USE se realiza al crear la tabla, no al añadir posteriormente una columna larga o LOB.

Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

cláusula-distribución

Especifica el particionamiento de base de datos o el modo en que se distribuyen los datos en diversas particiones de base de datos.

DISTRIBUTE BY HASH (*nombre-columna,...*)

Especifica la utilización de la función de hash por omisión en las columnas especificadas (denominada *clave de distribución*) como método de distribución en las particiones de base de datos. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No se puede utilizar como parte de una clave de distribución ninguna columna cuyo tipo de datos sea BLOB, CLOB, DBCLOB, XML, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado (SQLSTATE 42962). La clave de distribución no puede contener una columna ROW CHANGE TIMESTAMP (SQLSTATE 429BV). No puede especificarse una clave de distribución para una tabla que sea una subtabla (SQLSTATE 42613) porque la clave de distribución se hereda de la tabla raíz de la jerarquía de tablas o una tabla con una columna que tenga el tipo de datos XML (SQLSTATE 42997). Una clave de distribución no puede contener columnas de inicio de fila, de fin de fila o de ID de inicio de transacción. Si no se especifica esta cláusula y la tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de distribución se define de la forma siguiente:

- Si la tabla es una tabla con tipo, la columna de identificador de objeto es la clave de distribución.
- Si se especifica una clave primaria, la primera columna de la clave primaria es la clave de distribución.
- En caso contrario, la primera columna con un tipo de datos válido para una clave de distribución se convierte en la clave de distribución.

Las columnas de la clave de distribución deben ser un subconjunto de las columnas que forman alguna restricción de unicidad.

Si ninguna columna cumple los requisitos de una clave de distribución por omisión, la tabla se crea sin ninguna. Estas tablas solo están permitidas en espacios de tablas definidos en grupos de particiones de base de datos de partición única.

CREATE TABLE

Para las tablas de los espacios de tablas que están definidos en grupos de particiones de base de datos de partición única, se puede utilizar cualquier conjunto de columnas con tipos de datos válidos para una clave de distribución a fin de definir la clave de distribución. Si no especifica esta cláusula, no se crea ninguna clave de distribución.

Para ver las restricciones relacionadas con la clave de distribución, consulte el apartado Normas.

DISTRIBUTE BY REPLICATION

Especifica que los datos almacenados en la tabla se duplican físicamente en cada partición de base de datos del grupo de particiones de base de datos de los espacios de tablas en que se define la tabla. Esto significa que existe una copia de todos los datos de la tabla en cada una de las particiones de base de datos. Esta opción sólo puede especificarse para una tabla de consulta materializada (SQLSTATE 42997).

cláusula-particionamiento

Especifica cómo se particionan los datos en una partición de base de datos.

PARTITION BY RANGE *espec-partición-rangos*

Especifica el esquema de particionamiento de tabla para la tabla.

expresión-partición

Especifica los datos de clave en que se define el rango para determinar la partición de datos de destino de los datos.

nombre-columna

Identifica una columna de clave de particionamiento de tabla. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No puede utilizarse ninguna columna con un tipo de datos que sea BLOB, CLOB, DBCLOB, XML, un tipo diferenciado basado en cualquiera de esos tipos o un tipo estructurado como parte de una clave de particionamiento de tabla (SQLSTATE 42962).

Las normas para los literales numéricos gobiernan los literales numéricos utilizadas en la especificación del rango. Todos los literales numéricos (excepto los valores especiales de coma flotante decimal) utilizados en los rangos correspondientes a las columnas numéricas se interpretan como constantes decimales, de coma flotante o entero, con arreglo a las normas especificadas para las constantes numéricas. En consecuencia, para las columnas de coma flotante decimal, el valor de constante numérica mínima y máxima que puede utilizarse en la especificación del rango de una partición de datos es el valor de DOUBLE más pequeño y el valor de DOUBLE más grande, respectivamente. Los valores especiales de coma flotante decimal pueden utilizarse en la especificación de rangos. Todos los valores especiales de coma flotante decimal se interpretan como superiores a MINVALUE e inferiores a MAXVALUE.

Las columnas de particionamiento de tablas no pueden contener una columna ROW CHANGE TIMESTAMP (SQLSTATE 429BV). El número de columnas identificadas no debe superar 16 (SQLSTATE 54008).

NULLS LAST

Indica que los valores nulos se comparan altos.

NULLS FIRST

Indica que los valores nulos se comparan bajos.

elemento-partición

Especifica los rangos de una clave de particionamiento de datos y el espacio de tablas donde se almacenarán las filas de la tabla del rango.

PARTITION *nombre-partición*

Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

espec-límite

Especifica los límites de una partición de datos. La partición de datos inferior debe contener una cláusula-inicial y la partición de datos superior debe contener una cláusula-final (SQLSTATE 56016). Las particiones de datos entre la inferior y la superior pueden contener una cláusula-inicial, una cláusula-final o ambas. Si solo se especifica la cláusula-final, la partición de datos anterior también debe contener una cláusula-final (SQLSTATE 56016).

cláusula-inicial

Especifica el extremo inferior del rango de una partición de datos. Debe haber como mínimo un valor inicial, y el número máximo de valores es el número de columnas de la clave de particionamiento de datos (SQLSTATE 53038). Si hay especificados menos valores que el número de columnas, los demás valores son implícitamente MINVALUE.

STARTING FROM

Especifica la *cláusula-inicial*.

constante

Especifica un valor constante con un tipo de datos asignable al tipo de datos del *nombre-columna* al que corresponde (SQLSTATE 53045). El valor no puede estar dentro del rango de ninguna otra *espec-límite* de la tabla (SQLSTATE 56016).

MINVALUE

Especifica un valor inferior al valor mínimo posible para el tipo de datos del *nombre-columna* al que corresponde.

MAXVALUE

Especifica un valor superior al valor máximo posible para el tipo de datos del *nombre-columna* al que corresponde.

INCLUSIVE

Indica que los valores de rango especificados deben incluirse en la partición de datos.

EXCLUSIVE

Indica que los valores *constantes* especificados deben excluirse de la partición de datos. Esta especificación se omite cuando se especifica MINVALUE o MAXVALUE.

cláusula-final

Especifica el extremo superior del rango de una partición de datos. Debe haber como mínimo un valor inicial, y el número máximo de valores es el número de columnas de la clave de particionamiento de datos (SQLSTATE 53038). Si hay especificados menos valores que el número de columnas, los demás valores son implícitamente MAXVALUE.

ENDING AT

Especifica la *cláusula-final*.

constante

Especifica un valor constante con un tipo de datos asignable al tipo de datos del *nombre-columna* al que corresponde (SQLSTATE 53045). El valor no puede estar dentro del rango de ninguna otra especificación de la tabla (SQLSTATE 56016).

MINVALUE

Especifica un valor inferior al valor mínimo posible para el tipo de datos del *nombre-columna* al que corresponde.

MAXVALUE

Especifica un valor superior al valor máximo posible para el tipo de datos del *nombre-columna* al que corresponde.

INCLUSIVE

Indica que los valores de rango especificados deben incluirse en la partición de datos.

EXCLUSIVE

Indica que los valores *constantes* especificados deben excluirse de la partición de datos. Esta especificación se omite cuando se especifica MINVALUE o MAXVALUE.

IN *nombre-espacio-tablas*

Especifica el espacio de tablas donde debe almacenarse la partición de datos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE. Si no se especifica esta cláusula, por omisión se asigna un espacio de tablas de modo rotativo a partir de la lista de espacios de tablas especificada para la tabla. Si no se ha especificado un espacio de tablas para objetos grandes con la cláusula LONG IN, los objetos grandes se colocan en el mismo espacio de tablas que las demás filas de la partición de datos. Para las tablas particionadas, se puede emplear la cláusula LONG IN para proporcionar una lista de espacios de tablas. Esta lista se utiliza de modo rotativo para colocar los objetos grandes de cada una de las particiones de datos. Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

Si no se especifica la cláusula INDEX IN en la sentencia CREATE TABLE o CREATE INDEX, el índice se coloca en el mismo espacio de tablas que la primera partición visible o enlazada de la tabla.

INDEX IN *nombre-espacio-tablas*

Especifica el espacio de tablas donde se debe almacenar el índice particionado de la tabla particionada.

La cláusula INDEX IN de nivel de elemento-partición sólo afecta al almacenamiento de los índices particionados. El almacenamiento del índice se realiza como se indica a continuación:

- Si la cláusula INDEX IN se especifica a nivel de partición al crear la tabla, el índice particionado se almacena en el espacio de tablas especificado.
- Si la cláusula INDEX IN no se especifica a nivel de partición al crear la tabla, el índice particionado se almacena en el espacio de tablas de la partición de datos correspondiente.

La cláusula INDEX IN sólo se puede especificar si los espacios de tablas de datos son espacios de tablas DMS y el espacio de tablas especificado por la cláusula INDEX IN es un espacio de tablas DMS. Si el espacio de tablas de datos es un espacio de tablas SMS, se devuelve un error (SQLSTATE 42839).

LONG IN *nombre-espacio-tablas*

Identifica los espacios de tablas donde se almacenarán los valores de todas las columnas largas. Las columnas largas incluyen las de tipos de datos LOB, tipo XML, tipos diferenciados con cualquiera de estos tipos como fuente o cualquier columna definida con tipos estructurados definidos por el usuario cuyos valores no se pueden almacenar en línea. Esta opción sólo está permitida si la cláusula IN identifica un espacio de tablas DMS.

El espacio de tablas especificado debe existir. Puede ser un espacio de tablas normal si es el mismo espacio de tablas en el que están almacenados los datos; de lo contrario, debe ser un espacio de tablas DMS grande en el que el ID de autorización de la sentencia tenga el privilegio USE. También debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tablas* (SQLSTATE 42838).

Se puede especificar qué espacio de tablas contendrá las columnas largas, LOB o XML al crear una tabla. La comprobación del privilegio USE se realiza al crear la tabla, no al añadir posteriormente una columna larga o LOB.

Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

EVERY (*constante*)

Especifica el ancho de cada rango de partición de datos al utilizar el formato de la sintaxis generado automáticamente. Las particiones de datos se crearán comenzando por el valor STARTING FROM y con este número de valores en el rango. Este formato de la sintaxis solo está permitido para las tablas particionadas por una única columna numérica o de fecha y hora (SQLSTATE 53038).

CREATE TABLE

Si la columna de clave de particionamiento es de tipo numérico, el valor inicial de la primera partición es el valor especificado en la cláusula-inicial. El valor final de la primera partición y todas las demás se calcula añadiendo el valor inicial de la partición al valor de incremento especificado como *constante* en la cláusula EVERY. El valor inicial de todas las demás particiones se calcula tomando el valor inicial de la partición anterior y añadiendo el valor de incremento especificado como *constante* en la cláusula EVERY.

Si la columna de clave de particionamiento es una fecha o una indicación de fecha y hora, el valor inicial de la primera partición es el valor especificado en la cláusula-inicial. El valor final de la primera partición y todas las demás se calcula añadiendo el valor inicial de la partición al valor de incremento especificado como duración etiquetada en la cláusula EVERY. El valor inicial de todas las demás particiones se calcula tomando el valor inicial de la partición anterior y añadiendo el valor de incremento especificado como duración etiquetada en la cláusula EVERY.

Para una columna de tipo numérico, el valor EVERY debe ser una constante numérica positiva; para una columna de fecha y hora, el valor EVERY debe ser una duración etiquetada (SQLSTATE 53045).

COMPRESS

Especifica si se aplica la compresión de datos a las filas de la tabla.

NO Especifica que la compresión de filas de datos está inhabilitada.

YES

Especifica que la compresión de filas de datos está habilitada. Las operaciones de inserción y actualización de la tabla estarán sujetas a compresión. También se comprimirán todos los objetos de almacenamiento XML que existan. Tanto para la compresión de filas adaptativa como para la clásica, se crea automáticamente un diccionario de compresión para la tabla después de que la tabla se haya llenado de datos de forma suficiente. Esto también se aplica a los datos del objeto de almacenamiento XML; si existen datos suficientes en el objeto de almacenamiento XML, se crea automáticamente un diccionario de compresión y los documentos XML están sujetos a compresión.

Nota: La compresión aplicada al objeto de almacenamiento XML es la misma, tanto si utiliza la compresión de filas adaptativa como la clásica.

Para la compresión de filas adaptativa, los diccionarios de compresión de nivel de página se crean o actualizan a medida que se especifican datos o se cambian en la tabla.

ADAPTIVE

Habilita la compresión adaptativa y los registros quedan sujetos a compresión con un diccionario de compresión de nivel de tabla o de nivel de página. Es la opción por omisión cuando se especifica COMPRESS YES. La funcionalidad de COMPRESS YES ADAPTIVE es un superconjunto de la funcionalidad de COMPRESS YES STATIC.

STATIC

Habilita la compresión de filas clásica mediante un diccionario de compresión de nivel de tabla. Es la misma funcionalidad de compresión de filas que existía en las versiones de DB2 anteriores.

VALUE COMPRESSION

Determina el formato de fila que debe utilizarse. Cada tipo de datos tiene un número total de bytes distinto según el formato de fila que se utilice. Para obtener más información, consulte el apartado Número total de bytes. 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).

El valor null se almacena utilizando tres bytes. Es el mismo espacio o menor que cuando VALUE COMPRESSION no se ha activado para las columnas de todos los tipos de datos, con la excepción de CHAR(1). El hecho de definir o no una columna como con posibilidad de nulos no afecta al cálculo del tamaño de fila. Los valores de datos de longitud cero para las columnas cuyo tipo de datos es VARCHAR, VARGRAPHIC, LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB, BLOB o XML deben almacenarse utilizando dos bytes únicamente, que es un valor inferior al almacenamiento necesario cuando VALUE COMPRESSION no se ha activado. Cuando una columna se define con la opción COMPRESS SYSTEM DEFAULT, se permite también que el valor por omisión del sistema perteneciente a la columna se almacene utilizando tres bytes de almacenamiento total. El formato de fila utilizado para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos al actualizar al valor null, a un valor de longitud cero o al valor por omisión del sistema o bien al actualizar desde estos valores.

WITH RESTRICT ON DROP

Indica que la tabla no puede descartarse y que el espacio de tablas que contiene la tabla no puede descartarse.

NOT LOGGED INITIALLY

Los cambios realizados en la tabla por una operación de Inserción, Supresión, Actualización, Creación de índice, Descarte 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. Para conocer otras consideraciones que deben observarse al utilizar esta opción, consulte el apartado “Notas” de esta sentencia.

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.

Nota: Si se produce actividad que no es de anotaciones cronológicas para una tabla que tiene activado el atributo NOT LOGGED INITIALLY y si una sentencia no se ejecuta satisfactoriamente (dando lugar a una retroacción) o si se ejecuta ROLLBACK TO SAVEPOINT, se retrotraerá la unidad de trabajo completa (SQL1476N). Además, la tabla para la que se había activado el atributo NOT LOGGED INITIALLY se marcará como no accesible tras producirse la retroacción y sólo podrá descartarse. 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.

CCSID

Especifica el esquema de codificación para los datos de serie almacenados en la tabla. Si no se especifica la cláusula CCSID, el valor por omisión es CCSID UNICODE para bases de datos Unicode y CCSID ASCII para las demás bases de datos.

CREATE TABLE

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8.

Si la base de datos no es Unicode, se pueden crear tablas con CCSID UNICODE, pero se aplican las normas siguientes:

- Se debe especificar el orden de clasificación alternativo en la configuración de la base de datos antes de crear la tabla (SQLSTATE 56031). Las tablas CCSID UNICODE se clasifican con el orden de clasificación alternativo especificado en la configuración de la base de datos.
- No se pueden utilizar conjuntamente las tablas o las funciones de tablas creadas con CCSID ASCII y las tablas o las funciones de tablas creadas con CCSID UNICODE en una sola sentencia de SQL (SQLSTATE 53090). Esto se aplica a las tablas y a las funciones de tabla a las que se hace referencia directamente en la sentencia, así como a las tablas y las funciones de tabla a las que se hace referencia indirectamente (por ejemplo, mediante restricciones de integridad referencial, activadores, tablas de consulta materializada y tablas de los cuerpos de las vistas).
- No se puede hacer referencia a las tablas creadas con CCSID UNICODE en funciones de SQL o en métodos de SQL (SQLSTATE 560C0).
- Una sentencia de SQL que hace referencia a una tabla creada con CCSID UNICODE no puede invocar una función de SQL ni un método de SQL (SQLSTATE 53090).
- Los tipos gráficos, el tipo XML y los tipos definidos por el usuario no se pueden utilizar en tablas CCSID UNICODE (SQLSTATE 560C1).
- Los tipos de datos anclados de una tabla creada con CCSID UNICODE (SQLSTATE 428HS).
- Las tablas no pueden especificar las cláusulas CCSID UNICODE y DATA CAPTURE CHANGES a la vez (SQLSTATE 42613).
- Las tablas de Explain no se pueden crear con CCSID UNICODE (SQLSTATE 55002).
- Las tablas temporales creadas, y declaradas tablas temporales, no se pueden crear con CCSID UNICODE (SQLSTATE 56031).
- Las tablas CCSID UNICODE no se pueden crear en una sentencia CREATE SCHEMA (SQLSTATE 53090).
- La tabla de excepciones para una operación de carga debe tener el mismo CCSID que la tabla de destino de la operación (SQLSTATE 428A5).
- La tabla de excepciones para una sentencia SET INTEGRITY debe tener el mismo CCSID que la tabla de destino para la sentencia (SQLSTATE 53090).
- La tabla de destino para los datos del supervisor de sucesos no se debe declarar como CCSID UNICODE (SQLSTATE 55049).

- Las sentencias que hacen referencia a una tabla CCSID UNICODE sólo se pueden invocar desde un cliente de DB2 Versión 8.1 o posterior (SQLSTATE 42997).
- Las sentencias de SQL siempre se interpretan en la página de códigos de la base de datos. En particular, significa que cada carácter de los literales, literales hexadecimales e identificadores delimitados debe tener una representación en la página de códigos de la base de datos; de lo contrario, el carácter se sustituirá por el carácter de sustitución.

Las variables del lenguaje principal de la aplicación siempre están en la página de códigos de la aplicación, sin tener en cuenta el CCSI de ninguna tabla de las sentencias de SQL que se invocan. DB2 realizará conversiones de páginas de códigos cuando sea necesario para convertir datos entre la página de códigos de la aplicación y la página de códigos de la sección. Se puede establecer la variable de registro DB2CODEPAGE en el cliente para cambiar la página de códigos de la aplicación.

SECURITY POLICY

Indica el nombre de la política de seguridad que se asociará a la tabla.

nombre-política

Identifica una política de seguridad que ya existe en el servidor actual (SQLSTATE 42704).

OPTIONS (*nombre-opción-tabla constante-serie, ...*)

Se utilizan opciones de tabla para identificar la tabla base remota. El *nombre-opción-tabla* es el nombre de la opción. La *constante-serie* especifica el valor para la opción de tabla. La *constante-serie* se debe especificar entre comillas simples.

El servidor remoto (el nombre de servidor que se ha especificado en la sentencia CREATE SERVER) se debe especificar en la cláusula OPTIONS. La cláusula OPTIONS también se puede utilizar para alterar temporalmente el esquema o el nombre no calificado de la tabla base remota que se crea.

Se recomienda especificar el nombre de esquema. Si no se especifica un nombre de esquema remoto, se utiliza el calificador para el nombre de tabla. Si el nombre de tabla no tiene calificador, se utiliza el ID de autorización de la sentencia.

Si no se especifica un nombre no calificado para la tabla base remota, se utiliza *nombre-tabla*.

Normas

- La suma de las cuentas de bytes de las columnas, incluidas las longitudes en línea de todas las columnas de tipo estructurado o XML 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). Para obtener más información, consulte el apartado Número total de bytes. Para las tablas con tipo, el número total de bytes se aplica a las columnas de la tabla raíz de la jerarquía de tablas y cada columna adicional que cada subtabla incorpora a la jerarquía de tablas (las columnas de subtabla adicionales deben considerarse como columnas con posibilidad de nulos para la obtención del número total de bytes, incluso si se han definido como columnas sin posibilidad de nulos). También hay 4 bytes adicionales de actividad general para identificar la subtabla a la que pertenece cada fila.

CREATE TABLE

- El número de columnas de una tabla no puede ser superior a 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).
- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de distribución (SQLSTATE 42997).
- Las normas siguientes solo se aplican a las bases de datos de varias particiones de base de datos.
 - Las tablas compuestas únicamente de columnas con tipos LOB, XML, un tipo diferenciado basado en uno de estos tipos o un tipo estructurado sólo pueden crearse en los espacios de tablas que se definen en grupos de particiones de base de datos de partición única.
 - La definición de clave de distribución de una tabla de un espacio de tablas que se ha definido en un grupo de particiones de base de datos de varias particiones no puede modificarse.
 - La columna de clave de distribución de una tabla con tipo debe ser la columna de OID.
 - No están permitidas las tablas de etapas particionadas.
- Para las bases de datos que se ejecuten en un entorno DB2 pureScale, la cláusula ORGANIZE BY no puede especificarse (SQLSTATE 42997).
- Se aplican las siguientes restricciones a las tablas agrupadas por clústeres de rangos:
 - No se puede especificar una tabla agrupada por clústeres de rangos en un entorno DB2 pureScale (SQLSTATE 42997).
 - No se puede crear un índice de clústeres.
 - No se da soporte a la modificación de la tabla para añadir una columna.
 - No se da soporte a la modificación de la tabla para cambiar el tipo de datos de una columna.
 - No se da soporte a la modificación de la tabla para cambiar PCTFREE.
 - No se da soporte a la modificación de la tabla para establecer APPEND ON.
 - Las estadísticas DETALLADAS no están disponibles.
 - No se puede utilizar el programa de utilidad de carga para llenar la tabla.
 - Las columnas no pueden ser de tipo XML.
- Una tabla no está protegida salvo que tenga asociada una política de seguridad y contenga una columna de tipo DB2SECURITYLABEL o una columna definida con la cláusula SECURED WITH. El primer caso indica que la tabla es una tabla protegida con **granularidad en el nivel de fila** y el segundo indica que la tabla es una tabla protegida con **granularidad en el nivel de columna**.
- La declaración de una columna de tipo DB2SECURITYLABEL falla si la tabla no tiene asociada una política de seguridad (SQLSTATE 55064).
- No se puede añadir una política de seguridad a una tabla con tipo (SQLSTATE 428DH), tabla de consulta materializada o tabla de etapas (SQLSTATE 428FG).
- No se puede especificar una *expresión-tabla-anidada* tolerante a errores en la selección completa de una *definición-consulta-materializada* (SQLSTATE 428GG).
- Al crear una tabla de consulta materializada, si alguna de las tablas base de las que depende está protegida con control de acceso basado en etiquetas, se aplican las siguientes normas:
 - Seguridad a nivel de fila

- Solo una tabla de la selección completa de la tabla de consulta materializada puede tener un tipo de columna DB2SECURITYLABEL (SQLSTATE 428FG).
- Se debe seleccionar la columna de etiqueta de seguridad de fila y hacer referencia a ella como una columna autónoma en la lista SELECT más externa de la definición de tabla de consulta materializada (SQLSTATE 428FG). La columna correspondiente en la tabla de consulta materializada quedará marcada como la columna de etiqueta de seguridad de fila.
- Seguridad a nivel de columna
 - Si una tabla implicada en la definición de tabla de consulta materializada tiene una columna protegida con una etiqueta de seguridad y esa columna aparece en la definición de tabla de consulta materializada, la etiqueta de seguridad de esa columna es heredada por la columna correspondiente de la tabla de consulta materializada. Para más información, vea los ejemplos incluidos en este tema.
- Al crear una tabla de consulta materializada que depende de una o más tablas protegidas por control de acceso basado en etiquetas, todas las tablas base deben tener el mismo objeto de política de seguridad (SQLSTATE, 428FG). La tabla de consulta materializada quedará automáticamente protegida con ese objeto de política de seguridad.
- La etiqueta de seguridad asociada a una columna de una tabla de consulta materializada se calcula como el agregado de una o más etiquetas de seguridad. Este agregado consiste en las etiquetas de seguridad asociadas a las columnas de las tablas base que participan en la definición de la columna de dicha tabla de consulta materializada. El agregado también consiste en las etiquetas de seguridad asociadas a cualquier columna de tabla base que aparezca en otras partes de la definición de tabla de consulta materializada, como las cláusulas WHERE, ORDER BY y HAVING. **ALTER SECURITY POLICY** posee una descripción de cómo se agregan dos etiquetas de seguridad. Para más información, vea los ejemplos incluidos en este tema.
- Cuando se crea una tabla de etapas para una tabla de consulta materializada que está protegida con control de acceso basado en etiquetas, esa tabla de etapas posee protección automática como la tabla de consulta materializada. Para más información, vea los ejemplos incluidos en este tema.
- El control de acceso basado en etiquetas se aplica para el acceso directo a una tabla de consulta materializada, del mismo modo que se aplica en el caso de una tabla normal. No hay diferencias desde esta perspectiva. Cuando el compilador de SQL atiende una consulta a través de una tabla de consulta materializada, no es necesario aplicar el control de acceso basado en etiquetas definido en la propia tabla de consulta materializada. El compilador de SQL utiliza la tabla de consulta materializada que tiene en cuenta las normas de control de acceso basado en etiquetas desde las tablas base adecuadas.
- La *cláusula-isolation* no puede especificarse en la *selección-completa* de la *definición-tabla-consulta-materializada* (SQLSTATE 42601).
- Las sentencias de subselección que contienen una *cláusula-lock-request* no son candidatas al direccionamiento de MQT.
- Solamente se pueden especificar caracteres nacionales para los tipos de datos gráficos en una base de datos Unicode (SQLSTATE 560AA).
- Se aplican las siguientes restricciones para insertar las tablas de agrupación en clústeres de tiempo de inserción (ITC):
 - Las tablas ITC no se admiten en un espacio de tablas SMS (SQLSTATE 42838).
 - Los índices definidos en tablas ITC no se admiten en un espacio de tablas SMS (SQLSTATE 42838).

Notas

- La creación de una tabla 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre 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.
- Las 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 si se invoca la función con un valor de argumento que excede estas longitudes (SQLSTATE 22001).
- La utilización de NO ACTION o RESTRICT como normas de supresión o actualización para restricciones de referencia determina cuándo se aplica la restricción. Una norma de supresión o actualización de RESTRICT se aplicará *antes* que las demás restricciones, incluidas las restricciones de referencia con normas de modificación como CASCADE o SET NULL. Una norma de supresión o de actualización de NO ACTION se aplica *después* de otras restricciones de referencia. Un ejemplo en el que es evidente un comportamiento distinto implica la supresión de filas de una vista que se ha definido como UNION ALL de las tablas relacionadas.

La tabla T1 es una tabla padre de la tabla T3; norma de supresión, como se explica a continuación.

La tabla T2 es una tabla padre de la tabla T3; norma 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 que tiene la norma de supresión RESTRICT, se producirá una violación de restricción (SQLSTATE 23001) si existen filas hijas para las claves padre de T1 en T3.

Si la tabla T1 es una tabla padre de la tabla T3 que tiene la norma de supresión de NO ACTION, la norma de supresión de CASCADE puede suprimir las filas hijas al suprimirse las filas de T2, antes de que se aplique la norma de supresión de NO ACTION para las supresiones de T1. Si las supresiones de T2 no han tenido como resultado la supresión de todas las filas hijas 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 norma de supresión o actualización es RESTRICT o NO ACTION.

- Para las tablas de los espacios de tablas que se han definido en grupos de particiones de base de datos de varias particiones, en la elección de las claves de distribución debe tenerse en cuenta la colocación de la tabla. A continuación encontrará una lista de elementos que debe tener en cuenta:
 - Para la colocación, las tablas deben encontrarse en el mismo grupo de particiones de base de datos. Los espacios de tablas pueden ser distintos, pero deben haberse definido en el mismo grupo de particiones de base de datos.
 - Las claves de distribución de las tablas deben tener el mismo número de columnas, y las columnas de clave correspondientes deben tener particiones de base de datos compatibles para la colocación.

- La elección de la clave de distribución también afecta al rendimiento de las uniones. Si una tabla se une con frecuencia a otra tabla, plantéese la conveniencia de unir las columnas como una clave de distribución para ambas tablas.
- La opción NOT LOGGED INITIALLY es útil para las situaciones en que se debe crear un conjunto de resultados 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 ejecuta el programa de utilidad de avance (Rollforward) y éste detecta un registro de anotaciones cronológicas que indica que el programa de utilidad de carga (Load) ha llenado una tabla de la base de datos o que ésta se ha creado con la opción NOT LOGGED INITIALLY, la tabla se marcará como no disponible. El programa de utilidad de avance (Rollforward) descartará 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 descartar 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.
- Una tabla de consulta materializada mantenida por el sistema REFRESH DEFERRED definida con ENABLE QUERY OPTIMIZATION puede utilizarse para optimizar el proceso de las consultas si CURRENT REFRESH AGE se ha establecido en ANY y si CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se ha establecido de forma que incluya tablas de consulta materializada mantenidas por el sistema. Una tabla de consulta materializada mantenida por el usuario REFRESH DEFERRED definida con ENABLE QUERY OPTIMIZATION puede utilizarse para optimizar el proceso de las consultas si CURRENT REFRESH AGE se ha establecido en ANY y si CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se ha establecido de forma que incluya tablas de consulta materializada mantenidas por el usuario. Para la optimización, siempre se considera una tabla de consulta materializada REFRESH IMMEDIATE definida con ENABLE QUERY OPTIMIZATION. Para que esta optimización pueda utilizar una tabla de consulta materializada REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa debe ajustarse a determinadas normas, además de las normas que ya se han descrito:
 - La selección completa debe ser una subselección con una cláusula GROUP BY o una subselección con una sola referencia de tabla.
 - La selección completa no debe incluir registros especiales ni funciones incorporadas que dependen del valor de un registro especial.
 - La selección completa no debe incluir ninguna variable global.
 - La selección completa no debe incluir funciones que no sean deterministas.
 - La selección completa no debe incluir una cláusula FETCH FIRST *n* ROWS ONLY.
 - La selección completa no debe incluir una cláusula ORDER BY.

Si la consulta que se ha especificado al crear una tabla de consulta materializada no se ajusta a estas normas, se devolverá un mensaje de aviso (SQLSTATE 01633).

CREATE TABLE

- Si la tabla de consulta materializada se define con REFRESH IMMEDIATE, o una tabla de etapas se define con PROPAGATE IMMEDIATE, es posible que se produzca un error al intentar aplicar el cambio resultante de una operación de inserción, actualización o supresión en una tabla subyacente. El error provocará la anomalía de la operación de inserción, actualización o supresión de la tabla subyacente.
- Las tablas de consulta materializada o las tablas de etapas no se pueden utilizar como tablas de excepción cuando las restricciones se comprueban masivamente, por ejemplo, durante las operaciones de carga o durante la ejecución de la sentencia SET INTEGRITY.
- Algunas operaciones no se pueden realizar en la tabla a la que hace referencia una tabla de consulta materializada definida con REFRESH IMMEDIATE, o definida con REFRESH DEFERRED con una tabla de etapas asociada:
 - No se puede utilizar IMPORT REPLACE.
 - No se puede llevar a cabo ALTER TABLE NOT LOGGED INITIALLY WITH EMPTY TABLE.
- En un sistema federado, los apodos para las fuentes de datos relacionales o las tablas locales pueden utilizarse como tablas subyacentes para crear una tabla de consulta materializada. Los apodos para las fuentes de datos no relacionales no están soportados. Cuando un apodo es una de las tablas subyacentes, debe utilizarse la opción REFRESH DEFERRED. No se da soporte a las tablas de consulta materializada mantenidas por el sistema que hacen referencia a apodos en un entorno de base de datos particionada.
- **Consideraciones sobre las columnas de ID de inicio de transacción:** Una columna de ID de inicio de transacción contiene un valor nulo si la columna soporta valores nulos, y existe una columna de inicio de fila y el valor de la columna es único entre los valores de las columnas de inicio de fila generados para otras transacciones. Puesto que la columna puede contener valores nulos, se recomienda que se utilice uno de los métodos siguientes cuando se recupere un valor de la columna:

```
COALESCE (col_ID_inicio_transacción, col_inicio_fila)

CASE WHEN col_ID_inicio_transacción IS NOT NULL
      THEN col_ID_inicio_transacción
      ELSE col_inicio_fila END
```
- **Definición de una tabla temporal de período de sistema:** una definición de tabla temporal de inicio de sistema incluye lo siguiente:
 - Un período de sistema denominado SYSTEM_TIME, que se define mediante una columna de inicio de fila y una columna de fin de fila. Consulte las descripciones de AS ROW BEGIN, AS ROW END y de definición de período.
 - Una columna de ID de inicio de transacción. Consulte la descripción de AS TRANSACTION START ID.
 - Una definición de mantenimiento de versiones de datos de período de sistema especificada en una sentencia ALTER TABLE posterior que especifique la acción ADD VERSIONING, que incluye el nombre de la tabla histórica asociada. Consulte la descripción de la cláusula ADD VERSIONING de ALTER TABLE.

Para asegurarse de que la tabla histórica no se pueda descartar de forma implícita cuando se descarte una tabla temporal de período del sistema, utilice la cláusula WITH RESTRICT ON DROP en la definición de la tabla histórica. Una tabla histórica puede descartarse manualmente solo si se elimina el atributo RESTRICT ON DROP mediante una sentencia ALTER TABLE.

- **Definición de una tabla temporal de período de aplicación:** una definición de tabla temporal de período de aplicación incluye un período de aplicación denominado BUSINESS_TIME. El período de aplicación se define mediante una columna de indicación de fecha y hora de inicio y una columna de indicación de fecha y hora de finalización. Consulte la descripción de definición de período.

Las operaciones de cambio de datos en una tabla temporal de período de aplicación pueden generar una inserción automática de una o dos filas adicionales cuando se actualiza o se suprime una fila. Cuando se especifica una actualización o una supresión de una fila en una tabla temporal de período de aplicación para una parte del período que representa esa fila, la fila se actualiza o se suprime y se insertan una o dos filas de forma automática para representar la parte de la fila que no cambia. Se generan nuevos valores para cada columna generada en una tabla temporal de período de aplicación para cada fila que se inserta de forma automática como resultado de una operación de actualización o supresión en la tabla. Si se define una columna generada como parte de una clave exclusiva o primaria, una clave principal de una restricción referencial o un índice exclusivo, es posible que una inserción automática viole una restricción o índice, en cuyo caso se devolverá un error.

- **Consideraciones para columnas ocultas implícitamente:** La creación de una tabla con columnas ocultas implícitamente puede afectar al comportamiento de los programas de utilidad de movimiento de datos que funcionan con la tabla. Cuando una tabla de base de datos contiene columnas ocultas implícitamente, algunos programas de utilidad como IMPORT, INGEST y LOAD requieren que especifique si los datos para las columnas ocultas han de incluirse en la operación. Por ejemplo, esto podría significar que una operación de carga que se ejecuta correctamente para una tabla sin ninguna columna oculta podría no ejecutarse correctamente para una tabla que contiene columnas ocultas implícitamente (SQLCODE SQL2437N). De forma similar, EXPORT requiere que especifique si los datos para las columnas ocultas han de incluirse en la operación.

Los programas de movimiento de datos utilizan la variable de registro DB2_DMU_DEFAULT, o los modificadores de tipo de archivo **implicitlyhiddeninclude** o **implicitlyhiddenmissing**, al trabajar con tablas que contienen columnas ocultas implícitamente.

- **DDL transparente:** En un sistema federado, se puede crear, modificar o descartar una tabla base remota utilizando DB2 SQL. Esta posibilidad se conoce como *DDL transparente*. Para poder crear una tabla base remota en una fuente de datos, el servidor federado debe estar configurado para acceder a esa fuente de datos. Esta configuración incluye la creación de un derivador para la fuente de datos, el suministro de la definición del servidor para el servidor en el que se ubicará la tabla base remota y la creación de correlaciones de usuario entre el servidor federado y la fuente de datos.

El DDL transparente impone algunas limitaciones en los elementos que pueden incluirse en la sentencia CREATE TABLE:

- Sólo se pueden crear columnas y una clave primaria en la tabla base remota.
- Entre las clases específicas que admite el DDL transparente se encuentran las siguientes:
 - *definición-columna* y *restricción-unicidad* en la cláusula *lista-elementos*
 - NOT NULL y PRIMARY KEY en la cláusula *opciones-columna*
 - OPTIONS
- La fuente de datos remota debe dar soporte a:
 - Los tipos de datos de la columna remota con los que están correlacionados los tipos de datos de la columna DB2

CREATE TABLE

- La opción de clave primaria de la sentencia CREATE TABLE

Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.

Cuando se crea una tabla base remota utilizando un DDL transparente, se crea automáticamente un apodo para esta tabla base remota.

- 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 norma 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 que la tabla descartada es la tabla padre o la tabla dependiente, se descarta la restricción
 - para las restricciones de referencia en las que la tabla padre es una supertabla de la tabla descartada, se consideran descartadas de la supertabla las filas de la tabla descartada. Se comprueba la restricción de referencia y se invoca su norma de supresión para cada una de las filas suprimidas.
 - para las restricciones de referencia en las que la tabla dependiente es una supertabla de la tabla descartada, 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.
- **Privilegios:** cuando se crea una tabla, el definidor de la tabla recibe el privilegio CONTROL. 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.
- **Límite del tamaño de fila:** el número máximo de bytes permitido en la fila de una tabla depende del tamaño de página del espacio de tablas donde se crea la tabla (*nombre1-espacio-tablas*). 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 21. Límites para el número de columnas y el tamaño de fila de 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:

$$\text{Columnas en total} * 8 + \text{Número de columnas LOB} * 12 \leq \text{Límite del tamaño de fila para el tamaño de página}$$

- **Número total de bytes:** la tabla siguiente contiene el número total de bytes de las columnas por tipo de datos. Esta tabla se utiliza para calcular el tamaño de fila. El número total de bytes depende de si se ha activado o no VALUE COMPRESSION. Cuando VALUE COMPRESSION no se ha activado, el número total de bytes también depende de si la columna puede contener nulos o no. El número total de bytes mostrados se aplica cuando la compresión de filas no está habilitada. Si la compresión de filas está activa, el número total de bytes utilizados por una fila generalmente será inferior que los utilizados para una versión no comprimida de la fila; nunca será superior.

Si una tabla se basa en un tipo estructurado, se reservarán 4 bytes adicionales de actividad general para identificar las filas de subtablas, con independencia de si se han definido o no subtablas. Las columnas de subtabla adicionales deben considerarse como columnas con posibilidad de nulos para la obtención del número total de bytes, incluso si se han definido como columnas sin posibilidad de nulos.

Tabla 22. Número total de bytes de las columnas por tipo de datos

Tipo de datos	VALUE COMPRESSION no se ha activado		
	VALUE COMPRESSION se ha activado ¹	La columna puede contener nulos	La columna no puede contener nulos
SMALLINT	4	3	2
INTEGER	6	5	4
BIGINT	10	9	8
REAL	6	5	4
DOUBLE	10	9	8
DECIMAL	La parte integral de $(p/2)+3$, donde p es la precisión	La parte integral de $(p/2)+2$, donde p es la precisión	La parte integral de $(p/2)+1$, donde p es la precisión
DECFLOAT(16)	10	9	8
DECFLOAT(34)	18	17	16
CHAR (n)	$n+2$	$n+1$	n
VARCHAR (n)	$n+2$	$n+5$ (dentro de una tabla)	$n+4$ (dentro de una tabla)
LONG VARCHAR ²	22	25	24
GRAPHIC (n)	$n*2+2$	$n*2+1$	$n*2$
VARGRAPHIC (n)	$n*2+2$	$n*2+5$ (dentro de una tabla)	$n*2+4$ (dentro de una tabla)
LONG VARGRAPHIC ²	22	25	24
DATE	6	5	4
TIME	5	4	3
TIMESTAMP(p)	La parte integral de $(p+1)/2+9$, donde p es la precisión de segundos fraccionarios	La parte integral de $(p+1)/2+8$, donde p es la precisión de segundos fraccionarios	La parte integral de $(p+1)/2+7$, donde p es la precisión de segundos fraccionarios
XML (sin INLINE LENGTH especificado)	82	85	84
XML (con INLINE LENGTH especificado)	INLINE LENGTH +2	INLINE LENGTH +4	INLINE LENGTH +3
Longitud máxima de LOB ³ 1.024 (sin INLINE LENGTH especificado)	70	73	72

CREATE TABLE

Tabla 22. Número total de bytes de las columnas por tipo de datos (continuación)

Tipo de datos	VALUE COMPRESSION se ha activado ¹	VALUE COMPRESSION no se ha activado	
		La columna puede contener nulos	La columna no puede contener nulos
Longitud máxima de LOB 8.192 (sin INLINE LENGTH especificado)	94	97	96
Longitud máxima de LOB 65.536 (sin INLINE LENGTH especificado)	118	121	120
Longitud máxima de LOB 524.000 (sin INLINE LENGTH especificado)	142	145	144
Longitud máxima de LOB 4.190.000 (sin INLINE LENGTH especificado)	166	169	168
Longitud máxima de LOB 134.000.000 (sin INLINE LENGTH especificado)	198	201	200
Longitud máxima de LOB 536.000.000 (sin INLINE LENGTH especificado)	222	225	224
Longitud máxima de LOB 1.070.000.000 (sin INLINE LENGTH especificado)	254	257	256
Longitud máxima de LOB 1.470.000.000 (sin INLINE LENGTH especificado)	278	281	280
Longitud máxima de LOB 2.147.483.647 (sin INLINE LENGTH especificado)	314	317	316
LOB sin INLINE LENGTH especificado	INLINE LENGTH +2	INLINE LENGTH +5	INLINE LENGTH +4

¹ Existen 2 bytes adicionales de almacenamiento utilizados por cada fila cuando VALUE COMPRESSION se ha activado para esa fila.

² Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro.

³ Cada valor de LOB tiene un *descriptor de 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.

Para un *tipo diferenciado*, el número total de bytes equivale a la longitud del tipo de fuente del tipo diferenciado. Para un *tipo de referencia*, el número total de bytes equivale a la longitud del tipo de datos incorporado en el que se basa el tipo de referencia. Para un *tipo estructurado*, el número total de bytes equivale a `INLINE LENGTH + 4`. El valor `INLINE LENGTH` es el valor que se especifica (o que se calcula implícitamente) para la columna de la cláusula *opciones-columna*.

Los tamaños de fila para las tablas de ejemplo siguiente parten de que no se especifica VALUE COMPRESSION:

```
DEPARTMENT 63 (0 + 3 + 33 + 7 + 3 + 17)
ORG         57 (0 + 3 + 19 + 2 + 15 + 18)
```

Si VALUE COMPRESSION se especifica, los tamaños de fila pasarán a ser los siguientes:

DEPARTMENT 69 (2 + 5 + 31 + 8 + 5 + 18)
 ORG 53 (2 + 4 + 16 + 4 + 12 + 15)

- **Número total de bytes de almacenamiento:** las tablas siguientes contienen el número total de bytes de almacenamiento de las columnas por tipo de datos para valores de datos.

La primera tabla define los conjuntos de atributos. Se hace referencia a estos atributos en la segunda tabla, que contiene los detalles del número total de bytes para cada tipo de datos.

El número total de bytes depende de si se ha activado VALUE COMPRESSION. Cuando VALUE COMPRESSION no se ha activado, el número total de bytes también depende de si la columna puede contener nulos. Los valores de la tabla representan la cantidad de almacenamiento (en bytes) que se utiliza para almacenar el valor. El número total de bytes mostrados se aplica cuando la compresión de filas no está habilitada. Si la compresión de filas está activa, el número total de bytes utilizados por una fila generalmente será inferior que los utilizados para una versión no comprimida de la fila; nunca será superior.

Tabla 23. Definiciones de los criterios expuestos en la tabla relacionada

Caso	Valor de datos	VALUE COMPRESSION	Capacidad de nulos de columna
A	NULL	No activo	Posibilidad de nulos
B	NULL	Activo ²	Posibilidad de nulos
C	Longitud cero	Activo ²	No aplicable
D	Valor por omisión del sistema ¹	Activo ²	No aplicable
E	Resto de valores de datos	No activo	Posibilidad de nulos
F	Resto de valores de datos	No activo	Sin posibilidad de nulos
G	Resto de valores de datos	Activo ²	No aplicable

¹ Cuando COMPRESS SYSTEM DEFAULT se ha especificado para la columna.

² Existen 2 bytes adicionales de almacenamiento utilizados por cada fila cuando VALUE COMPRESSION se ha activado para esa fila.

Tabla 24. Número total de bytes de almacenamiento basado en el formato de fila, tipo de datos y valor de datos

Tipo de datos	Caso A	Caso B	Caso C	Caso D	Caso E	Caso F	Caso G
SMALLINT	3	3	-	3	3	2	4
INTEGER	5	3	-	3	5	4	6
BIGINT	9	3	-	3	9	8	10
REAL	5	3	-	3	5	4	6
DOUBLE	9	3	-	3	9	8	10
DECIMAL	La parte integral de $(p/2)+2$, donde p es la precisión	3	-	3	La parte integral de $(p/2)+2$, donde p es la precisión	La parte integral de $(p/2)+1$, donde p es la precisión	La parte integral de $(p/2)+3$, donde p es la precisión
DECFLOAT(16)	9	3	-	3	9	8	10
DECFLOAT(34)	17	3	-	3	17	16	18
CHAR (n)	$n+1$	3	-	3	$n+1$	n	$n+2$

CREATE TABLE

Tabla 24. Número total de bytes de almacenamiento basado en el formato de fila, tipo de datos y valor de datos (continuación)

Tipo de datos	Caso A	Caso B	Caso C	Caso D	Caso E	Caso F	Caso G
VARCHAR (<i>n</i>)	5	3	2	2	$N+5$, donde N es el número de bytes en los datos	$N+4$, donde N es el número de bytes en los datos	$N+2$, donde N es el número de bytes en los datos
LONG VARCHAR ²	5	3	2	2	25	24	22
GRAPHIC (<i>n</i>)	n^*2+1	3	-	3	n^*2+1	n^*2	n^*2+2
VARGRAPHIC (<i>n</i>)	5	3	2	2	N^*2+5 , donde N es el número de bytes en los datos	N^*2+4 , donde N es el número de bytes en los datos	N^*2+2 , donde N es el número de bytes en los datos
LONG VARGRAPHIC ²	5	3	2	2	25	24	22
DATE	5	3	-	-	5	4	6
TIME	4	3	-	-	4	3	5
TIMESTAMP(<i>p</i>)	La parte integral de $(p+1)/2+8$, donde p es la precisión de segundos fraccionarios	3	-	-	La parte integral de $(p+1)/2+8$, donde p es la precisión de segundos fraccionarios	La parte integral de $(p+1)/2+7$, donde p es la precisión de segundos fraccionarios	La parte integral de $(p+1)/2+9$, donde p es la precisión de segundos fraccionarios
Longitud máxima de LOB ¹ 1024	5	3	2	2	(De 60 a 68)+5	(De 60 a 68)+4	(De 60 a 68)+2
Longitud máxima de LOB 8192	5	3	2	2	(De 60 a 92)+5	(De 60 a 92)+4	(De 60 a 92)+2
Longitud máxima de LOB 65.536	5	3	2	2	(De 60 a 116)+5	(De 60 a 116)+4	(De 60 a 116)+2
Longitud máxima de LOB 524.000	5	3	2	2	(De 60 a 140)+5	(De 60 a 140)+4	(De 60 a 140)+2
Longitud máxima de LOB 4.190.000	5	3	2	2	(De 60 a 164)+5	(De 60 a 164)+4	(De 60 a 164)+2
Longitud máxima de LOB 134.000.000	5	3	2	2	(De 60 a 196)+5	(De 60 a 196)+4	(De 60 a 196)+2
Longitud máxima de LOB 536.000.000	5	3	2	2	(De 60 a 220)+5	(De 60 a 220)+4	(De 60 a 220)+2
Longitud máxima de LOB 1.070.000.000	5	3	2	2	(De 60 a 252)+5	(De 60 a 252)+4	(De 60 a 252)+2
Longitud máxima de LOB 1.470.000.000	5	3	2	2	(De 60 a 276)+5	(De 60 a 276)+4	(De 60 a 276)+2
Longitud máxima de LOB 2.147.483.647	5	3	2	2	(De 60 a 312)+5	(De 60 a 312)+4	(De 60 a 312)+2
XML	5	3	-	-	85	84	82

¹ Cuando COMPRESS SYSTEM DEFAULT se ha especificado para la columna.

² Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro.

- **Columnas de dimensión:** Puesto que cada valor diferenciado de una columna de dimensión se asigna a un bloque distinto de la tabla, puede que se desee aplicar

un clúster en una expresión como, por ejemplo, "INTEGER(ORDER_DATE)/100". En este caso, puede definirse una columna generada para la tabla y, a continuación, esta columna generada puede utilizarse en la cláusula ORGANIZE BY DIMENSIONS. Si la expresión es monótonica respecto a una columna de la tabla, DB2 puede utilizar el índice de dimensiones para satisfacer los predicados de rango en esa columna. Por ejemplo, si la expresión es simplemente *nombre-columna + alguna-constante-positiva*, tiene lugar un incremento monótonico. Las funciones definidas por el usuario, determinadas funciones incorporadas y la utilización de más de una columna en una expresión impiden que tenga lugar la monotonidad o su detección.

Las dimensiones que implican columnas generadas cuyas expresiones son no monótonicas o cuya monotonidad no puede determinarse, pueden seguir creándose, pero las consultas de rango, junto con los límites de porción o de célula, de estas dimensiones no reciben soporte. La igualdad y los predicados IN *pueden* procesarse por medio de porciones o células.

Una columna generada será monótonica si se cumple lo siguiente respecto a la función generadora, fn:

- Incremento monótonico.

Para cada par de valores x_1 y x_2 posible, si $x_2 > x_1$, entonces $fn(x_2) > fn(x_1)$. Por ejemplo:

SALARY - 10000

- Reducción monótonica.

Para cada par de valores x_1 y x_2 posible, si $x_2 > x_1$, entonces $fn(x_2) < fn(x_1)$. Por ejemplo:

-SALARY

- Sin reducción monótonica.

Para cada par de valores x_1 y x_2 posible, si $x_2 > x_1$, entonces $fn(x_2) \geq fn(x_1)$. Por ejemplo:

SALARY/1000

- Sin incremento monótonico.

Para cada par de valores x_1 y x_2 posible, si $x_2 > x_1$, entonces $fn(x_2) \leq fn(x_1)$. Por ejemplo:

-SALARY/1000

La expresión "PRICE*DISCOUNT" no es monótonica, porque implica a más de una columna de la tabla.

- **Tablas agrupadas por clústeres de rangos:** La organización de una tabla por secuencia de clave es efectiva para determinados tipos de tablas. La tabla debe tener una clave de entero que esté estrechamente agrupada (densa) sobre el rango de valores posibles. Las columnas de esta clave de entero no se deben poder anular y la clave debe ser lógicamente la clave principal de la tabla. La organización de una tabla agrupada por clústeres de rangos precede la necesidad de disponer de un objeto de índice exclusivo separado, proporcionando acceso directo a la fila para un valor de clave especificado o un rango de filas para un rango especificado de valores de clave. La asignación de todo el espacio correspondiente a un conjunto completo de filas en el rango de secuencia de clave definido se realiza durante la creación de la tabla y se debe tener en cuenta al definir una tabla agrupada por clústeres de rangos. El espacio de almacenamiento no está disponible para ningún otro uso, aunque las filas estén inicialmente marcadas para su supresión. Si el rango completo de secuencia de clave se va a llenar sólo con datos durante un largo período de tiempo, esta organización de tabla probablemente no sea una elección adecuada.
- Una tabla puede tener como máximo una política de seguridad.

CREATE TABLE

- DB2 aplica las restricciones de integridad referencial definidas en las tablas protegidas. En este caso las violaciones de restricción pueden ser difíciles de depurar, ya que DB2 no le permitirá ver qué fila ha ocasionado una violación si no tiene la etiqueta de seguridad o las credenciales de exenciones adecuadas.
- Al definir el orden de las columnas de una tabla, las columnas actualizadas con frecuencia deben colocarse al final de la definición para reducir la cantidad de datos anotados cronológicamente. Esto incluye las columnas ROW CHANGE TIMESTAMP. Se garantiza que las columnas ROW CHANGE TIMESTAMP se actualizarán con cada actualización de fila.
- **Seguridad y duplicación:** La duplicación puede hacer que las filas de datos de una tabla protegida se dupliquen fuera de la base de datos. Tenga cuidado al configurar la duplicación para una tabla protegida, porque DB2 no puede proteger los datos situados fuera de la base de datos.
- **Consideraciones sobre un entorno DB2 pureScale** o de varias particiones:
 - En un entorno DB2 pureScale o de varias particiones, si las opciones CACHE y NO ORDER están en vigor, se pueden activar varias antememorias a la vez y puede que las solicitudes de asignaciones de valor siguiente de los distintos miembros no asignen valores en un orden numérico estricto. Supongamos, por ejemplo, que los miembros DB2A y DB2B están utilizando la misma secuencia y que DB2A obtiene los valores de antememoria del 1 al 20 y DB2B obtiene los valores de antememoria del 21 al 40. En este caso, si DB2A ha solicitado primero el siguiente valor y, a continuación, lo ha solicitado DB2B y, entonces, DB2A lo ha vuelto a solicitar, el orden real de valores asignado sería 1,21,2. Por lo tanto, para garantizar que los números de secuencia se generen en un orden numérico estricto entre varios miembros que utilicen la misma secuencia de forma simultánea, especifique la opción ORDER.
 - En un entorno DB2 pureScale, el uso de la opción ORDER o NO CACHE garantiza que los valores asignados a una secuencia que comparten aplicaciones en varios miembros sigan un orden numérico estricto. En un entorno DB2 pureScale, si se especifica ORDER, la opción NO CACHE es implícita aunque se especifique CACHE *n*.
- **Consideraciones sobre el control de acceso a filas y columnas (RCAC):** las cláusulas ACTIVATE ROW ACCESS CONTROL, ACTIVATE COLUMN ACCESS CONTROL, DEACTIVATE ROW ACCESS CONTROL y DEACTIVATE COLUMN ACCESS CONTROL no reciben soporte. Utilice la sentencia ALTER TABLE para activar o desactivar el control de acceso de nivel de fila o columna en una tabla.
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - La sintaxis siguiente se acepta como comportamiento por omisión:
 - IN nombre-base-de-datos.nombre-espacio-tablas
 - IN DATABASE nombre-base-datos
 - FOR MIXED DATA
 - FOR SBCS DATA
 - Se puede especificar PART en lugar de PARTITION
 - Se puede especificar PARTITION *número-partición* en lugar de PARTITION *nombre-partición*. Un *número-partición* no puede identificar una partición que no se haya especificado anteriormente en la sentencia CREATE TABLE. Si no se especifica un *número-partición*, el gestor de bases de datos genera un número de partición exclusivo.
 - Se puede especificar VALUES en lugar de ENDING AT

- La palabra clave CONSTRAINT puede omitirse en una *definición-columna* que defina a una cláusula de referencias
- El *nombre-restricción* puede especificarse a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT)
- SUMMARY puede especificarse opcionalmente tras CREATE
- DEFINITION ONLY puede especificarse en lugar de WITH NO DATA
- La cláusula PARTITIONING KEY puede especificarse en lugar de la cláusula DISTRIBUTE BY
- Se puede especificar REPLICATED en lugar de DISTRIBUTE BY REPLICATION
- Puede utilizarse una coma para separar varias opciones en la cláusula *opciones-identidad*
- Se puede especificar NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER en lugar de NO MINVALUE, NO MAXVALUE, NO CYCLE, NO CACHE y NO ORDER, respectivamente
- ADD puede especificarse antes de *nombre-opción-tabla constante-serie*.

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 normas descritas para la cláusula *IN nombre-espacio-tablas*.

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 dinámico, suponga que el registro especial CURRENT SCHEMA es JOHNDOE y el registro especial CURRENT PATH tiene el valor por omisión ("SYSIBM","SYSPFUN","JOHNDOE").

Si no se especifica un valor para SALARY, deberá establecerse en 0 y si no se especifica un valor para LIVING_DIST, deberá establecerse en 1 milla (1,6 Km).

```
CREATE TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS
```

```
CREATE 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 ha especificado ningún espacio de tablas para que la tabla se cree en un espacio de tablas que el sistema selecciona basándose en las normas que se describen para la cláusula IN *nombre-espacio-tablas*. Suponga que el registro especial CURRENT PATH tiene el valor por omisión.

```
CREATE TYPE IMAGE AS BLOB (10M)
```

```
CREATE 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 la parte de los datos en ACCOUNTING y la parte del 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.
- El valor de PROJNO debe coincidir con un valor existente para la columna PROJNO de la tabla PROJECT y, si el proyecto se suprime, todas las filas que hacen referencia al proyecto en EMP_ACT también deben suprimirse.

```
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)
REFERENCES PROJECT (PROJNO) ON DELETE CASCADE
)
IN SCHED
```

Se crea automáticamente un índice exclusivo denominado EMP_ACT_UNIQ en el mismo esquema para aplicar 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, así como una descripción. La columna de descripción puede contener 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) )
```

- *Ejemplo 13:* Supongamos que se necesita una tabla de excepción para la tabla EMPLOYEE. Se puede crear una utilizando la sentencia siguiente.

CREATE TABLE

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
(SELECT EMPLOYEE.*,
CURRENT_TIMESTAMP AS TIMESTAMP,
CAST (' ' AS CLOB(32K)) AS MSG
FROM EMPLOYEE
) WITH NO DATA
```

- *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,
...)
```

- *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))
```

- *Ejemplo 17:* Cree una tabla SALES que se distribuya en la columna YEAR y tenga dimensiones en las columnas REGION y YEAR. Los datos se distribuirán por las particiones de base de datos según los valores de generación aleatoria de la columna YEAR. En cada partición de base de datos, los datos se organizarán en

extensiones basadas en combinaciones exclusivas de los valores de las columnas REGION y YEAR de esas particiones de base de datos.

```
CREATE TABLE SALES
(CUSTOMER          VARCHAR(80),
 REGION           CHAR(5),
 YEAR             INTEGER)
DISTRIBUTE BY HASH (YEAR)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
```

- *Ejemplo 18:* Cree una tabla SALES con la columna PURCHASEYEARMONTH que se genera a partir de la columna PURCHASEDATE. Utilice una expresión para crear una columna que sea monótonica con respecto a la columna PURCHASEDATE original y, por lo tanto, que sea adecuada para utilizarla como dimensión. La tabla se distribuye en la columna REGION y cada partición de base de datos se organiza en extensiones según la columna PURCHASEYEARMONTH; es decir, las distintas regiones estarán en particiones de base de datos diferentes y los distintos meses de compras pertenecerán a diferentes celdas (o conjuntos de extensiones) de esas particiones de base de datos.

```
CREATE TABLE SALES
(CUSTOMER          VARCHAR(80),
 REGION           CHAR(5),
 PURCHASEDATE     DATE,
 PURCHASEYEARMONTH INTEGER
                GENERATED ALWAYS AS (INTEGER(PURCHASEDATE)/100))
DISTRIBUTE BY HASH (REGION)
ORGANIZE BY DIMENSIONS (PURCHASEYEARMONTH)
```

- *Ejemplo 19:* Cree la tabla CUSTOMER con la columna CUSTOMERNUMDIM que se genera a partir de la columna CUSTOMERNUM. Utilice una expresión para crear una columna que sea monótonica con respecto a la columna CUSTOMERNUM original y, por lo tanto, que sea adecuada para utilizarla como dimensión. La tabla se organiza en celdas según la columna CUSTOMERNUMDIM, de modo que hay una celda diferente en la tabla para cada 50 clientes. Si se crease un índice exclusivo en CUSTOMERNUM, los números de cliente se contendrían en clústeres de tal manera que, cada conjunto de 50 valores se encontraría en un conjunto de extensiones en particular de la tabla.

```
CREATE TABLE CUSTOMER
(CUSTOMERNUM      INTEGER,
 CUSTOMERNAME     VARCHAR(80),
 ADDRESS          VARCHAR(200),
 CITY             VARCHAR(50),
 COUNTRY          VARCHAR(50),
 CODE             VARCHAR(15),
 CUSTOMERNUMDIM   INTEGER
                GENERATED ALWAYS AS (CUSTOMERNUM/50))
ORGANIZE BY DIMENSIONS (CUSTOMERNUMDIM)
```

- *Ejemplo 20:* cree una tabla base remota denominada EMPLOYEE en el servidor Oracle, ORASERVER. También se creará automáticamente un apodo, denominado EMPLOYEE, que hará referencia a esta tabla base remota que acaba de crear.

```
CREATE TABLE EMPLOYEE
(EMP_NO          CHAR(6)          NOT NULL,
 FIRST_NAME     VARCHAR(12)     NOT NULL,
 MID_INT        CHAR(1)          NOT NULL,
 LAST_NAME      VARCHAR(15)     NOT NULL,
 HIRE_DATE      DATE,
 JOB            CHAR(8),
 SALARY         DECIMAL(9,2),
 PRIMARY KEY (EMP_NO))
```

CREATE TABLE

```
OPTIONS
(REMOTE_SERVER 'ORASERVER',
 REMOTE_SCHEMA 'J15USER1',
 REMOTE_TABNAME 'EMPLOYEE')
```

Las sentencias CREATE TABLE siguientes muestran cómo especificar el nombre de tabla, o el nombre de tabla y el nombre de tabla base remota explícito, para obtener las mayúsculas o minúsculas que se requieran. El identificador en minúsculas, employee, se utiliza para ilustrar la conversión implícita de identificadores.

Cree una tabla base remota denominada EMPLOYEE (en mayúsculas) en un servidor Informix y cree un apodo denominado EMPLOYEE (en mayúsculas) en esa tabla:

```
CREATE TABLE employee
(EMP_NO CHAR(6) NOT NULL,
 ... )
OPTIONS
(REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* no está delimitado, el nombre de tabla base remota aparecerá en letras mayúsculas, incluso si la fuente de datos remota almacena los nombres en minúsculas.

Cree una tabla base remota denominada employee (en minúsculas) en un servidor Informix y cree un apodo denominado EMPLOYEE (en mayúsculas) en esa tabla:

```
CREATE TABLE employee
(EMP_NO CHAR(6) NOT NULL,
 ... )
OPTIONS
(REMOTE_SERVER 'INFX_SERVER',
 REMOTE_TABNAME 'empToyee')
```

Al crear una tabla en una fuente de datos remota que da soporte a los identificadores delimitados, utilice la opción REMOTE_TABNAME y una constante de serie de caracteres que especifique el nombre de tabla en mayúsculas o minúsculas, según se requiera.

Cree una tabla base remota denominada employee (en minúsculas) en un servidor Informix y cree un apodo denominado employee (en minúsculas) en esa tabla:

```
CREATE TABLE "employee"
(EMP_NO CHAR(6) NOT NULL,
 ... )
OPTIONS
(REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* está delimitado, el nombre de tabla base remota será idéntico a *nombre-tabla*.

- *Ejemplo 21:* Cree una tabla agrupada por clústeres de rangos que se pueda utilizar para localizar un estudiante utilizando un ID de estudiante. Para cada registro de estudiante, incluya el ID de escuela, ID de programa, número de estudiante, ID de estudiante, nombre de estudiante, apellido de estudiante y promedio de puntos del curso de estudiante (GPA).

```
CREATE TABLE STUDENTS
(SCHOOL_ID    INTEGER NOT NULL,
 PROGRAM_ID   INTEGER NOT NULL,
 STUDENT_NUM  INTEGER NOT NULL,
 STUDENT_ID   INTEGER NOT NULL,
 FIRST_NAME   CHAR(30),
 LAST_NAME    CHAR(30),
```

```

GPA          DOUBLE)
ORGANIZE BY KEY SEQUENCE
(STUDENT_ID
  STARTING FROM 1
  ENDING AT 1000000)
DISALLOW OVERFLOW

```

El tamaño de cada registro es la suma de las columnas, más alineación, más cabecera de fila de tabla agrupada por clústeres de rangos. En este caso, el tamaño de fila es de 98 bytes: 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (para columnas con posibilidad de nulos) + 1 (para alineación) + 10 (para la cabecera). Con un tamaño de página de 4 KB (o 4096 bytes), después de contabilizar la actividad general de página, hay 4038 bytes disponibles, lo que constituye espacio suficiente para 41 registros por página. Si se permiten 1 millón de registros de estudiantes, se necesitan (1 millón dividido por 41 registros por página) 24.391 páginas. Con dos páginas adicionales por actividad general de tabla, el número final de páginas de 4-KB que se asignan cuando se crea la tabla es 24.393.

- *Ejemplo 22:* Cree una tabla denominada DEPARTMENT con una dependencia funcional que no tenga especificada ningún nombre de restricción.

```

CREATE TABLE DEPARTMENT
(DEPTNO      SMALLINT      NOT NULL,
DEPTNAME    VARCHAR(36)  NOT NULL,
MGRNO       CHAR(6),
ADMRDEPT    SMALLINT      NOT NULL,
LOCATION     CHAR(30),
CHECK (DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED)

```

- *Ejemplo 23:* Cree una tabla con filas protegidas.

```

CREATE TABLE TOASTMASTERS
(PERFORMANCE DB2SECURITYLABEL,
POINTS      INTEGER,
NAME        VARCHAR(50))
SECURITY POLICY CONTRIBUTIONS

```

- *Ejemplo 24:* Cree una tabla con columnas protegidas.

```

CREATE TABLE TOASTMASTERS
(PERFORMANCE CHAR(8),
POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
NAME        VARCHAR(50))
SECURITY POLICY CONTRIBUTIONS

```

- *Ejemplo 25:* Cree una tabla con filas y columnas protegidas.

```

CREATE TABLE TOASTMASTERS
(PERFORMANCE DB2SECURITYLABEL,
POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
NAME        VARCHAR(50))
SECURITY POLICY CONTRIBUTIONS

```

- *Ejemplo 26:* Los objetos grandes de una tabla particionada residen, por omisión, en el mismo espacio de tablas que los datos. Este uso por omisión se puede alterar temporalmente utilizando la cláusula LONG IN para especificar uno o varios espacios de tablas para los objetos grandes. Cree una tabla denominada DOCUMENTS cuyos datos de objetos grandes se almacenarán (de modo rotativo para cada partición de datos) en los espacios de tablas TBSP1 y TBSP2.

```

CREATE TABLE DOCUMENTS
(ID          INTEGER,
CONTENTS   CLOB)
LONG IN TBSP1, TBSP2
PARTITION BY RANGE (ID)
(STARTING 1 ENDING 1000
  EVERY 100)

```

CREATE TABLE

De modo alternativo, puede emplear el formato largo de la sintaxis para identificar explícitamente un espacio de tablas grande para cada partición de datos. En este ejemplo, los datos CLOB de la primera partición de datos se colocan en LARGE_TBSP3, y los datos CLOB de las demás particiones de datos se reparten entre LARGE_TBSP1 y LARGE_TBSP2 de modo rotativo.

```
CREATE TABLE DOCUMENTS
  (ID          INTEGER,
   CONTENTS   CLOB)
LONG IN LARGE_TBSP1, LARGE_TBSP2
PARTITION BY RANGE (ID)
  (STARTING 1 ENDING 100
   IN TBSP1 LONG IN LARGE_TBSP3,
   STARTING 101 ENDING 1000
   EVERY 100)
```

- *Ejemplo 27:* Cree una tabla particionada denominada ACCESSNUMBERS con dos particiones de datos. La fila (10, NULL) debe colocarse en la primera partición y la fila (NULL, 100) debe colocarse en la segunda partición de datos (la última).

```
CREATE TABLE ACCESSNUMBERS
  (AREA INTEGER,
   EXCHANGE INTEGER)
PARTITION BY RANGE (AREA NULLS LAST, EXCHANGE NULLS FIRST)
  (STARTING (1,1) ENDING (10,100),
   STARTING (11,1) ENDING (MAXVALUE,MAXVALUE))
```

Como los valores nulos de la segunda columna se clasificarían en primer lugar, la fila (11, NULL) se clasificaría debajo del límite inferior de la última partición de datos (11, 1); el intento de insertar esta fila devuelve un error. La fila (12, NULL) quedaría en la última partición de datos.

- *Ejemplo 28:* Cree una tabla denominada RATIO con una sola partición de datos y la columna de particionamiento PERCENT.

```
CREATE TABLE RATIO
  (PERCENT INTEGER)
PARTITION BY RANGE (PERCENT)
  (STARTING (MINVALUE) ENDING (MAXVALUE))
```

Esta definición de tabla permite insertar cualquier valor entero para la columna PERCENT. La definición siguiente de la tabla RATIO permite insertar en la columna PERCENT cualquier valor entero entre 1 y 100, ambos incluidos.

```
CREATE TABLE RATIO
  (PERCENT INTEGER)
PARTITION BY RANGE (PERCENT)
  (STARTING 0 EXCLUSIVE ENDING 100 INCLUSIVE)
```

- *Ejemplo 29:* Cree una tabla denominada MYDOCS con dos columnas: una es un identificador, y la otra almacena documentos XML.

```
CREATE TABLE MYDOCS
  (ID          INTEGER,
   DOC XML)
IN HLTBSPACE
```

- *Ejemplo 30:* Cree una tabla denominada NOTES con cuatro columnas, una de ellas para almacenar notas XML.

```
CREATE TABLE NOTES
  (ID          INTEGER,
   DESCRIPTION VARCHAR(255),
   CREATED     TIMESTAMP,
   NOTE        XML)
```

- *Ejemplo 31:* cree una tabla, EMP_INFO, que contenga un número de teléfono y una dirección para cada empleado. Incluya una columna ROW CHANGE TIMESTAMP en la tabla para seguir la modificación de la información del empleado.

```
CREATE TABLE EMP_INFO
(EMPNO          CHAR(6) NOT NULL,
 EMP_INFOCHANGE  TIMESTAMP NOT NULL GENERATED ALWAYS
 FOR EACH ROW ON UPDATE
 AS ROW CHANGE TIMESTAMP,
 EMP_ADDRESS     VARCHAR(300),
 EMP_PHONENO     CHAR(4),
 PRIMARY KEY (EMPNO) )
```

- *Ejemplo 32:* Cree una tabla particionada denominada DOCUMENTS con dos particiones de datos:
 - El objeto de datos de la primera partición reside en el espacio de tablas TBSP11. La partición de índice particionado de la partición reside en el espacio de tablas TBSP21. El objeto de datos XML reside en el espacio de tablas TBSP31.
 - El objeto de datos de la segunda partición reside en el espacio de tablas TBSP12. La partición de índice particionado de la partición reside en el espacio de tablas TBSP22. El objeto de datos XML reside en el espacio de tablas TBSP32.

La cláusula INDEX IN de nivel de tabla no tiene ningún impacto en la selección de espacio de tablas para los índices particionados.

```
CREATE TABLE DOCUMENTS
(ID          INTEGER,
 CONTENTS XML) INDEX IN TBSPX
PARTITION BY (ID NULLS LAST)
(STARTING FROM 1 INCLUSIVE ENDING AT 100 INCLUSIVE
 IN TBSP11 INDEX IN TBSP21 LONG IN TBSP31,
 STARTING FROM 101 INCLUSIVE ENDING AT 200 INCLUSIVE
 IN TBSP21 INDEX IN TBSP22 LONG IN TBSP32)
```

- *Ejemplo 33:* Cree una tabla particionada denominada SALES con dos particiones de datos:
 - El objeto de datos de la primera partición reside en el espacio de tablas TBSP11. La partición de índice particionado de la partición reside en el espacio de tablas TBSP21.
 - El objeto de datos de la segunda partición reside en el espacio de tablas TBSP12. El objeto de índice particionado reside en el espacio de tablas TBSP22.

La cláusula INDEX IN de nivel de tabla no tiene ningún impacto en la selección de espacio de tablas para los índices particionados.

```
CREATE TABLE SALES
(SID          INTEGER,
 AMOUNT INTEGER) INDEX IN TBSPX
PARTITION BY RANGE (SID NULLS LAST)
(STARTING FROM 1 INCLUSIVE ENDING AT 100 INCLUSIVE
 IN TBSP11 INDEX IN TBSP21,
 STARTING FROM 101 INCLUSIVE ENDING AT 200 INCLUSIVE
 IN TBSP12 INDEX IN TBSP22)
```

- *Ejemplo 34:* Cree una tabla denominada BOOKS con cuatro columnas, incluida una llamada DATE_ADDED, que inserta la indicación de fecha y hora actual por omisión.

CREATE TABLE

```
CREATE TABLE BOOKS
  (ISBN_NUM  INTEGER,
   TITLE     VARCHAR(255),
   AUTHOR    VARCHAR(255),
   DATE_ADDED  TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP)
```

- *Ejemplo 35:* Cree una tabla Unicode denominada STUDENTS en una base de datos que no es Unicode. Supongamos que la base de datos se ha creado con el conjunto de códigos 1252 y el territorio CA, y que el parámetro de configuración de base de datos ALT_COLLATE se ha actualizado a IDENTITY_16BIT.

```
CREATE TABLE STUDENTS (
  STUDENTID INT NOT NULL,
  FAMILY_NAME VARCHAR(36) NOT NULL,
  GIVEN_NAME VARCHAR(36) NOT NULL,
  PRIMARY KEY(STUDENTID))
CCSID UNICODE
```

- *Ejemplo 36:* cree una tabla con el nombre TDEPT_TEMP, basada en la tabla TDEPT que se ha creado en el Ejemplo 1.

```
CREATE TABLE TDEPT_TEMP LIKE TDEPT
```

La tabla TDEPT_TEMP tendrá la misma definición que TDEPT, excepto que la clave primaria no se definirá y que se elegirá implícitamente un espacio de tablas por omisión.

- *Ejemplo 37:* Etiquetas de seguridad de columna heredadas por una tabla de consulta materializada.

```
CREATE SECURITY LABEL COMPONENT level_array ARRAY ['A', 'B', 'C']

CREATE SECURITY POLICY P COMPONENTS level_array WITH DB2LBACRULES

CREATE SECURITY LABEL P.A COMPONENT level_array 'A'

CREATE SECURITY LABEL P.B COMPONENT level_array 'B'

CREATE SECURITY LABEL P.C COMPONENT level_array 'C'

CREATE TABLE t1 (c1 INT, c2 INT SECURED WITH B, c3 REAL SECURED WITH A)
SECURITY POLICY P

CREATE TABLE t2 (c4 REAL, c5 INT SECURED WITH C, c6 DB2SECURITYLABEL)
SECURITY POLICY P
```

Genere una tabla de consulta materializada

```
CREATE TABLE m1 AS(SELECT c1, c3, c5, c6 FROM t1,t2 WHERE c2 !=100)
DATA INITALLY DEFERRED REFRESH DEFERRED
```

La etiqueta de seguridad de t1.c2 se utiliza para calcular las etiquetas de seguridad de todas las columnas de m1 porque aparece en los predicados de la consulta. Las propiedades de control de acceso basado en etiquetas de la tabla de consulta materializada m1 son:

- Política de seguridad = P
- Etiqueta de seguridad de la columna m1.c1 = P.B
- Etiqueta de seguridad de la columna m1.c3 = P.A
- Etiqueta de seguridad de la columna m1.c5 = P.B
- Etiqueta de seguridad de la columna m1.c6 = P.B y también DB2SECURITYLABEL.

- *Ejemplo 38:* Una tabla de etapas para una tabla de consulta materializada está protegida con control de acceso basado en etiquetas. Siguiendo el Ejemplo 33, si la tabla de etapas st1 está definida como:

```
CREATE TABLE st1 FOR m1 PROPAGATE IMMEDIATE
```

Las propiedades de control de acceso basado en etiquetas de la tabla de etapas st1 son:

- Política de seguridad = P
- Etiqueta de seguridad de la columna st1.c1 = P.B
- Etiqueta de seguridad de la columna st1.c3 = P.A
- Etiqueta de seguridad de la columna st1.c5 = P.B
- Etiqueta de seguridad de la columna st1.c6 = P.B y también DB2SECURITYLABEL.

CREATE TABLESPACE

La sentencia CREATE TABLESPACE define un nuevo espacio de tablas en la base de datos, asigna contenedores al espacio de tablas y registra la definición y los atributos del espacio de tablas en el catálogo.

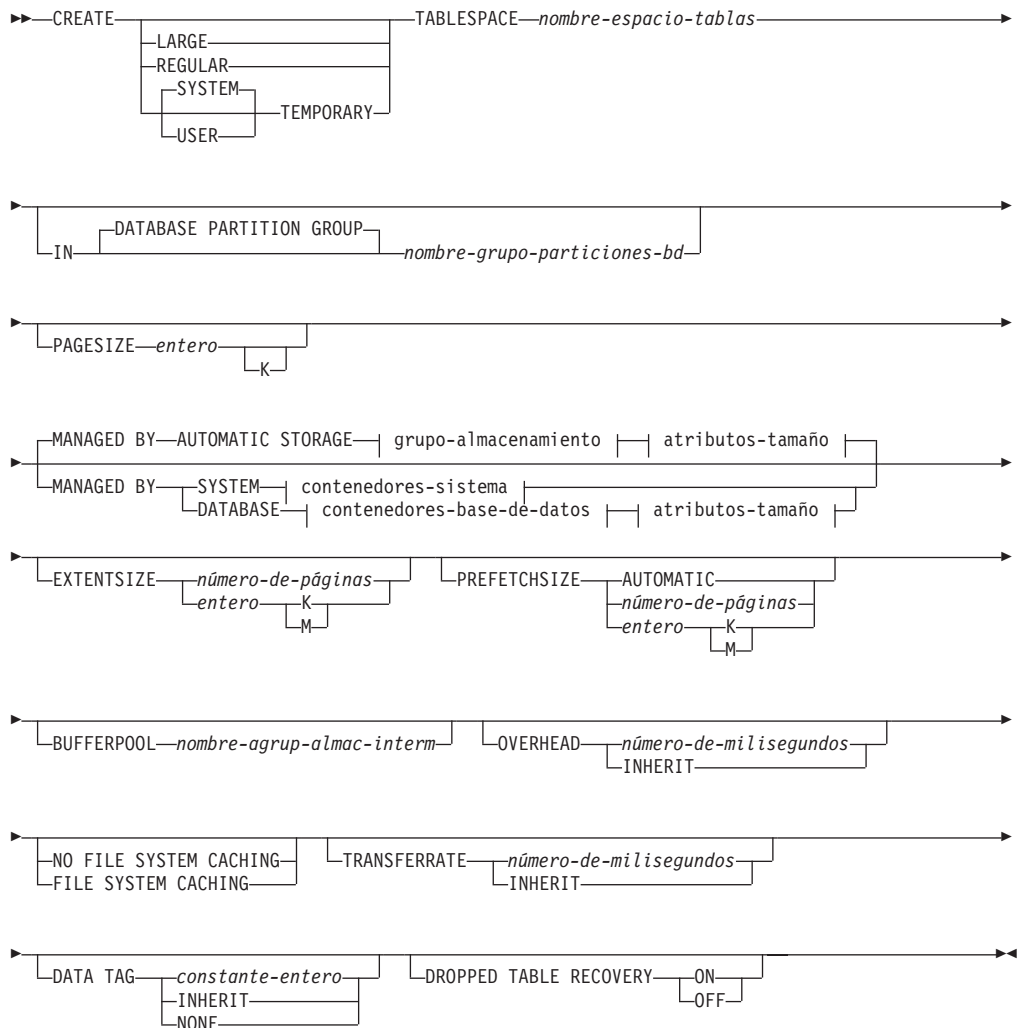
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

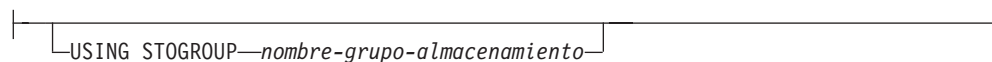
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

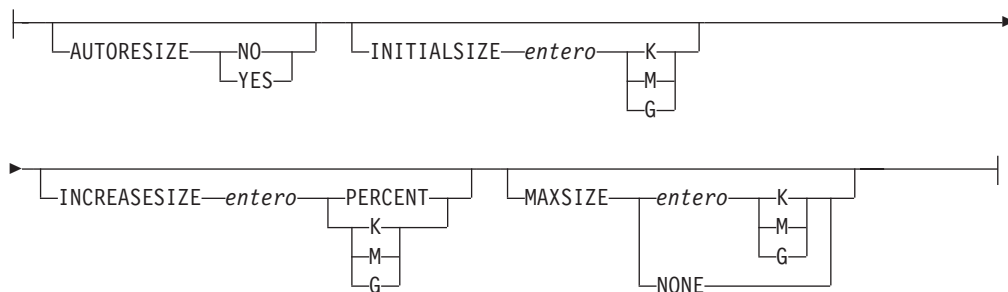
Sintaxis



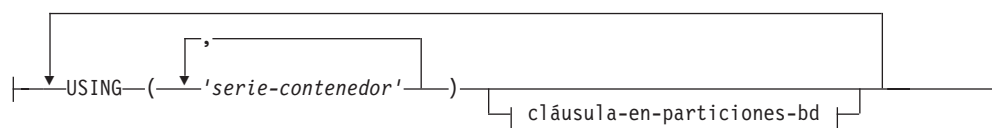
grupo-almacenamiento:



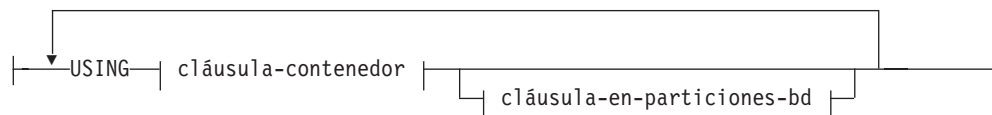
atributos-tamaño:



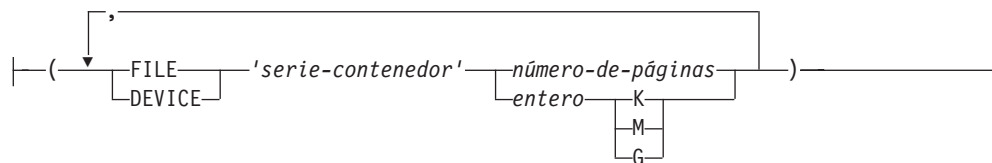
contenedores-sistema:



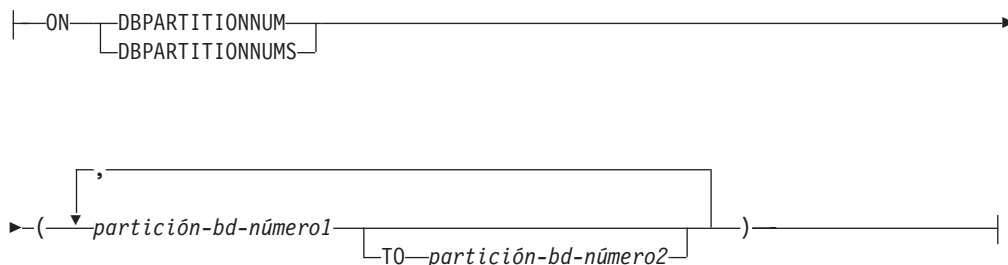
contenedores-basedatos:



cláusula-contenedor:



cláusula-en-particiones-bd:



CREATE TABLESPACE

Descripción

LARGE, REGULAR, SYSTEM TEMPORARY o USER TEMPORARY

Especifica el tipo de espacio de tablas que se creará. Si no se especifica ningún tipo, la cláusula **MANAGED BY** determina el valor por omisión.

LARGE

Almacena todos los datos permanentes. Este tipo sólo se permite en espacios de tablas de espacio gestionado por la base de datos (DMS). También es el tipo por omisión de los espacios de tablas DMS si no se especifica ningún tipo. Si una tabla se coloca en un espacio de tablas grande:

- La tabla puede ser mayor que una tabla de un espacio de tablas normal. Para obtener detalles sobre los límites de tabla y de espacio de tablas, consulte “Límites de SQL y XML”.
- La tabla puede dar soporte a más de 255 filas por página de datos, lo que puede mejorar la utilización del espacio de las páginas de datos.
- Los índices definidos en la tabla requerirán una entrada de dos bytes por fila adicional, a diferencia de los índices definidos en una tabla que reside en un espacio de tablas normal.

REGULAR

Almacena todos los datos permanentes. Este tipo se aplica a espacios de tablas DMS y SMS. Éste es el único tipo permitido en los espacios de tablas SMS y también es el tipo por omisión para espacios de tablas SMS si no se ha especificado ningún tipo.

SYSTEM TEMPORARY

Almacena temporalmente tablas, áreas de trabajo utilizadas por el gestor de bases de datos para llevar a cabo operaciones como clasificaciones o uniones. Una base de datos siempre debe tener al menos un espacio de tablas **SYSTEM TEMPORARY**, puesto que las tablas temporales sólo se pueden almacenar en un espacio de tablas de este tipo. Al crear una base de datos, se crea automáticamente un espacio de tablas temporal.

USER TEMPORARY

Almacena tablas temporales creadas y tablas declaradas temporales. Cuando se crea una base de datos, no existen espacios de tablas temporales de usuario. Para permitir la definición de tablas temporales creadas o tablas temporales declaradas, al menos debe crearse un espacio de tablas temporal de usuario con los privilegios **USE** adecuados.

nombre-espacio-tablas

Nombra el espacio de tablas. Este nombre consta de una sola parte. Se trata de 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 DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*

Especifica el grupo de particiones de base de datos para el espacio de tablas. El grupo de particiones de base de datos debe existir. El único grupo de particiones de base de datos que se puede especificar al crear un espacio de tablas **SYSTEM TEMPORARY** es **IBMTEMPGROUP**. Las palabras clave **DATABASE PARTITION GROUP** son opcionales.

Si no se especifica el grupo de particiones de base de datos, se utiliza el grupo de particiones de base de datos por omisión (**IBMDEFAULTGROUP**) para los

espacios de tablas REGULAR, LARGE y USER TEMPORARY. Para los espacios de tablas SYSTEM TEMPORARY, se utiliza el grupo de particiones de base de datos 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 o 32.768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 o 32. Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE) o si el tamaño de página no es el mismo que el de la agrupación de almacenamientos intermedios asociada al espacio de tablas (SQLSTATE 428CB).

El valor por omisión se proporciona mediante el parámetro de configuración de base de datos **tamaño-página**, que se establece cuando se crea la base de datos.

MANAGED BY AUTOMATIC STORAGE

Especifica que el espacio de tablas debe ser un espacio de tablas de almacenamiento automático. Si no existen grupos de almacenamiento definidos, se devuelve un error (SQLSTATE 55060).

El gestor de bases de datos decide cómo se crea el espacio de tablas de almacenamiento automático inicialmente. Los espacios de tablas temporales se inician como espacios de tablas de espacio gestionado por el sistema (SMS) y los espacios de tablas permanentes se inician como espacios de tablas de espacio gestionado por la base de datos (DMS). Cuando se crea un espacio de tablas permanente y no se especifica el tipo del espacio de tablas, el comportamiento por omisión consiste en crear un espacio de tablas grande. Con un espacio de tablas de almacenamiento automático, el gestor de bases de datos determina qué contenedores deben asignarse al espacio de tablas, basándose en las vías de acceso de almacenamiento asociadas al grupo de almacenamiento que utiliza el espacio de tablas.

grupo-almacenamiento

Especifique el grupo de almacenamiento para un espacio de tablas de almacenamiento automático.

USING STOGROUP

Para un espacio de tablas de almacenamiento automático, identifica el grupo de almacenamiento para el espacio de tablas en el que se almacenarán los datos de espacio de tablas. Si no se especifica un *nombre-grupo-almacenamiento*, se utiliza el grupo de almacenamiento por omisión designado actualmente. Esta cláusula sólo se aplica a espacios de tablas de almacenamiento automático (SQLSTATE 42613).

nombre-grupo-almacenamiento

Identifica el grupo de almacenamiento en el que se almacenarán los datos de espacio de tablas. *nombre-grupo-almacenamiento* debe identificar un grupo de almacenamiento que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte.

atributos-tamaño

Especifique los atributos de tamaño para un espacio de tablas de almacenamiento automático o para un espacio de tablas DMS que no sea de almacenamiento automático. Los espacios de tablas SMS no permiten cambiar el tamaño automáticamente.

AUTORESIZE

Especifica si debe habilitarse o no la posibilidad de cambiar

CREATE TABLESPACE

automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático. Los espacios de tablas de redimensionamiento automático aumentan automáticamente de tamaño cuando se llenan. El valor por omisión es NO para los espacios de tablas DMS y YES para los espacios de tablas de almacenamiento automático.

NO Especifica que debe inhabilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

YES

Especifica que debe habilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

INITIALSIZE *entero* **K | M | G**

Especifica el tamaño inicial, por partición de base de datos, de un espacio de tablas de almacenamiento automático. Esta opción sólo es válida para espacios de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un tamaño coherente en los contenedores del espacio de tablas. Además, si el espacio de tablas permite cambiar el tamaño automáticamente y el tamaño inicial no es lo suficientemente grande para contener los metadatos que se deben añadir al nuevo espacio de tablas, el gestor de bases de datos seguirá ampliando el espacio de tablas por el valor de INCREASESIZE hasta que haya espacio suficiente. Si no se especifica la cláusula INITIALSIZE, el gestor de bases de datos determinará un valor apropiado. El valor para *entero* debe ser como mínimo 48 K.

INCREASESIZE *entero* **PERCENT** o **INCREASESIZE** *entero* **K | M | G**

Especifica la cantidad, por partición de base de datos, en la que aumentará automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente, cuando se llene el espacio de tablas y se haya efectuado una petición de espacio. El valor entero debe ir seguido de:

- PERCENT para especificar la cantidad como porcentaje del tamaño de espacio de tablas en el momento en el que se efectúe una petición de espacio. Cuando se especifique PERCENT, el valor entero debe estar entre 0 y 100 (SQLSTATE 42615).
- K (de kilobytes), M (de megabytes) o G (de gigabytes) para especificar la cantidad en bytes.

Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior o superior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas. Si el espacio de tablas permite cambiar el tamaño automáticamente, pero no se ha especificado la cláusula INCREASESIZE, el gestor de bases de datos determinará un valor apropiado.

MAXSIZE *entero* **K | M | G** o **MAXSIZE NONE**

Especifica el tamaño máximo hasta el que se puede aumentar automáticamente un espacio de tablas habilitado para redimensionamiento automático. Si el espacio de tablas permite cambiar el tamaño automáticamente, pero no se ha especificado la cláusula MAXSIZE, el valor por omisión es NONE.

entero

Especifica un límite fijo sobre el tamaño, por partición de base de

datos, hasta el cual puede aumentar automáticamente un espacio de tablas DMS o un espacio de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

NONE

Especifica que debe permitirse incrementar el espacio de tablas hasta la capacidad del sistema de archivos o hasta el tamaño máximo del espacio de tablas (descrito en “Límites de SQL y XML”).

MANAGED BY SYSTEM

Especifica que el espacio de tablas debe ser un espacio de tablas SMS.

MANAGED BY SYSTEM no se puede especificar en un entorno DB2 pureScale (SQLSTATE 42997).

Importante: El tipo de espacio de tablas SMS ha quedado en desuso en la Versión 10.1 para los espacios de tablas permanentes definidos por el usuario y podría eliminarse en un release futuro. El tipo de espacio de tablas para espacios de tablas temporales y catálogos no ha quedado en desuso. Para obtener más información, consulte “Los espacios de tablas permanentes SMS han quedado en desuso” en *Novedades en DB2 Versión 10.1*

contenedores-sistema

Especifique los contenedores para un espacio de tablas SMS.

USING (*'serie-contenedor',...*)

Para un espacio de tablas SMS, identifica uno o varios contenedores que pertenecerán al espacio de tablas y en el que se almacenarán los datos del espacio de tablas. La variable *serie-contenedor* no puede superar los 240 de longitud.

Cada *serie-contenedor* puede ser un nombre de directorio absoluto o relativo.

El nombre del directorio, en el caso de que no sea absoluto, es relativo al directorio de la base de datos y puede ser un alias de nombre de vía de acceso (un enlace simbólico en sistemas UNIX) a almacenamiento que no está físicamente asociado al directorio de la base de datos. Por ejemplo, *dirbd/work/c1* podría ser un enlace simbólico a otro sistema de archivos.

Si algún componente del nombre del directorio no existe, el gestor de bases de datos lo crea. Cuando se descarta un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos. Si el directorio identificado por *serie-contenedor* existe, no debe contener ningún archivo ni subdirectorio (SQLSTATE 428B2).

El formato de *serie-contenedor* depende del sistema operativo. En sistemas operativos Windows, un nombre de vía de acceso de directorio absoluto comienza por una letra de unidad y un signo de dos puntos (:); en sistemas UNIX, comienza por una barra inclinada (/). Un nombre de vía de acceso relativo en cualquier plataforma no comienza por ningún carácter que dependa del sistema operativo.

En la actualidad sólo se da soporte a recursos remotos (como, por ejemplo, unidades redirigidas a LAN o sistemas de archivos montados en NFS) cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o

CREATE TABLESPACE

S4100, o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que NEC Storage NS Series sólo recibe soporte con el uso de una fuente de alimentación ininterrumpible (UPS); se recomienda utilizar UPS continuo (no en espera). Un sistema de archivos montado en NFS en AIX se debe montar en modalidad ininterrumpible utilizando la opción **-o nointr**.

cláusula-en-particiones-bd

Especifica las particiones de base de datos 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 de base de datos en el grupo de particiones de base de datos que no están especificadas explícitamente en ninguna otra *cláusula-en-particiones-bd*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, si la *cláusula-en-particiones-bd* no está especificada, los contenedores también se crearán en todas las particiones de base de datos nuevas añadidas a la base de datos.

MANAGED BY DATABASE

Especifica que el espacio de tablas debe ser un espacio de tablas DMS. Si no está especificado el tipo de espacio de tablas, el comportamiento por omisión es crear un espacio de tablas grande.

MANAGED BY DATABASE no se puede especificar en un entorno DB2 pureScale (SQLSTATE 42997).

Importante: A partir de la Versión 10.1 Fixpack 1, el tipo de espacio de tablas DMS queda en desuso para los espacios de tablas permanentes definidos por el usuario y podría eliminarse en un release futuro. El tipo de espacio de tablas DMS para espacios de tablas temporales y catálogos no ha quedado en desuso. Para obtener más información, consulte el tema sobre los espacios de tablas permanentes DMS que han quedado en desuso en *Novedades en DB2 Versión 10.1*.

contenedores-basedatos

Especifica los contenedores de un espacio de tablas DMS.

USING

Especifica una cláusula-contenedor.

cláusula-contenedor

Especifica los contenedores de un espacio de tablas DMS.

(FILE|DEVICE 'serie-contenedor' número-de-páginas,...)

Para un espacio de tablas DMS, identifica uno o varios 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 variable *serie-contenedor* no puede superar los 254 de longitud.

En el caso de un contenedor de tipo FILE, *serie-contenedor* debe ser un nombre de archivo absoluto o relativo. El nombre de archivo, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre del directorio no existe, el gestor de bases de

datos lo crea. Si el archivo no existe, el gestor de bases de datos lo creará y lo inicializará en el tamaño especificado. Cuando se descarta un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos.

Nota: Si el archivo existe, se sobregraba, y si es menor de lo especificado, se amplía. El archivo no se truncará si es más grande de lo que se ha especificado.

Para un contenedor de tipo DEVICE, *serie-contenedor* debe ser un nombre de dispositivo. El dispositivo ya debe existir.

Todos los contenedores deben ser exclusivos en todas las bases de datos. Un contenedor sólo puede pertenecer a un espacio de tablas. El tamaño de los contenedores puede diferir; sin embargo, se consigue un rendimiento óptimo cuando todos los contenedores tienen el mismo tamaño. El formato exacto de *serie-contenedor* depende del sistema operativo.

En la actualidad sólo se da soporte a recursos remotos (como, por ejemplo, unidades redirigidas a LAN o sistemas de archivos montados en NFS) cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o S4100, o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que NEC Storage NS Series sólo recibe soporte con el uso de una fuente de alimentación ininterrumpible (UPS); se recomienda utilizar UPS continuo (no en espera).

cláusula-en-particiones-bd

Especifica las particiones de base de datos 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 de base de datos en el grupo de particiones de base de datos que no están especificadas explícitamente en ninguna otra *cláusula-en-particiones-bd*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, si la *cláusula-en-particiones-bd* no está especificada, los contenedores también se crearán en todas las particiones de base de datos nuevas añadidas a la base de datos.

cláusula-en-particiones-bd

Especifica las particiones de base de datos en las que se crean contenedores en una base de datos particionada.

ON DBPARTITIONNUMS

Palabras clave que indican que las particiones individuales de la base de datos están especificadas. DBPARTITIONNUM es sinónimo de DBPARTITIONNUMS.

part-bd-núm1

Especifique un número de partición de base de datos.

TO *partición-bd-núm-2*

Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9). Los contenedores se crean en cada una de las particiones de base de datos entre los valores especificados e incluidos estos. Una partición de base de datos especificada debe encontrarse en el grupo de particiones de base de datos del espacio de tablas.

CREATE TABLESPACE

La partición de base de datos especificada por el número y cada una de las particiones de base de datos que hay dentro de un rango especificado deben existir en el grupo de particiones de base de datos del espacio de tablas (SQLSTATE 42729). Un número de partición de base de datos sólo puede aparecer explícitamente o dentro de un rango en exactamente una *cláusula-en-particiones-bd* de 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 del tamaño de extensión también puede especificarse como un valor entero seguido de K (para kilobytes) o de M (para megabytes). Si se especifica de este modo, se utiliza el límite inferior del número de bytes dividido por el tamaño de página para determinar el valor del tamaño de extensión. El gestor de bases de datos pasa periódicamente por los contenedores a medida que se almacenan datos.

En un entorno DB2 pureScale, debe utilizar un tamaño de extensión de 32 páginas como máximo. Este tamaño de extensión mínimo reduce la cantidad de tráfico de mensajes internos dentro del entorno DB2 pureScale cuando se añaden extensiones para una tabla o índice.

El valor por omisión lo proporciona el parámetro de configuración de la base de datos **dft_extent_sz**, que tiene un rango válido de 2 a 256 páginas.

PREFETCHSIZE

Especifica que han de leerse los datos que una consulta necesita antes de que la consulta haga referencia a éstos con el fin de que la consulta no tenga que esperar a que se produzca la E/S.

El valor por omisión lo proporciona el parámetro de configuración de la base de datos **dft_prefetch_sz**.

AUTOMATIC

Especifica que el tamaño de captación previa de un espacio de tablas debe actualizarse automáticamente, es decir, el tamaño de captación previa se gestionará a través del gestor de bases de datos de DB2.

DB2 actualizará el tamaño de captación previa automáticamente siempre que el número de contenedores de un espacio de tablas cambie (seguido por una ejecución satisfactoria de una sentencia ALTER TABLESPACE que añada o descarte uno o más contenedores). El tamaño de captación previa se actualiza al arrancar la base de datos.

número-de-páginas

Especifica el número de páginas PAGESIZE del espacio de tablas que se leerán cuando se realice la captación previa de datos. El valor máximo es 32767.

entero K | M

Especifica los valores de tamaño de captación previa como valor de entero seguido de K (de kilobytes) o de M (de megabytes). Si se especifica de este modo, se utiliza el límite inferior del número de bytes dividido por el tamaño de página para determinar el valor correspondiente al número de páginas para el tamaño de captación previa.

BUFFERPOOL *nombre-agrup-almac-interm*

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 tomado por omisión) para el espacio de tablas (SQLSTATE 428CB). Debe haberse definido el grupo de particiones de base de datos del espacio de tablas para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos* u OVERHEAD INHERIT

Especifica la actividad general del controlador de E/S, la búsqueda de disco y el tiempo de latencia. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. Si no se especifica OVERHEAD para un espacio de tablas de almacenamiento no automático, el valor se establecerá por omisión en el valor por omisión de creación de base de datos que aparece más adelante en la descripción de esta palabra clave. Si no se especifica OVERHEAD para un espacio de tablas de almacenamiento automático, el valor por omisión es heredar (INHERIT) el valor del grupo de almacenamiento que está utilizando. Si el valor OVERHEAD del grupo de almacenamiento no está definido, el valor por omisión de OVERHEAD será el valor por omisión de creación de base de datos.

número-de-milisegundos

El valor de *número-de-milisegundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todos los contenedores, el número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y OVERHEAD se hereda de forma dinámica del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42613).

En el caso de una base de datos que se ha creado en DB2 V10.1 o posterior, la actividad general del controlador de E/S por omisión y el tiempo de búsqueda y latencia de disco para el espacio de tablas PAGESIZE de 4 KB es de 6,725 milisegundos.

En el caso de una base de datos que se ha actualizado de una versión anterior de DB2 a DB2 V10.1 o posterior, la actividad general del controlador de E/S y el tiempo de búsqueda y latencia de disco por omisión son los siguientes:

- 7,5 milisegundos para una base de datos creada en DB2 versión 9.1 o posterior
- 12,67 milisegundos para las bases de datos creadas entre DB2 versión 8.2 y DB2 versión 9.1
- 24,1 milisegundos para las versiones DB2 anteriores a 8.2

FILE SYSTEM CACHING o NO FILE SYSTEM CACHING

Especifica si las operaciones de E/S se deben almacenar en antememoria o no a nivel del sistema de archivos. Si no se especifica ninguna opción, el valor por omisión es:

CREATE TABLESPACE

- FILE SYSTEM CACHING para JFS en AIX, Linux System z, todos los sistemas de archivos no VxFS en Solaris, HP-UX, archivos de espacio de tablas temporal SMS en todas las plataformas y todos los datos grandes y LOB
- NO FILE SYSTEM CACHING en todas las otras plataformas y tipos de sistema de archivos

FILE SYSTEM CACHING

Especifica que todas las operaciones de E/S en el espacio de tablas de destino se deben almacenar en antememoria en el nivel del sistema de archivos.

NO FILE SYSTEM CACHING

Especifica que todas las operaciones de E/S no se deben almacenar en antememoria en el nivel del sistema de archivos.

TRANSFERRATE *número-de-milisegundos* o TRANSFERRATE INHERIT

Especifica el tiempo necesario para leer una página en memoria. Si no se especifica TRANSFERRATE para un espacio de tablas de almacenamiento no automático, el valor se establecerá por omisión en el valor por omisión de creación de base de datos que aparece más adelante en la descripción de esta palabra clave. Si no se especifica TRANSFERRATE para un espacio de tablas de almacenamiento automático, el valor por omisión es heredar (INHERIT) el valor del grupo de almacenamiento que está utilizando. Si el valor DEVICE READ RATE del grupo de almacenamiento no está definido, el valor por omisión de TRANSFERRATE será el valor por omisión de creación de base de datos.

número-de-milisegundos

Este valor sirve para determinar el coste de E/S durante la optimización de una consulta. El valor de *número-de-milisegundos* es cualquier literal numérico (entero, decimal o coma flotante). Si este valor no es el mismo para todos los contenedores, el número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y TRANSFERRATE se hereda de forma dinámica de la DEVICE READ RATE del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42613).

Cuando un espacio de tablas de almacenamiento automático hereda el valor TRANSFERRATE del grupo de almacenamiento que utiliza, la DEVICE READ RATE del grupo de almacenamiento, que se expresa en megabytes por segundo, se convierte en la cuenta de milisegundos por página leída para el valor PAGESIZE del espacio de tablas. La fórmula de conversión es la siguiente:

$$\text{TRANSFERRATE} = (1 / \text{DEVICE READ RATE}) * 1000 / 1024000 * \text{PAGESIZE}$$

En el caso de una base de datos que se ha creado en DB2 V10.1 o posterior, el tiempo por omisión para leer una página en memoria para el espacio de tablas PAGESIZE de 4 KB es 0,04 milisegundos.

En el caso de una base de datos que se ha actualizado de una versión anterior de DB2 a DB2 V10.1 o posterior, el tiempo por omisión para leer una página en memoria es el siguiente:

- 0,06 milisegundos para una base de datos creada en DB2 versión 9.1 o posterior
- 0,18 milisegundos para las bases de datos creadas entre DB2 versión 8.2 o versión 9.1
- 0,9 milisegundos para las versiones de DB2 anteriores a la 8.2

DATA TAG *constante-entero*, DATA TAG INHERIT o DATA TAG NONE

Especifica un código para los datos en el espacio de tablas. Si no se especifica DATA TAG, el valor por omisión para los espacios de tablas de almacenamiento automático es heredar (INHERIT) del grupo de almacenamiento que está utilizando y, para los espacios de tablas no automáticos, se establecerá en NONE. Este valor se puede utilizar como parte de una configuración de WLM en una definición de clase de trabajo o se puede referenciar en una definición de umbral; para obtener más información, consulte las sentencias CREATE WORK CLASS SET y CREATE THRESHOLD. Esta cláusula no se puede especificar si también se especifica TEMPORARY (SQLSTATE 42613).

constante-entera

Los valores válidos para *constante-entera* son los enteros del 1 al 9. Si se especifica una *constante-entera* y existe un grupo de almacenamiento asociado, el código de datos especificado para el espacio de tablas alterará temporalmente cualquier valor de código de datos que se especifique para el grupo de almacenamiento asociado.

INHERIT

Si se especifica INHERIT, el espacio de tablas debe definirse mediante el almacenamiento automático y el código de datos se hereda de forma dinámica del grupo de almacenamiento. INHERIT no puede especificarse si el espacio de tablas no se define mediante el almacenamiento automático (SQLSTATE 42613).

NONE

Si se especifica NONE, no hay ningún código de datos.

DROPPED TABLE RECOVERY

Indica si las tablas descartadas en el espacio de tablas especificado se pueden recuperar mediante la opción **RECOVER DROPPED TABLE** del mandato **ROLLFORWARD DATABASE**. Esta cláusula sólo se puede especificar para un espacio de tablas normal o más grande (SQLSTATE 42613).

ON Especifica que las tablas descartadas pueden recuperarse. Éste ha sido el valor por omisión desde la versión 8.

OFF

Especifica que las tablas descartadas no pueden recuperarse. Éste es el valor por omisión en la versión 7.

Normas

- Si el almacenamiento automático no se define para la base de datos, se devuelve un error (SQLSTATE 55060).
- La cláusula INITIALSIZE no puede especificarse con la cláusula MANAGED BY SYSTEM o MANAGED BY DATABASE (SQLSTATE 42601).
- La cláusula AUTORESIZE, INCREASESIZE o MAXSIZE no puede especificarse con la cláusula MANAGED BY SYSTEM (SQLSTATE 42601).
- La cláusula AUTORESIZE, INITIALSIZE, INCREASESIZE o MAXSIZE no puede especificarse para la creación de un espacio de tablas de almacenamiento automático temporal (SQLSTATE 42601).

CREATE TABLESPACE

- La cláusula INCREASESIZE o MAXSIZE no puede especificarse si el espacio de tablas no permite cambiar el tamaño automáticamente (SQLSTATE 42601).
- AUTORESIZE no puede habilitarse para los espacios de tablas DMS que están definidos para utilizar contenedores de dispositivo sin procesar (SQLSTATE 42601).
- Un espacio de tablas debe ser, inicialmente, lo bastante grande como para contener cinco extensiones (SQLSTATE 57011).
- El tamaño máximo de un espacio de tablas debe ser superior a su tamaño inicial (SQLSTATE 560B0).
- Las operaciones con contenedores (ADD, EXTEND, RESIZE, DROP o BEGIN NEW STRIPE SET) no pueden realizarse en espacios de tablas de almacenamiento automático porque el gestor de bases de datos controla la gestión del espacio de dichos espacios de tablas (SQLSTATE 42858).
- Cada definición de contenedor requiere 53 bytes más el número de bytes necesario para almacenar el nombre de contenedor. La longitud combinada de todas las definiciones de contenedores para el espacio de tablas no puede superar los 208 kilobytes (SQLSTATE 54034).
- Para una base de datos particionada, si más de una partición de base de datos reside en el mismo nodo físico, no se puede especificar el mismo dispositivo ni la misma vía de acceso para más de una partición de base de datos (SQLSTATE 42730). En este caso, especifique una *serie-contenedor* para cada partición de base de datos o utilice un nombre de vía de acceso relativo.
- Sólo los espacios de tablas de almacenamiento automático pueden crearse en un entorno DB2 pureScale. (SQLSTATE 42997)
- **Límites de tamaño de contenedor:** En espacios de tablas DMS, un contenedor debe tener al menos el doble del tamaño de extensión en páginas de longitud (SQLSTATE 54039). El tamaño máximo de un contenedor depende del sistema operativo.

Notas

- La elección entre un espacio de tablas gestionado por la base de datos y un espacio de tablas gestionado por el sistema es una elección fundamental que implica concesiones.
- Si existe más de un espacio de tablas TEMPORARY en la base de datos, estos se utilizan de modo rotativo para equilibrar el uso.
- El propietario del espacio de tablas recibe el privilegio USE con WITH GRANT OPTION para el espacio de tablas durante su creación.
- Un espacio de tablas de almacenamiento automático se crea como un espacio de tablas SMS o como un espacio de tablas DMS. DMS se selecciona para espacios de tablas grandes y normales, y SMS se selecciona para espacios de tablas temporales. Tenga en cuenta que no puede contar con este comportamiento, dado que es posible que cambie en un release posterior. Cuando se elige DMS y no se especifica el tipo del espacio de tablas, el comportamiento por omisión consiste en crear un espacio de tablas grande.
- La creación de un espacio de tablas de almacenamiento automático no incluye definiciones de contenedor. El gestor de bases de datos determina automáticamente la ubicación y el tamaño, si procede, de los contenedores basándose en las vías de acceso de almacenamiento asociadas con el grupo de almacenamiento especificado o el grupo de almacenamiento por omisión. El gestor de bases de datos intentará ampliar los espacios de tablas grandes y normales cuanto sea necesario, siempre y cuando no se alcance el tamaño máximo. Esto puede implicar la ampliación de los contenedores existentes o la adición de contenedores a un nuevo conjunto de bandas. Cada vez que se activa

la base de datos, el gestor de bases de datos reconfigura automáticamente el número de contenedores y su ubicación para los espacios de tablas temporales que no se encuentren en estado anómalo.

- Un espacio de tablas de almacenamiento automático grande o normal no utilizará nuevas vías de acceso a almacenamiento (consulte la descripción de la sentencia ALTER STOGROUP) hasta que no se agote el espacio en una de las vías de acceso a almacenamiento existentes que el espacio de tablas utiliza. Los espacios de tablas de almacenamiento automático temporales sólo pueden utilizar las nuevas vías de acceso de almacenamiento una vez que la base de datos se haya desactivado y, luego, reactivado.
- **Atributos de soportes:** la tabla siguiente muestra cómo se tratan los atributos de soportes de los espacios de tablas recién creados en las bases de datos DB2 V10.1 actualizadas y recién creadas.

Tabla 25. Atributos de soportes entre distintas versiones de DB2

Atributos de soportes	Base de datos actualizada	Base de datos recién creada
Nuevos espacios de tablas de almacenamiento automático/DEVICE READ RATE de grupo de almacenamiento establecida como <i>indefinida</i>	El valor por omisión depende de la versión con la que se creó la base de datos (sin cambios)	No aplicable
Nuevos espacios de tablas de almacenamiento automático/OVERHEAD de grupo de almacenamiento establecida como <i>indefinida</i>	El valor por omisión depende de la versión con la que se creó la base de datos (sin cambios)	No aplicable
Nuevos espacios de tablas de almacenamiento automático/DEVICE READ RATE de grupo de almacenamiento establecida	Heredar de grupo de almacenamiento que tiene en cuenta PAGESIZE	Heredar de grupo de almacenamiento que tiene en cuenta PAGESIZE
Nuevos espacios de tablas de almacenamiento automático/OVERHEAD de grupo de almacenamiento establecida	Heredar de grupo de almacenamiento	Heredar de grupo de almacenamiento
Nuevos espacios de tablas de almacenamiento no automáticos	El valor por omisión depende de la versión con la que se creó la base de datos (sin cambios)	Valores por omisión de soportes de DB2 V10.1 que tienen en cuenta PAGESIZE

- **TRANSFERRATE por omisión en DB2 V10.1:** en la tabla siguiente se muestra cómo difiere el valor de TRANSFERRATE para los espacios de tablas recién creados en DB2 V10.1.

Tabla 26. TRANSFERRATE por omisión en DB2 V10.1

PAGESIZE	TRANSFERRATE
4 KB	0,04 ms por página leída
8 KB	0,08 ms por página leída
16 KB	0,16 ms por página leída
32 KB	0,32 ms por página leída

CREATE TABLESPACE

- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NODE puede especificarse en lugar de DBPARTITIONNUM
 - NODES puede especificarse en lugar de DBPARTITIONNUMS
 - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
 - LONG puede especificarse en lugar de LARGE

Ejemplos

- *Ejemplo 1:* Cree un espacio de tablas DMS grande en un sistema UNIX utilizando tres dispositivos de 10.000 páginas de 4 K cada uno. 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 12.67
TRANSFERRATE 0.18
```

- *Ejemplo 2:* Cree un espacio de tablas SMS normal en Windows mediante tres directorios en tres unidades distintas, con un tamaño de extensión de 64 páginas y un tamaño de captación previa 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 de sistema en un sistema UNIX utilizando dos archivos de 50.000 páginas cada uno y un tamaño de extensión de 256 páginas.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
MANAGED BY DATABASE
USING (FILE 'dbtmp/tempspace2.f1' 50000,
       FILE 'dbtmp/tempspace2.f2' 50000)
EXTENTSIZE 256
```

- *Ejemplo 4:* Cree un espacio de tablas DMS grande en un grupo de particiones de base de datos ODDNODEGROUP (particiones 1, 3 y 5 de base de datos) en un sistema UNIX. Utilice el dispositivo /dev/rhdisk0 para 10.000 páginas de 4 K en cada partición de base de datos. Especifique un dispositivo específico de particiones de base de datos con 40.000 páginas de 4K para cada partición de base de datos.

```
CREATE TABLESPACE PLANS
MANAGED BY DATABASE
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
ON DBPARTITIONNUM (1)
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
ON DBPARTITIONNUM (3)
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
ON DBPARTITIONNUM (5)
```

- *Ejemplo 5:* Cree un espacio de tablas de almacenamiento automático grande denominado DATATS que permita que el sistema realice todas las decisiones con respecto al tamaño y al crecimiento del espacio de tablas.

```
CREATE TABLESPACE DATATS
```

o

```
CREATE TABLESPACE DATATS
MANAGED BY AUTOMATIC STORAGE
```

- *Ejemplo 6:* cree un espacio de tablas de almacenamiento automático temporal llamado TEMPDATA.

```
CREATE TEMPORARY TABLESPACE TEMPDATA
```

O

```
CREATE TEMPORARY TABLESPACE TEMPDATA
MANAGED BY AUTOMATIC STORAGE
```

- *Ejemplo 7:* cree un espacio de tala de almacenamiento automático grande llamado USERSPACE3 con un tamaño inicial de 100 megabytes y un tamaño máximo de 1 gigabyte.

```
CREATE TABLESPACE USERSPACE3
INITIALSIZE 100 M
MAXSIZE 1 G
```

- *Ejemplo 8:* Cree un espacio de tablas de almacenamiento automático grande denominado LARGEDATA con un índice de incremento del 10 por ciento (es decir, su tamaño total aumenta en un 10 por ciento cada vez que cambia automáticamente) y un tamaño máximo de 512 megabytes. En lugar de especificar la cláusula INITIALSIZE, permita que el gestor de bases de datos determine un tamaño inicial apropiado para el espacio de tablas.

```
CREATE LARGE TABLESPACE LARGEDATA
INCREASESIZE 10 PERCENT
MAXSIZE 512 M
```

- *Ejemplo 9:* cree un espacio de tablas DMS grande llamado USERSPACE4 con dos contenedores de archivo (cada contenedor tiene 1 megabyte de tamaño), un índice de incremento de 2 megabytes y un tamaño máximo de 100 megabytes.

```
CREATE TABLESPACE USERSPACE4
MANAGED BY DATABASE USING (FILE '/db2/file1' 1 M, FILE '/db2/file2' 1 M)
AUTORESIZE YES
INCREASESIZE 2 M
MAXSIZE 100 M
```

- *Ejemplo 10:* cree espacios de tablas DMS grandes utilizando dispositivos RAW en un sistema operativo Windows.

– Para especificar unidades físicas completas, utilice el formato `\\.\unidad-fisica:`

```
CREATE TABLESPACE TS1
MANAGED BY DATABASE USING (DEVICE '\\.\PhysicalDrive5' 10000,
DEVICE '\\.\PhysicalDrive6' 10000)
```

– Para especificar particiones lógicas mediante letras de unidad:

```
CREATE TABLESPACE TS2
MANAGED BY DATABASE USING (DEVICE '\\.\G:' 10000,
DEVICE '\\.\H:' 10000)
```

– Para especificar particiones lógicas mediante identificadores exclusivos globales (GUID) del volumen, emplee el programa de utilidad `db2listvolumes` para recuperar el GUID del volumen de todas las particiones locales y, a continuación, copie el GUID de la partición lógica que desee incluir en la cláusula del contenedor de espacios de tablas:

```
CREATE TABLESPACE TS3
MANAGED BY DATABASE USING (
DEVICE '\\?\Volume{2ca6a0c1-8542-11d8-9734-00096b5322d2}\' 20000M)
```

Es preferible que utilice los GUID del volumen en lugar del formato de letra de unidad si tiene más particiones que letras de unidad disponibles en la máquina.

CREATE TABLESPACE

- Para especificar particiones lógicas mediante puntos de unión (o puntos de montaje de volúmenes), monte la partición RAW en otro volumen con formato NTFS como un punto de unión y, a continuación, especifique la vía de acceso al punto de unión en el volumen NTFS como la vía de acceso al contenedor. Por ejemplo:

```
CREATE TABLESPACE TS4  
MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,  
DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

DB2 primero consulta la partición para ver si esta incluye un sistema de archivos; en caso afirmativo, la partición no se trata como un dispositivo RAW y DB2 lleva a cabo operaciones de E/S del sistema de archivos en la partición.

CREATE THRESHOLD

La sentencia CREATE THRESHOLD define un umbral.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Sintaxis

```

▶▶ CREATE THRESHOLD nombre-umbral FOR dominio-umbral | ACTIVITIES →
▶ ENFORCEMENT | ámbito-implantación | [ ENABLE ]
| [ DISABLE ] →
▶ WHEN | predicado-umbral | | acciones-superaron-umbral | →▶
  
```

dominio-umbral:

```

|-----|
| DATABASE |-----|
| SERVICE CLASS nombre-clase-servicio |-----|
| | UNDER nombre-clase-servicio |-----|
| STATEMENT |-----|
| | TEXT texto-sentencia |-----|
| | REFERENCE ID-ejecutable |-----|
| WORKLOAD nombre-carga-trabajo |-----|
  
```

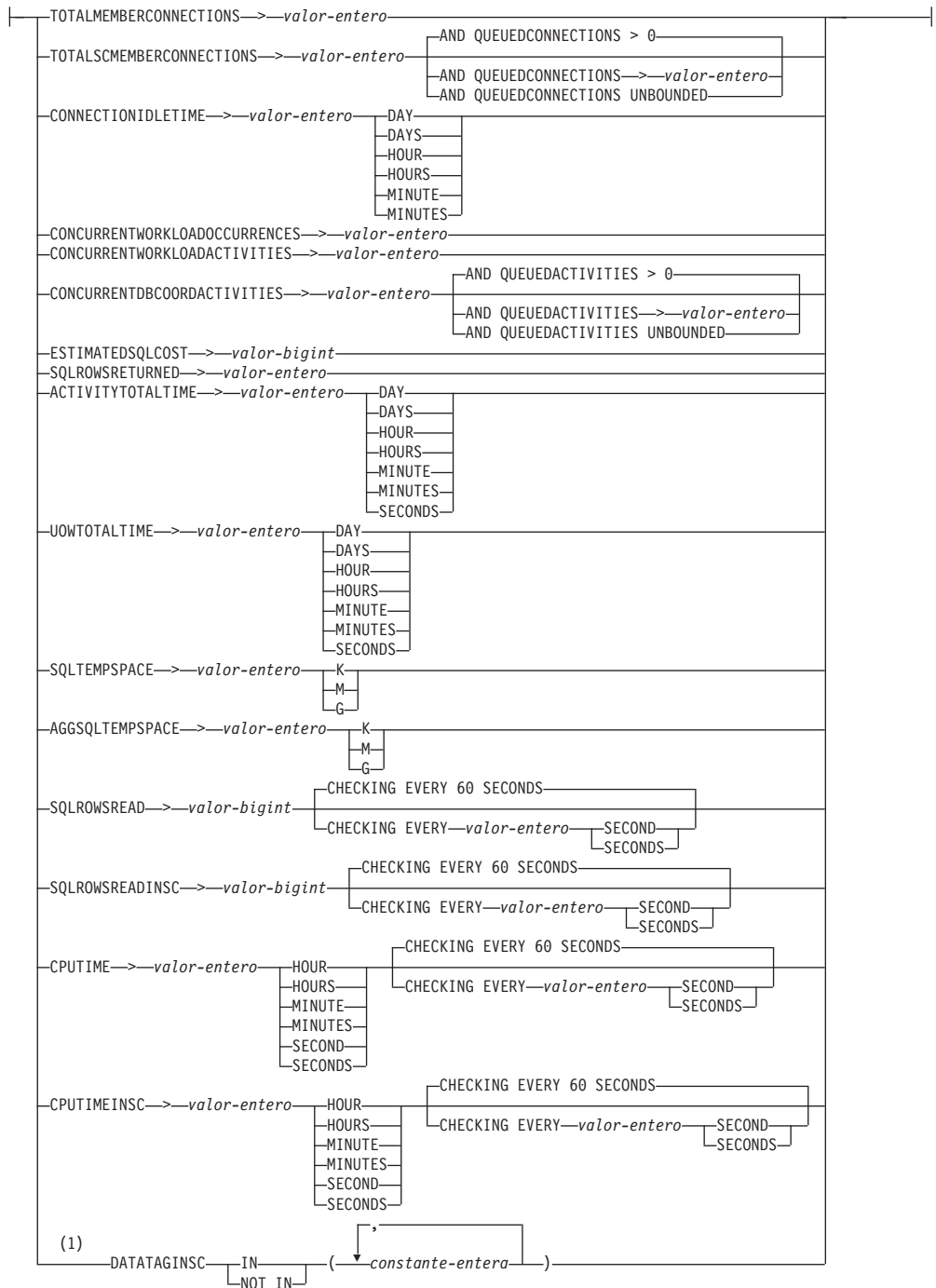
ámbito-implantación:

```

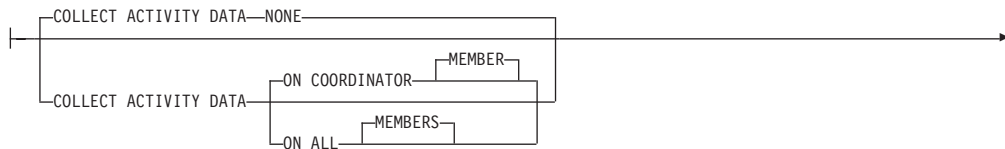
|-----|
| DATABASE |-----|
| MEMBER |-----|
| WORKLOAD OCCURRENCE |-----|
  
```

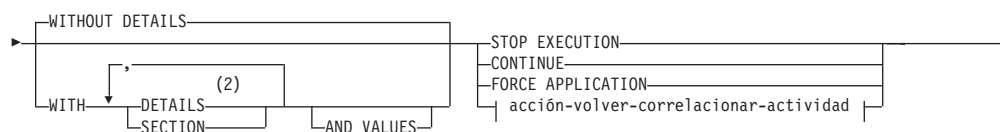
predicado-umbral:

CREATE THRESHOLD

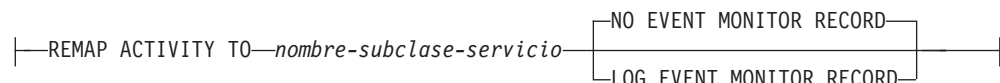


acciones-superaron-umbral:





acción-volver-correlacionar-actividad:



Notas:

- 1 Cada valor de etiqueta de datos sólo se puede especificar una vez.
- 2 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-umbral

Da nombre al umbral. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-umbral* no debe identificar una carga de trabajo que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

FOR dominio-umbral ACTIVITIES

Especifica el dominio de definición del umbral.

DATABASE

Este umbral se aplica a cualquier actividad de la base de datos.

SERVICE CLASS nombre-clase-servicio

Este umbral se aplica a las actividades que se ejecutan en la clase de servicio *nombre-clase-servicio*. Si no se especifica UNDER, *nombre-clase-servicio* debe identificar una superclase de servicio existente (SQLSTATE 42704). Si se especifica UNDER, *nombre-clase-servicio* debe identificar una subclase de servicio específica de la superclase de servicio especificada como consecuencia de la palabra clave UNDER (SQLSTATE 42704). *nombre-clase-servicio* no puede ser la clase de servicio SYSDEFAULTSYSTEMCLASS o la clase de servicio SYSDEFAULTMAINTENANCECLASS (SQLSTATE 5U032).

UNDER nombre-clase-servicio

Especifica una superclase de servicio. El *nombre-clase-servicio* debe identificar una superclase de servicio existente (SQLSTATE 42704).

STATEMENT

Este umbral se aplica a las actividades para una sentencia de SQL específica. Identifique la sentencia que se debe utilizar para el umbral especificando el texto de la sentencia o el ID ejecutable de la sentencia.

TEXT texto-sentencia

Este umbral se aplica a las sentencias que coinciden con el texto especificado en el texto especificado en *texto-sentencia*. Se tienen en cuenta las sentencias tanto de SQL estático como dinámico al evaluar la condición para el umbral. Durante la ejecución, el texto especificado para el parámetro *texto-sentencia* debe ser una coincidencia exacta del texto de una sentencia de la antememoria de paquetes para que se sobrepase el umbral. No se encuentran coincidencias entre el

CREATE THRESHOLD

parámetro *texto-sentencia* y las sentencias de SQL en ejecución si existen diferencias en las mayúsculas o minúsculas o el uso de espacios en blanco. El texto para *texto-sentencia* se debe especificar como una constante de tipo serie. Como tal, la longitud máxima para el texto de una sentencia para un umbral de sentencia es de 32 672 bytes y no el límite máximo de 2 MB, como es habitual para las sentencias.

Las diferencias de plan de acceso no afecta a la coincidencia de sentencias. Es posible que varias sentencias en antememoria que tengan el mismo texto pero planes de acceso distintos coincidan con el texto del umbral definido por el parámetro *texto-sentencia*.

Si se altera o transforma una sentencia que, de otro modo, coincide con la sentencia indicada para el parámetro *texto-sentencia* durante la compilación de manera que ya no es igual a *texto-sentencia*, las sentencias no coincidirán. Por ejemplo, si se habilita el concentrador de sentencias, los valores literales pueden sustituirse por marcadores de parámetro. No se aplica esa transformación al texto proporcionado por *texto-sentencia* que se indica en la sentencia CREATE THRESHOLD. El texto proporcionado para CREATE THRESHOLD debe coincidir exactamente con el texto transformado de la sentencia en cuestión. Puede determinar el texto exacto de las sentencias mientras se ejecutan utilizando funciones de tabla de supervisión como **MON_GET_PKG_CACHE_STMT** y **MON_GET_ACTIVITY_DETAILS**.

Los predicados siguientes se pueden utilizar con un umbral de sentencia: **ESTIMATEDSQLCOST**, **SQLROWSRETURNED**, **ACTIVITYTOTALTIME**, **SQLROWSREAD**, **CPUTIME**, **SQLTEMPSPACE**.

REFERENCE *ID-ejecutable*

Este umbral se aplica a las sentencias con texto que coincidan con el texto de la sentencia con el ID ejecutable especificado. El gestor de base de datos utiliza el ID ejecutable para localizar el texto de la sentencia en su sección de la antememoria de paquetes. El texto de la sentencia que se utiliza para el umbral es el que se ha puesto en antememoria para la sección en el momento en el que se ha creado el umbral. Para un SQL dinámico, la sentencia a la que hace referencia el ID ejecutable debe estar en la antememoria de paquetes. Para un SQL estático, si la sentencia no está en la antememoria, el gestor de base de datos la recuperará de los catálogos del sistema.

Una vez que el texto de la sentencia se recupera de la antememoria de paquetes, no existe ninguna relación directa entre el umbral y el ID ejecutable especificado; se puede desalojar de la antememoria la sección en antememoria sin que ello afecte a ningún umbral que se derive de ella. Una vez que se ha determinado el texto asociado con el ID ejecutable, el umbral creado por esta cláusula actúa exactamente del mismo modo que uno creado mediante la cláusula STATEMENT TEXT.

WORKLOAD *nombre-carga-trabajo*

Este umbral se aplica a la carga de trabajo especificada. El *nombre-carga-trabajo* debe identificar una carga de trabajo existente (SQLSTATE 42704).

ENFORCEMENT *ámbito-implantación*

El ámbito de implantación del umbral.

DATABASE

El umbral se implanta en todos los miembros dentro del dominio de definición, es decir, todos los miembros de la base de datos y todos los miembros de la clase de servicio.

MEMBER

El umbral se implanta según el miembro. No existe ninguna coordinación entre todos los miembros para implantar el umbral.

WORKLOAD OCCURRENCE

El umbral se implanta sólo en una aparición de carga de trabajo. Dos apariciones de carga de trabajo que se ejecuten simultáneamente en el mismo miembro dispondrán cada una de ellas de su propio número de ejecución para este umbral.

ENABLE o DISABLE

Especifica si el umbral está habilitado o no para que lo utilice el gestor de la base de datos.

ENABLE

El gestor de base de datos utiliza el umbral para restringir la ejecución de actividades de base de datos.

DISABLE

El gestor de base de datos no utiliza el umbral para restringir la ejecución de actividades de base de datos.

WHEN *predicado-umbral*

Especifica la condición del umbral.

TOTALMEMBERCONNECTIONS > *valor-entero*

Esta condición define un límite superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en un miembro. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá la conexión de cualquier conexión de coordinador nueva. Proseguirán todas las conexiones en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser DATABASE y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037). Este umbral no se aplica para los usuarios con autorización DBADM o WLMADM.

TOTALSCMEMBERCONNECTIONS > *valor-entero*

Esta condición define un límite superior en el número de conexiones de coordinador que pueden ejecutarse simultáneamente en un miembro de una superclase de servicio específica. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá que cualquier conexión nueva se una a la clase de servicio. Proseguirán todas las conexiones en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser SERVICE SUPERCLASS y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037).

AND QUEUEDCONNECTIONS > *valor-entero* o **AND QUEUEDCONNECTIONS UNBOUNDED**

Especifica un tamaño de cola para el momento en que se supere el número máximo de conexiones de coordinador. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna conexión de coordinador. Especificar UNBOUNDED pondrá en cola todas las

CREATE THRESHOLD

conexiones que superen el número máximo especificado de conexiones de coordinador y las *acciones-umbral-superado* nunca se ejecutarán. El valor por omisión es cero.

CONNECTIONIDLETIME > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES

Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá a una conexión permanecer desocupada. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. El dominio de definición para esta condición debe ser DATABASE o SERVICE SUPERCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037). Esta condición se aplica en el miembro coordinador.

Si se especifica la acción STOP EXECUTION con umbrales CONNECTIONIDLETIME, la conexión para la aplicación se descarta cuando se supera el umbral. Cualquier intento posterior por parte de la aplicación para acceder al servidor de datos no recibirá SQLSTATE 5U026.

El valor máximo para este umbral es 2.147.483.640 segundos. Cualquier valor que se especifique que tenga un número de segundos equivalente a 2.147.483.640 se establecerá en este número de segundos.

CONCURRENTWORKLOADOCCURRENCES > *valor-entero*

Esta condición define un límite superior en el número de apariciones simultáneas para la carga de trabajo en cada miembro. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser WORKLOAD y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037).

CONCURRENTWORKLOADACTIVITIES > *valor-entero*

Esta condición define un límite superior en el número de actividades de coordinador simultáneas y actividades anidadas para la carga de trabajo en cada miembro. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser WORKLOAD y el ámbito de implantación para esta condición debe ser WORKLOAD OCCURRENCE (SQLSTATE 5U037).

Cada actividad anidada debe satisfacer las condiciones siguientes:

- Debe ser una actividad de coordinador reconocida. No se contará ninguna actividad de coordinador anidada que no caiga dentro de los tipos de actividades reconocidos. De modo análogo, no se contará ninguna actividad de subagente anidado, como por ejemplo las peticiones de nodo remoto.
- Debe invocarse directamente desde la lógica de usuario, como por ejemplo un procedimiento escrito por el usuario que emita sentencias de SQL.

En consecuencia, las actividades coordinadoras anidadas que se han iniciado automáticamente bajo la invocación de un programa de utilidad de DB2 o rutinas de los esquemas SYSIBM, SYSFUN o SYSPROC no cuentan con relación a la vinculación superior especificada mediante este umbral.

Este umbral tampoco cuenta las actividades de SQL internas, como las que se inician mediante el establecimiento de una restricción o la actualización de una tabla de consulta materializada, pues estas las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.

CONCURRENTDBCOORDACTIVITIES > *valor-entero*

Esta condición define un límite superior sobre el número de actividades de coordinador de base de datos reconocidas que pueden ejecutarse simultáneamente en todos los miembros del dominio especificado. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que se impedirá la ejecución de las actividades de cualquier coordinador de base de datos nueva. Proseguirán todas las actividades de coordinador de base de datos que estén en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser DATABASE, una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET), SERVICE SUPERCLASS. Asimismo, el ámbito de aplicación debe ser DATABASE (SQLSTATE 5U037) en entornos distintos de DB2 pureScale, donde la condición se aplica en toda la base de datos, y MEMBER (SQLSTATE 5U037) en DB2 pureScale donde la condición se aplica en cada uno de los miembros coordinadores. Esta condición realiza el seguimiento de todas las actividades, salvo de los elementos siguientes:

- Este umbral no controla las sentencias CALL, pero todas las actividades hijo anidadas iniciadas dentro de la rutina llamada se someten al control de este umbral. Los bloques anónimos y las rutinas autónomas se clasifican como sentencias CALL.
- Este umbral controla las funciones definidas por el usuario, pero las actividades hijo anidadas en una función definida por el usuario no se controlan. Si se llama a una rutina autónoma desde dentro de una función definida, ni la rutina autónoma ni las actividades hijo de la rutina autónoma están bajo el control del umbral.
- Este umbral no controla las acciones de activador que invocan sentencias CALL y las actividades hijo de esas sentencias CALL. Las sentencias INSERT, UPDATE o DELETE que pueden hacer que un activador se active siguen estando sometidas al control del umbral.

Cuando se define un umbral como parte de un conjunto de acciones de trabajo, el ámbito de aplicación se determina de forma automática en función del entorno actual (MEMBER, si el entorno actual es DB2 pureScale; DATABASE, si es otro entorno).

Importante: Antes de utilizar umbrales CONCURRENTDBCOORDACTIVITIES, asegúrese de haberse familiarizado con los efectos que pueden tener en el sistema de bases de datos. Para obtener más información, consulte el tema "Umbral CONCURRENTDBCOORDACTIVITIES".

AND QUEUEDACTIVITIES > *valor-entero* o **AND QUEUEDACTIVITIES UNBOUNDED**

Especifica un tamaño de cola para el momento en que se supere el número máximo de actividades de coordinador de base de datos. Este valor puede ser cero o cualquier entero positivo (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna actividad de coordinador de base de datos. Especificar UNBOUNDED pondrá en cola todas las actividades de coordinador de base de datos que superen el número máximo especificado de actividades de coordinador de base de datos y las *acciones-umbral-superado* nunca se ejecutarán. El valor por omisión es cero.

CREATE THRESHOLD

Nota: Si se especifica la acción de umbral CONTINUE para un umbral de cola, el tamaño de la cola no queda acotado, independientemente de los valores fijos que se hayan especificado para el tamaño de la cola.

ESTIMATEDSQLCOST > *valor-bigint*

Esta condición define una vinculación superior para el coste asignado de optimizador (en activaciones de temporizador) de una actividad. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser DATABASE, una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o a una base de datos), SERVICE SUPERCLASS, SERVICE SUBCLASS o WORKLOAD, y el ámbito de aplicación debe ser DATABASE (SQLSTATE 5U037). Esta condición se aplica en el miembro coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.
- Las actividades de DML anidado que se invocan desde la lógica del usuario. En consecuencia, esta condición no hace el seguimiento de las actividades de DML (a menos que su coste se incluya en la estimación del padre, en cuyo caso se hace un seguimiento indirecto de las mismas) que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

SQLROWSRETURNED > *valor-entero*

Esta condición define una vinculación superior para el número de filas devueltas a una aplicación cliente desde el servidor de la aplicación. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser DATABASE, una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o a una base de datos), SERVICE SUPERCLASS, SERVICE SUBCLASS o WORKLOAD, y el ámbito de aplicación debe ser DATABASE (SQLSTATE 5U037). Esta condición se aplica en el miembro coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML.
- Actividades de DML anidado que derivan de la lógica del usuario. Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

Los conjuntos de resultados devueltos desde un procedimiento se tratan como actividades individuales por separado. Las filas que devuelve el propio procedimiento no se agregan.

ACTIVITYTOTALTIME > *integer-value* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECONDS

Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá que se ejecute una actividad, incluyendo el tiempo que la actividad estuvo en cola. Las actividades que este umbral cubre incluyen la ejecución de sentencias de SQL, sin incluir el tiempo de compilación y el programa de utilidad de carga. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. El dominio de definición para esta condición debe ser DATABASE, una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o a una base de datos), SERVICE SUPERCLASS, SERVICE SUBCLASS o WORKLOAD, y el ámbito de aplicación debe ser DATABASE (SQLSTATE 5U037). Esta condición se aplica en el miembro coordinador.

Si la unidad de tiempo especificada es SECONDS, el valor debe ser un múltiplo de 10 (SQLSTATE 42615). El valor máximo que se puede especificar para este umbral es 2.147.483.640 segundos. Cualquier valor especificado (mediante la utilización de la unidad de tiempo DAY, HOUR, MINUTE o SECONDS) que tenga un equivalente en segundos mayor que 2.147.483.640 segundos se truncará en este número de segundos.

UOWTOTALTIME > *valor-entero* DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES | SECONDS

Esta condición define un límite superior para la cantidad de tiempo que el gestor de bases de datos concederá a la ejecución de una unidad de trabajo. Este valor puede ser cualquier entero positivo distinto de cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. Si la unidad de tiempo especificada es SECONDS, el valor debe ser un múltiplo de 10 (SQLSTATE 42615). El dominio de definición para esta condición debe ser DATABASE, SERVICE SUPERCLASS o WORKLOAD, y el ámbito de aplicación debe ser DATABASE (SQLSTATE 5U037). Esta condición se aplica en el miembro coordinador.

El valor máximo que se puede especificar para este umbral es 2.147.483.640 segundos. Si cualquier valor especificado (mediante la unidad de tiempo DAY, HOUR o MINUTE) tiene un equivalente en segundos superior al valor máximo, se devuelve un error (SQLSTATE 42615).

SQLTEMPSPACE > *valor-entero* K | M | G

Esta condición define la cantidad máxima de espacio temporal del sistema que una sentencia de SQL puede consumir en un miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Si se especifica *valor-entero* K (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero* M, el tamaño máximo es *valor entero* multiplicado por 1 048 576. Si se especifica *valor-entero* G, el tamaño máximo es *valor entero* multiplicado por 1 073 741 824.

El dominio de definición para esta condición debe ser DATABASE, una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o a una base de datos), SERVICE SUPERCLASS, SERVICE SUBCLASS o WORKLOAD, y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037). Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (ejecución de subsección). Las

CREATE THRESHOLD

actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

AGGSQLTEMPSPACE > *valor-entero* K | M | G

Esta condición define la cantidad máxima de espacio temporal del sistema que puede consumir un conjunto de sentencias en una clase de servicio en un miembro. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

Si se especifica *valor-entero* K (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero* M, el tamaño máximo es *valor entero* multiplicado por 1.048.576. Si se especifica *valor-entero* G, el tamaño máximo es *valor entero* multiplicado por 1.073.741.824.

El dominio de definición para esta condición debe ser SERVICE SUBCLASS y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037).

Las actividades que contribuyen al agregado cuyo seguimiento efectúa esta condición son:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente, como la ejecución de subsección.
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente, como la ejecución de subsección. Las actividades iniciadas por medio del gestor de bases de datos mediante un programa de utilidad, procedimiento o sentencia de SQL interna no resultan afectadas por esta condición.

SQLROWSREAD > *valor-bigint*

Esta condición define un límite superior en el número de filas que puede leer una actividad durante su vida útil en un miembro concreto. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). Tenga en cuenta que el número de filas leídas es distinto del número de filas devueltas, que se controla mediante la condición SQLROWSRETURNED.

El dominio de definición para esta condición debe ser DATABASE, SERVICE CLASS, una subclase de servicio (SERVICE CLASS especifica la cláusula UNDER), WORKLOAD o una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o una base de datos), y el ámbito de implementación debe ser MEMBER (SQLSTATE 5U037). Esta condición se implanta de forma independiente en cada miembro.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas como las iniciadas estableciendo una restricción o renovando una tabla

de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.

CHECKING EVERY *valor-entero* **SECOND | SECONDS**

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. El umbral se comprueba al final de cada petición (como una operación de captación, por ejemplo) y en el intervalo definido por la cláusula CHECKING. La cláusula CHECKING define una vinculación superior sobre el tiempo durante el que una infracción de umbral puede estar sin detectarse. El valor por omisión es 60 segundos. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

SQLROWSREADINSC *>valor-bigint*

Esta condición define un límite superior en el número de filas que puede leer una actividad en un miembro concreto mientras se ejecuta en una subclase de servicio. Las filas leídas antes de ejecutarse en la subclase de servicio especificada no cuentan. Este valor puede ser cualquier entero grande positivo distinto de cero (SQLSTATE 42820). Tenga en cuenta que el número de filas leídas es distinto del número de filas devueltas, que se controla mediante la condición SQLROWSRETURNED.

El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037). Esta condición se implanta de forma independiente en cada miembro.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.

CHECKING EVERY *valor-entero* **SECOND | SECONDS**

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. El umbral se comprueba al final de cada petición (como una operación de captación, por ejemplo) y en el intervalo definido por la cláusula CHECKING. La cláusula CHECKING define una vinculación superior sobre el tiempo durante el que una infracción de umbral puede estar sin detectarse. El valor por omisión es 60 segundos. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Establecer un valor bajo puede afectar negativamente al rendimiento del sistema.

CPUTIME *> valor-entero* **HOURL | HOURS | MINUTE | MINUTES | SECOND | SECONDS**

Esta condición define un límite superior para la cantidad de tiempo de procesador que una actividad puede consumir durante su vida útil en un

CREATE THRESHOLD

miembro. El tiempo de procesador cuyo seguimiento realiza este umbral se mide desde el momento en que la actividad empieza a ejecutarse. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

El dominio de definición para esta condición debe ser DATABASE, una superclase de servicio (SERVICE CLASS), una subclase de servicio (SERVICE CLASS especifica la cláusula UNDER), WORKLOAD o una acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea mediante una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET, y el conjunto de acciones de trabajo debe aplicarse a una carga de trabajo o una base de datos), y el ámbito de implementación debe ser MEMBER (SQLSTATE 5U037). Esta condición se implanta de forma independiente en cada miembro.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas, como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.
- Actividades de tipo CALL. En el caso de actividades CALL, el tiempo de procesador cuyo seguimiento se haya efectuado para el procedimiento no incluye el tiempo de procesador utilizado por cualquier actividad hija o por cualquier proceso de modalidad no delimitada. La condición de umbral sólo se comprobará cuando la devuelva la lógica de usuario al motor de la base de datos. Por ejemplo: durante la ejecución de una rutina fiable, la condición de umbral sólo se comprobará cuando la rutina emita una petición al motor de la base de datos.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. La granularidad del umbral CPUTIME es aproximadamente este número multiplicado por el grado de paralelismo de la actividad. Por ejemplo: si el umbral se comprueba cada 60 segundos y el grado de paralelismo es 2, la actividad puede utilizar 2 minutos más de tiempo de procesador, en lugar de 1 minuto antes de que se detecte la infracción de umbral. El valor por omisión es 60 segundos. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

CPUTIMEINSC > *valor-entero* HOUR | HOURS | MINUTE | MINUTES | SECOND | SECONDS

Esta condición define un límite superior para la cantidad de tiempo de procesador que una actividad puede consumir en un miembro concreto mientras se está ejecutando en una subclase de servicio particular. El tiempo de procesador cuyo seguimiento realiza este umbral se mide desde el momento en que la actividad empieza a ejecutarse en la subclase de servicio identificada en el dominio de umbral. El tiempo de procesador

utilizado con anterioridad a ese punto no cuenta con relación al límite que este umbral impone. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42820).

El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser MEMBER (SQLSTATE 5U037). Esta condición se implanta de forma independiente en cada miembro.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (como la ejecución de subsección).
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (como la ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad o procedimiento (salvo el procedimiento ADMIN_CMD) no resultan afectadas por esta condición.
- Este umbral tampoco efectúa el seguimiento de actividades SQL internas, como las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que las inicia el gestor de bases de datos y no las invoca directamente la lógica de usuario.
- Actividades de tipo CALL. En el caso de actividades CALL, el tiempo de procesador cuyo seguimiento se haya efectuado para el procedimiento no incluye el tiempo de procesador utilizado por cualquier actividad hija o por cualquier proceso de modalidad no delimitada. La condición de umbral sólo se comprobará cuando la devuelva la lógica de usuario al motor de la base de datos. Por ejemplo: durante la ejecución de una rutina fiable, la condición de umbral sólo se comprobará cuando la rutina emita una petición al motor de la base de datos.

CHECKING EVERY *valor-entero* SECOND | SECONDS

Especifica la frecuencia con la que se comprueba una actividad en la condición de umbral. La granularidad del umbral CPUTIMEINSC es aproximadamente este número multiplicado por el grado de paralelismo de la actividad. Por ejemplo: si el umbral se comprueba cada 60 segundos y el grado de paralelismo es 2, la actividad puede utilizar 2 minutos más de tiempo de procesador, en lugar de 1 minuto antes de que se detecte la infracción de umbral. El valor por omisión es 60 segundos. El valor puede ser cualquier entero positivo diferente de cero con un valor máximo de 86400 segundos (SQLSTATE 42820). Si se establece un valor bajo, el rendimiento del sistema puede verse afectado negativamente.

DATATAGINSC IN (*constante-entero*, ...)

Esta condición define uno o más valores de etiqueta de datos especificados en un espacio de tablas que la actividad toca. La etiqueta de datos de un espacio de tablas, o su grupo de almacenamiento subyacente (donde proceda) puede no establecerse o establecerse en un valor del 1 al 9. Si la actividad toca un espacio de tablas que no tiene ninguna etiqueta de datos establecida (en el nivel de espacio de tablas o de grupo de almacenamiento), este umbral no tendrá ningún efecto en esa actividad. El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037). Esta condición se implanta independientemente en cada partición de base de datos.

CREATE THRESHOLD

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.
- Las actividades de DML anidado que se invocan directamente desde la lógica del usuario.

Esta condición no hace el seguimiento de las actividades de DML que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

Este umbral sólo se comprueba cuando se abre una exploración en una tabla o cuando se realiza una inserción en una tabla. La recuperación de datos de una tabla después de que se haya abierto una exploración no superará el umbral.

DATATAGINSC NOT IN (*constante-entero, ...*)

Esta condición define uno o más valores de etiqueta de datos no especificados en un espacio de tablas que la actividad toca. La etiqueta de datos de un espacio de tablas, o su grupo de almacenamiento subyacente (donde proceda) puede no establecerse o establecerse en un valor del 1 al 9. Si la actividad toca un espacio de tablas que no tiene ninguna etiqueta de datos establecida (en el nivel de espacio de tablas o de grupo de almacenamiento), este umbral no tendrá ningún efecto en esa actividad. El dominio de definición para esta condición debe ser una subclase de servicio (SERVICE CLASS con la especificación de la cláusula UNDER) y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037). Esta condición se implanta independientemente en cada partición de base de datos.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo.
- Las actividades de DML anidado que se invocan directamente desde la lógica del usuario.

Esta condición no hace el seguimiento de las actividades de DML que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

Este umbral sólo se comprueba cuando se abre una exploración en una tabla o cuando se realiza una inserción en una tabla. La recuperación de datos de una tabla después de que se haya abierto una exploración no superará el umbral.

acciones-superaron-umbral

Especifica la acción que va a adoptarse cuando se supera una condición. Cada vez que se supera una condición, se registra un suceso en el supervisor de sucesos de violaciones, si hay alguno activo.

COLLECT ACTIVITY DATA

Especifica que los datos relacionados con cada actividad que excedan el umbral han de enviarse a cualquier supervisor de sucesos de actividades activo cuando se haya completado la actividad. El valor por omisión es COLLECT ACTIVITY DATA NONE. Si se especifica COLLECT ACTIVITY DATA, el valor por omisión es WITHOUT DETAILS. El valor COLLECT ACTIVITY DATA no es aplicable a umbrales que no sean de actividad, como los siguientes: CONNECTIONIDLETIME, TOTALDBPARTITIONCONNECTIONS,

TOTALSCPARTITIONCONNECTIONS,
CONCURRENTWORKLOADOCCURRENCES, UOWTOTALTIME.

NONE

Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que supera el umbral.

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Para umbrales de predicción, la información sobre actividades sólo se recopila en todos los miembros si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL MEMBERS no tiene efecto y la información sobre actividades siempre se recopila únicamente en el miembro coordinador. Para umbrales de predicción y de reacción, los detalles de actividades, la información de sección o los valores de actividades únicamente se recopilarán en el miembro coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida, deben enviarse a cualquier supervisor de sucesos de actividades activas cuando se completa la actividad. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH**DETAILS**

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. En el caso de umbrales predictivos, los datos reales de sección se recopilarán en cualquier miembro donde se recopilen los datos de actividad. En el caso de umbrales reactivos, los datos reales de sección se recopilarán sólo en el miembro coordinador.

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

STOP EXECUTION

La ejecución de la actividad se detiene y se devuelve un error (SQLSTATE 5U026). En el caso del umbral UOWTOTALTIME, la unidad de trabajo se retrotrae.

CREATE THRESHOLD

CONTINUE

La ejecución de la actividad no se detiene.

FORCE APPLICATION

Se fuerza que la aplicación salga del sistema (SQLSTATE 55032). Esta acción sólo puede especificarse para el umbral UOWTOTALTIME.

acción-volver-correlacionar-actividad

REMAP ACTIVITY TO *nombre-subclase-servicio*

La actividad se correlaciona con *nombre-subclase-servicio*. La ejecución de la actividad no se detiene. Esta acción sólo es válida para umbrales de clase en servicio como los umbrales CPUTIMEINSC, SQLROWSREADINSC, DATATAGINSC IN y DATATAGINSC NOT IN (SQLSTATE 5U037). El nombre-subclase-servicio debe identificar una subclase de servicio existente en la misma superclase que tenga asociada el umbral (SQLSTATE 5U037). El nombre-subclase-servicio no puede ser igual que la subclase de servicio asociada del umbral (SQLSTATE 5U037).

NO EVENT MONITOR RECORD

Especifica que no se grabará ningún registro de infracción de umbral.

LOG EVENT MONITOR RECORD

Especifica que si existe un supervisor de sucesos THRESHOLD VIOLATIONS y está activo, se grabará en él un registro de infracción de umbral.

Notas

- Los umbrales pueden definirse basándose en distintos aspectos del comportamiento de la base de datos para supervisar y controlar ese comportamiento. Cuando se define un umbral basado en actividades, a menos que se especifique algo distinto, éste sólo se aplicará durante la ejecución real de las sentencias de SQL, sin incluir el tiempo de compilación y el programa de utilidad de carga.
- El umbral CONCURRENTWORKLOADOCCURRENCES y el umbral CONCURRENTWORKLOADACTIVITIES son distintos en cuanto al ámbito. CONCURRENTWORKLOADOCCURRENCES controla cuántas conexiones pueden correlacionarse con una definición de carga de trabajo simultáneamente, y CONCURRENTWORKLOADACTIVITIES controla cuántas actividades puede someter simultáneamente cada correlación que se correlaciona con la definición de carga de trabajo.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- *Acción de umbral superado de CONTINUE y datos de supervisor de sucesos:* Los datos de supervisor de sucesos sólo se recopilan una vez por cada miembro cuando se ha superado una condición de umbral. Si la acción de umbral superado es CONTINUE, la actividad sigue ejecutándose y no se recopilan más datos de supervisor de sucesos para este umbral en el miembro afectado. Por ejemplo, tome en consideración un umbral de tiempo de 10 minutos con una

acción de CONTINUE. Una vez que una actividad haya superado el límite superior de 10 minutos, se recopilarán datos de supervisor de sucesos para el umbral en el miembro afectado.

- **Inmovilización de una clase de servicio:** La condición de umbral de TOTALSCPARTITIONCONNECTIONS puede utilizarse para simular clases de servicio de inmovilizadas que normalmente no pueden inmovilizarse (por ejemplo, la clase de usuario por omisión o la clase de sistema por omisión). Esto resulta útil, ya que los umbrales no se aplican a los usuarios con autorización DBADM que se ejecuten en SYSDEFAULTADMWORKLOAD, mientras que una clase de servicio inmovilizado no está disponible para todo el mundo. En consecuencia, las clases de servicio por omisión no pueden inmovilizarse directamente sino a través de un umbral que permita a los usuarios con autorización DBADM unirlos cuando se conecten a la base de datos utilizando SYSDEFAULTADMWORKLOAD.
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse TOTALDBPARTITIONCONNECTIONS en lugar de TOTALMEMBERCONNECTIONS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse TOTALSCPARTITIONCONNECTIONS en lugar de TOTALSCMEMBERCONNECTIONS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.

Ejemplos

- *Ejemplo 1:* Cree un umbral que implante una utilización de espacio de tablas temporal máximo de 50M (por partición de la base de datos) en cualquier actividad de la base de datos. Ha de detenerse cualquier actividad que viole este umbral.

```
CREATE THRESHOLD DBMAX50MEGTEMPSPACE
FOR DATABASE ACTIVITIES
ENFORCEMENT MEMBER
WHEN SQLTEMPSPACE > 50 M
STOP EXECUTION
```

- *Ejemplo 2:* Cree un segundo umbral para limitar el tiempo de ejecución por omisión de cualquier actividad de la base de datos a un máximo de 1 hora. Ha de detenerse cualquier actividad que viole este umbral.

```
CREATE THRESHOLD DBMAX1HOURLRUNTIME
FOR DATABASE ACTIVITIES
ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 1 HOUR
STOP EXECUTION
```

- *Ejemplo 3:* Suponga que se ha creado una superclase de servicio denominada BIGQUERIES para dar acomodo a las consultas utilizando más espacio temporal que el medio y en ejecución por más de 1 hora. Los umbrales definidos en esta clase de servicio anularán temporalmente los valores establecidos en el ejemplo anterior por encima del nivel de la base de datos. Tenga en cuenta el modo en

CREATE THRESHOLD

que se permite que siga la ejecución de las actividades que violen los umbrales de esta superclase, pero se recopila información detallada para un posterior análisis.

```
CREATE THRESHOLD BIGQUERIESMAX500MEGTEMPSPACE
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE MEMBER
WHEN SQLTEMPSPACE > 500 M
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
CONTINUE
```

```
CREATE THRESHOLD BIGQUERIESLONGRUNNINGTIME
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 10 HOURS
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
CONTINUE
```

- *Ejemplo 4:* Suponiendo que exista una carga de trabajo llamada PAYROLL, cree un umbral que implante el número máximo de actividades en la carga de trabajo en un número igual o inferior a 10.

```
CREATE THRESHOLD MAXACTIVITIESINPAYROLL
FOR WORKLOAD PAYROLL ACTIVITIES
ENFORCEMENT WORKLOAD OCCURRENCE
WHEN CONCURRENTWORKLOADACTIVITIES > 10
STOP EXECUTION
```

- *Ejemplo 5:* Cree un umbral que implante un número máximo de 2 actividades simultáneas en la clase de servicio BIGQUERIES.

```
CREATE THRESHOLD
MAXBIGQUERIESCONCURRENCY
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE
WHEN CONCURRENTDBCOORDACTIVITIES > 2
STOP EXECUTION
```

- *Ejemplo 6:* Cree un umbral que capture información de las actividades para una sentencia específica que se ejecute durante más de un minuto, pero que no detenga la ejecución de la sentencia.

```
CREATE THRESHOLD TH1
FOR STATEMENT
TEXT 'SELECT DISTINCT PARTS_BIN FROM STOCK WHERE PART_NUMBER = ?'
ACTIVITIES ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 1 MINUTE
COLLECT ACTIVITY DATA WITH DETAILS, SECTION AND VALUES
CONTINUE
```

CREATE TRANSFORM

La sentencia CREATE TRANSFORM define funciones de transformación, identificados por un nombre de grupo, que se utilizan para intercambiar valores de tipo estructurado con programas en lenguaje del sistema principal y con funciones externas.

Invocación

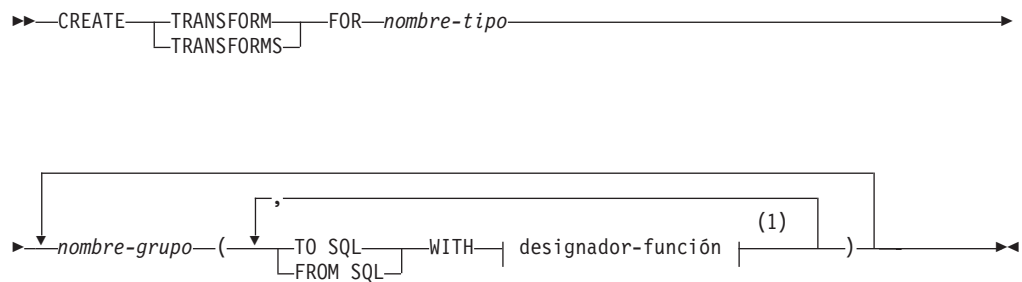
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

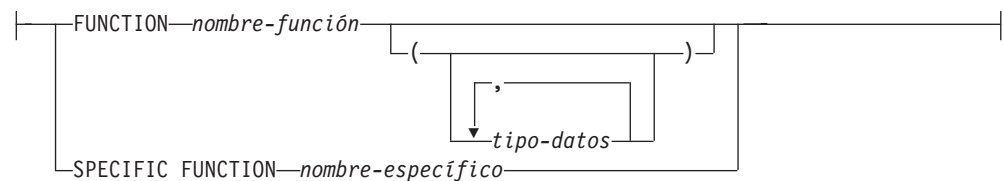
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Propietario del tipo identificado por *nombre-tipo* y privilegio EXECUTE en cada función especificada
- Autorización DBADM

Sintaxis



designador-función:



Notas:

- 1 Una misma cláusula no se debe especificar más de una vez.

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.

CREATE TRANSFORM

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un *nombre-tipo* no calificado. En las sentencias de SQL estático, 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

Da nombre al grupo de transformación. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo* no debe identificar un grupo de transformaciones que ya exista en el catálogo para el *nombre-tipo* especificado (SQLSTATE 42739). El *nombre-grupo* no debe empezar por los caracteres 'SYS' (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 *designador-función*

Identifica la función de transformación de forma exclusiva.

Si se especifica FROM SQL, *designador-función* debe identificar una función que satisface los requisitos siguientes:

- Existe un parámetro del tipo *nombre-tipo*.
- El tipo de retorno es un tipo incorporado o una fila donde todas sus columnas tienen tipos incorporados.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación FROM SQL que tiene LANGUAGE SQL.

Si se especifica TO SQL, *designador-función* debe identificar una función que satisface los requisitos siguientes:

- Todos los parámetros tienen tipos incorporados.
- El tipo de retorno es *nombre-tipo*.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación TO SQL que tiene LANGUAGE SQL.

Si *designador-función* identifica una función que no cumple estos requisitos (de acuerdo con su utilización como función de transformación FROM SQL o TO SQL), se genera un error (SQLSTATE 428DC).

No se pueden especificar métodos (incluso si se especifica con FUNCTION ACCESS) como transformaciones mediante *designador-función*. En lugar de ello, sólo las funciones definidas por la sentencia CREATE FUNCTION pueden actuar como transformaciones (SQLSTATE 42704 o 42883).

Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

Normas

- El tipo o los tipos incorporados que se devuelven de la función FROM SQL deben corresponder directamente al tipo o 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.

Notas

- Cuando no se especifica un grupo de transformación en un programa de aplicación (utilizando la opción de precompilación o de enlace TRANSFORM GROUP para el SQL estático o la sentencia SET CURRENT DEFAULT TRANSFORM GROUP para el SQL dinámico), las funciones de transformación del grupo de transformación 'DB2_PROGRAM' se utilizan (si se han definido) cuando el programa de aplicación está recuperando o enviando variables del lenguaje principal que están basadas en el tipo estructurado definido por el usuario identificado por *nombre-tipo*. Al recuperar un valor del tipo de datos *nombre-tipo*, se invoca la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado devuelto por la función de transformación. De forma similar, al enviar una variable del lenguaje principal que se asignará a un valor del tipo de datos *nombre-tipo*, se invocará la transformación TO SQL para transformar el valor del tipo de datos incorporado en el valor de tipo de datos estructurado. Si no se especifica un grupo de transformación o un grupo DB2_PROGRAM no está definido para el tipo estructurado indicado), se emite un error (SQLSTATE 42741).
- La representación del tipo de datos incorporado para una variable del lenguaje principal de tipo estructurado debe asignarse:
 - 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 normas 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 normas de asignación de memoria).

Si la asignación de una variable del lenguaje principal no es compatible con el tipo necesario para la función de transformación aplicable, se genera un error (para el enlace de entrada: SQLSTATE 42821; para el enlace de salida: SQLSTATE 42806). Para obtener información acerca de los errores que resultan de las asignaciones de series de caracteres, consulte “Asignaciones de series de caracteres”.

- Las funciones de transformación identificadas en el grupo de transformación por omisión denominado 'DB2_FUNCTION' se utilizan siempre que una función definida por el usuario no escrita en SQL se invoca utilizando el tipo de datos *nombre-tipo* como parámetro o tipo de retorno. Esto se aplica cuando la función no especifica la cláusula TRANSFORM GROUP. Cuando se invoca la función con un argumento del tipo de datos *nombre-tipo*, se ejecuta la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado devuelto por la función de transformación. De forma similar, cuando el tipo de datos de retorno de la función es un tipo de datos *nombre-tipo*, se invoca la transformación TO SQL para transformar el valor de tipo de datos incorporado devuelto del programa de función externa en el valor de tipo estructurado.

CREATE TRANSFORM

- Si un tipo estructurado contiene un atributo que también es un tipo estructurado, las funciones de transformación asociados deben expandir (o ensamblar) de forma recursiva todos los tipos estructurados anidados. Esto significa que los resultados o los parámetros de las funciones de transformación sólo constan del conjunto de tipos incorporados que representan todos los atributos base del tipo estructurado del sujeto (incluidos todos sus tipos estructurados anidados). No existe ninguna "aplicación en cascada" de las funciones de transformación para manejar los tipos estructurados anidados.
- Las funciones identificadas en esta sentencia se resuelven de acuerdo con las normas descritas anteriormente durante la ejecución de esta sentencia. Cuando estas funciones se utilizan (implícitamente) en sentencias de SQL subsiguientes, no son objeto de otro proceso de resolución. Las funciones de transformación definidos en esta sentencia se registran exactamente como se han resuelto en esta sentencia.
- Cuando se crean o descartan 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 transformaciones FROM SQL y TO SQL pueden especificarse en la misma cláusula de *nombre-grupo*, en cláusulas de *nombre-grupo* distintas o en sentencias CREATE TRANSFORM distintas. La única restricción es que una designación de transformación FROM SQL o TO SQL determinada no puede volver a definirse sin descartar primero la definición de grupo existente. Esto le permite, por ejemplo, definir primero una transformación FROM SQL para un grupo determinado y definir posteriormente la transformación TO SQL correspondiente para el mismo grupo.

Ejemplo

Crear dos grupos de transformación que asocien el tipo de estructura definida por el usuario polígono con funciones de transformación personalizadas para C y Java, respectivamente.

```
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

La sentencia CREATE TRIGGER define un activador en una base de datos.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio ALTER para la tabla en la que se define el activador BEFORE o AFTER
- Privilegio CONTROL para la vista en la que se define el activador INSTEAD OF
- Propietario de la vista en la que se define el activador INSTEAD OF
- Privilegio ALTERIN para el esquema de la tabla o de la vista en la que se define el activador
- Autorización DBADM

y uno de estos:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del activador no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema del activador hace referencia a un esquema existente
- Autorización DBADM

Si el ID de autorización de la sentencia no tiene autorización DATAACCESS, los privilegios (excluidos los privilegios de grupo) del ID de autorización de la sentencia deben incluir todas las autorizaciones siguientes, siempre y cuando exista el activador:

- En la tabla donde está definido el activador, si se especifica cualquier tabla o variable de transición:
 - Privilegio SELECT para la tabla donde está definido el activador, si se especifica cualquier tabla o variable de transición.
 - Privilegio CONTROL para la tabla donde está definido el activador, si se especifica cualquier tabla o variable de transición.
 - Autorización DATAACCESS
- En cualquier tabla o vista referenciada en la condición de la acción activada:
 - Privilegio SELECT para cualquier tabla o vista referenciada en la condición de la acción activada.
 - Privilegio CONTROL para cualquier tabla o vista referenciada en la condición de la acción activada.
 - Autorización DATAACCESS
- Los privilegios necesarios para invocar las sentencias de SQL activadas que se han especificado.

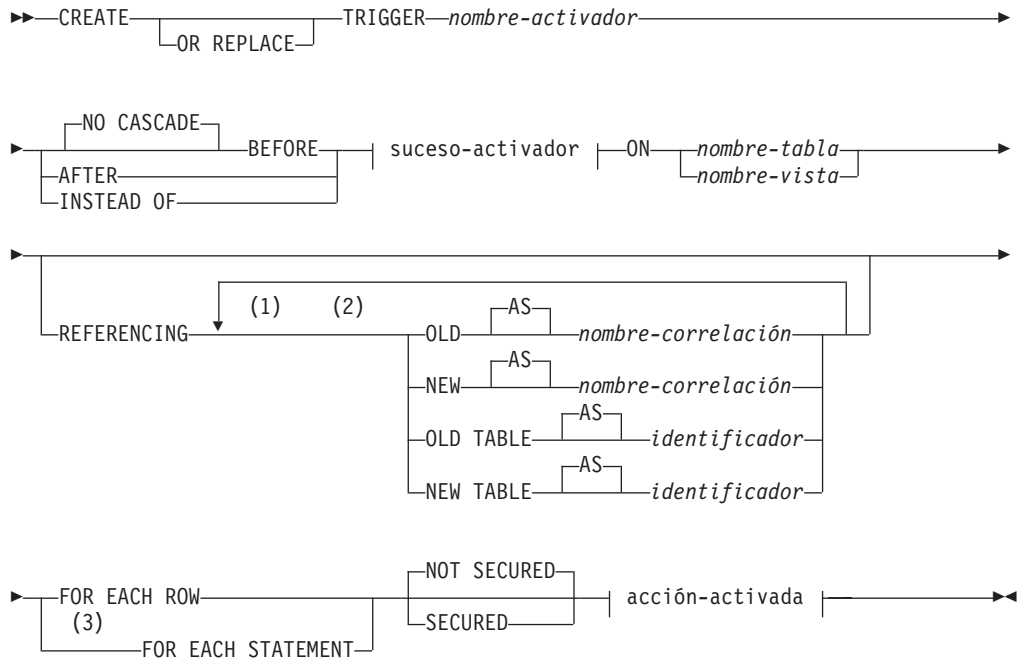
Los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE TRIGGER no se tienen en cuenta.

CREATE TRIGGER

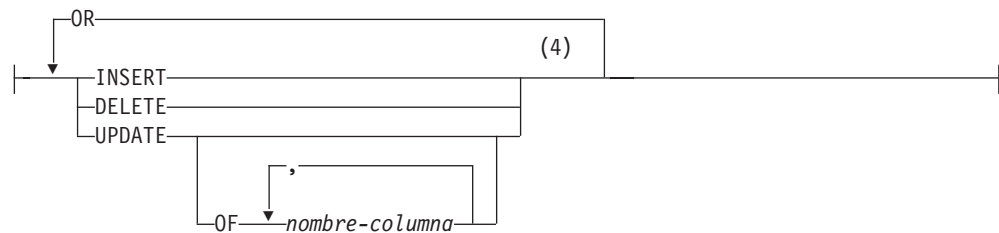
Para sustituir un activador existente, el ID de autorización de la sentencia debe ser el propietario del activador existente (SQLSTATE 42501).

Si se especifica la opción SECURED, los privilegios del ID de autorización de la sentencia deben incluir además la autorización SECADM o CREATE_SECURE_OBJECT (SQLSTATE 42501).

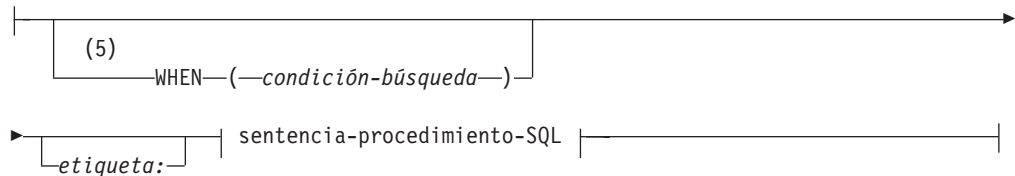
Sintaxis



suceso-activador:



acción-activada:



sentencia-procedimiento-SQL:

CALL	
	(6)
SQL compuesto (compilado)	
SQL compuesto (en línea)	
FOR	
	selección completa
WITH	expresión-tabla-común
GET DIAGNOSTICS	
IF	
INSERT	
ITERATE	
LEAVE	
MERGE	
delete-buscada	
update-buscada	
SET Variable	
SIGNAL	
WHILE	

Notas:

- 1 OLD y NEW sólo se pueden especificar una vez cada uno.
- 2 OLD TABLE y NEW TABLE sólo se pueden especificar una vez cada uno, y sólo para activadores AFTER o INSTEAD OF.
- 3 FOR EACH STATEMENT no puede especificarse para activadores BEFORE o para activadores INSTEAD OF.
- 4 No se debe especificar un suceso de activador más de una vez para la misma operación. Por ejemplo, se permite INSERT OR DELETE, pero no se permite INSERT OR INSERT.
- 5 La condición WHEN no puede especificarse para los activadores INSTEAD OF.
- 6 Una sentencia de SQL compuesto (compilado) no se puede especificar si la definición de activador incluye una cláusula REFERENCING OLD TABLE o una cláusula REFERENCING NEW TABLE. Una sentencia de SQL compuesto (compilada) no se puede especificar para una definición de activador en un entorno de base de datos particionada.

Descripción**OR REPLACE**

Especifica que se debe sustituir la definición del activador si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que se sustituya la nueva definición en el catálogo. Esta opción no se tiene en cuenta si no existe una definición para el activador en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

nombre-activador

Indica el nombre del activador. El nombre, incluido en el nombre de esquema implícito o explícito, no debe identificar un activador 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

CREATE TRIGGER

base de datos los cambios ocasionados por la actualización real de la tabla sujeto. También especifica que la acción activada del activador no provocará la activación de otros activadores.

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.

INSTEAD OF

Especifica que la acción activada asociada sustituye a la acción que corresponde a la vista sujeto. Sólo está permitido un activador INSTEAD OF para cada tipo de operación de una vista sujeto determinada (SQLSTATE 428FP).

suceso-activador

Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique uno de los sucesos a la tabla sujeto o a la vista sujeto. Se puede especificar cualquier combinación de sucesos, pero cada suceso (INSERT, DELETE y UPDATE) sólo se puede especificar una vez (SQLSTATE.42613). Si se especifican varios sucesos, la acción activada debe ser una sentencia de SQL compuesto (compilado) (SQLSTATE 42601).

INSERT

Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique una operación INSERT a la tabla sujeto o a la vista sujeto.

DELETE

Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique una operación DELETE a la tabla sujeto o a la vista sujeto.

UPDATE

Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique la operación UPDATE a la tabla sujeto o a la vista sujeto, de acuerdo con las columnas especificadas o implícitas.

Si no se especifica la lista opcional *nombre-columna*, estarán implicadas todas las columnas de la tabla. Por lo tanto, la omisión de la lista *nombre-columna* supone la activación del activador cuando se actualiza cualquier columna de la tabla o la vista.

OF *nombre-columna*,...

Cada *nombre-columna* especificada debe ser una columna de la tabla base (SQLSTATE 42703). Si se trata de un activador BEFORE, el *nombre-columna* especificado no puede ser una columna generada distinta de la columna de identidad (SQLSTATE 42989). Ningún *nombre-columna* puede aparecer más de una vez en la lista *nombre-columna* (SQLSTATE 42711). El activador sólo se activará mediante la actualización de una columna identificada en la lista *nombre-columna*. Esta cláusula no puede especificarse para un activador INSTEAD OF (SQLSTATE 42613).

ON

nombre-tabla

Designa la tabla sujeto de la definición del activador BEFORE o del activador AFTER. El nombre debe especificar una tabla base o un alias que se resuelva dando como resultado una tabla base (SQLSTATE 42704 ó 42809). El nombre no debe especificar una tabla de catálogos (SQLSTATE

42832), una tabla de consulta materializada (SQLSTATE 42997), una tabla temporal creada, una tabla temporal declarada (SQLSTATE 42995) o un apodo (SQLSTATE 42809).

nombre-vista

Designa la vista sujeto de la definición del activador INSTEAD OF. El nombre debe especificar una vista sin tipo o un alias que se resuelva dando como resultado una vista sin tipo sin columnas de tipo XML (SQLSTATE 42704 o 42809). El nombre no debe especificar una vista de catálogo (SQLSTATE 42832). El nombre no debe especificar una vista que se haya definido utilizando WITH CHECK OPTION (una vista simétrica) o una vista en la que se haya definido una vista simétrica, directa o indirectamente (SQLSTATE 428FQ).

NOT SECURED o SECURED

Especifica si el activador se considera seguro. El valor por omisión es NOT SECURED.

NOT SECURED

Especifica que el activador no se considera seguro.

SECURED

Especifica que el activador se considera seguro. SECURED debe especificarse para un activador cuya tabla sujeto sea una tabla en la que el control de acceso de nivel de fila o de nivel de columna se haya activado (SQLSTATE 428H8). Asimismo, SECURED debe especificarse para un activador que se cree en una vista y una o más de las tablas subyacentes de esa definición de vista tenga el control de acceso de nivel de fila o columna activado (SQLSTATE 428H8).

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 activador. Los nombres de tabla identifican todo el conjunto de filas afectadas. Cada fila afectada por la operación SQL activador 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 previo a la operación de SQL de activador.

NEW AS nombre-correlación

Especifica un nombre de correlación que identifica el estado de la fila tal como la ha modificado la operación SQL activador y cualquier sentencia SET de un activador BEFORE que ya se ha ejecutado.

La acción activada dispone del conjunto completo de filas afectadas por la operación SQL activador 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 con anterioridad a la operación de SQL de activador. Si el suceso de activador es INSERT, la tabla temporal está vacía.

NEW TABLE AS identificador

Especifica un nombre de tabla temporal que identifica las filas afectadas tal como las ha modificado la operación SQL activador y cualquier sentencia

CREATE TRIGGER

SET de un activador BEFORE que ya se ha ejecutado. Si el suceso de activador es DELETE, la tabla temporal está vacía.

Las siguientes normas se aplican a la cláusula REFERENCING:

- Los nombres de la correlación OLD y NEW ni los nombres de OLD TABLE y NEW TABLE no pueden ser idénticos (SQLSTATE 42712).
- Para un activador solamente se puede especificar un *nombre-correlación* OLD y uno NEW (SQLSTATE 42613).
- Sólo se puede especificar un *identificador* OLD TABLE y otro NEW TABLE para un activador (SQLSTATE 42613).
- Los identificadores OLD TABLE o NEW TABLE no se pueden definir en un activador BEFORE (SQLSTATE 42898).
- Una variable de transición NEW solamente puede ser el destino de una asignación en una activación BEFORE. De lo contrario, las variables de transición no pueden ser el destino de una asignación (SQLSTATE 42703 o 42987).
- Los nombres de correlación OLD o NEW no se pueden definir en un activador 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 activador.
- Si la acción activada incluye una sentencia de SQL compuesto (compilado):
 - Los identificadores OLD TABLE o NEW TABLE no se pueden definir.
 - 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. Para una operación de inserción *nombre-correlación* captura valores nulos para cada columna de una fila.
 - En el caso de una operación de inserción o actualización, el valor de NEW captura el nuevo estado de la fila como lo proporciona la operación original y modificado por cualquier activador BEFORE que se haya ejecutado hasta ese momento. En el caso de una operación de supresión, NEW *nombre-correlación* captura valores nulos para cada columna de una fila. En un activador BEFORE DELETE, los valores no nulos asignados a las nuevas variables de transición sólo persisten dentro del activador donde ha tenido lugar la asignación.
- Si la acción activada no incluye una sentencia de SQL compuesto (compilado):
 - El *nombre-correlación* OLD y el *identificador* OLD TABLE sólo se pueden utilizar si el suceso del activador es una operación DELETE o una operación 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. Lo mismo se aplica al *identificador* OLD TABLE y al conjunto de filas afectadas.
 - El *nombre-correlación* NEW y el *identificador* NEW TABLE sólo se pueden utilizar si el suceso del activador es una operación INSERT o una operación UPDATE (SQLSTATE 42898). En ambas operaciones, el valor de NEW captura el nuevo estado de la fila como lo proporciona la operación

original y modificado por cualquier activador BEFORE que se haya ejecutado hasta ese momento. Lo mismo se aplica al *identificador* NEW TABLE y al conjunto de filas afectadas.

FOR EACH ROW

Especifica que la acción activada va a aplicarse una vez para cada fila de la tabla sujeto o de la vista sujeto que se vea afectada por la operación de SQL activador.

FOR EACH STATEMENT

Especifica que la acción activada se debe aplicar una vez para toda la sentencia. Este tipo de granularidad de activador no puede especificarse para un activador BEFORE o para un activador INSTEAD OF (SQLSTATE 42613). Si se especifica, se activará un activador UPDATE o DELETE, aunque no exista ninguna fila que se vea afectada por la sentencia UPDATE o DELETE activador.

acción-activada

Especifica la acción que se debe realizar cuando se activa un activador. Una acción activada se compone de una *sentencia-procedimiento-SQL* y de una condición opcional para la ejecución de la *sentencia-procedimiento-SQL*.

Los predicados de suceso de activador se pueden utilizar en cualquier lugar de la acción activada de una sentencia CREATE TRIGGER que utiliza una sentencia de SQL compuesto (compilado) como *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 realiza siempre.

La cláusula WHEN no puede especificarse para los activadores INSTEAD OF (SQLSTATE 42613).

Una referencia a una variable de transición con un tipo de datos XML sólo se puede utilizar en un predicado VALIDATED.

etiqueta:

Especifica la etiqueta para una sentencia de procedimiento de SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento de SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control de SQL admiten la especificación de una lista de sentencias de procedimiento de SQL.

Sólo la sentencia FOR, la sentencia WHILE y la sentencia de SQL compuesto pueden incluir una etiqueta.

sentencia-procedimiento-SQL

Especifica la sentencia de SQL que debe formar parte de la acción activada. No se da soporte a una operación de actualización de búsqueda, supresión de búsqueda, inserción o fusión de apodos en SQL compuesto.

CREATE TRIGGER

La acción activada de un activador BEFORE en una columna de tipo XML puede invocar la función XMLVALIDATE a través de una sentencia SET, puede dejar los valores de tipo XML sin cambiar o asignarlos a NUL mediante una sentencia SET.

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), a un objeto federado (SQLSTATE 42997) o a una tabla temporal declarada (SQLSTATE 42995), o las columnas de inicio y de fin del período BUSINESS_TIME (SQLSTATE 42808).

La *sentencia-procedimiento-SQL* en un activador BEFORE no puede:

- Contener ninguna operación INSERT, DELETE o UPDATE ni invocar ninguna rutina definida con MODIFIES SQL DATA, si no es de SQL compuesto SQL (compilado).
- Contener ninguna operación DELETE o UPDATE en la tabla sujeto de activador ni invocar ninguna rutina que contenga dichas operaciones, si es de SQL compuesto (compilado).
- Hacer referencia a una tabla de consulta materializada definida con REFRESH IMMEDIATE (SQLSTATE 42997)
- Hacer referencia a una columna generada que no sea la columna de identidad de la variable de transición NEW (SQLSTATE 42989).

Notas

- Al añadir un activador a una tabla que ya contiene filas, no provocará la activación de ninguna acción activada. Así pues, si el activador tiene como objetivo imponer las restricciones de la tabla, es posible que las filas existentes no cumplan dichas restricciones.
- Si los sucesos para dos activadores tienen lugar simultáneamente (por ejemplo, si tienen el mismo suceso, tiempo de activación y tablas sujeto), el primer activador creado es el primero en ejecutarse. Si se utiliza la opción OR REPLACE para sustituir un activador creado anteriormente, la hora de creación cambia y, por lo tanto, puede afectar al orden de ejecución del activador.
- Si se añade una columna a la tabla sujeto cuando ya se han definido los activadores, se aplican las siguientes normas:
 - Si se trata de un activador UPDATE que se ha especificado sin una lista de columnas explícita, cualquier actualización de una columna nueva provocará la activación del activador.
 - La columna no estará visible en la acción activada de cualquier activador definido con anterioridad.
 - Las tablas de transición OLD TABLE y NEW TABLE no contendrán 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.
- Si un objeto al que se hace referencia en el cuerpo del activador no existe o se ha marcado como no válido o el definidor no tiene temporalmente los privilegios para acceder al objeto y, si el parámetro de configuración de la base de datos **auto_reval** no se ha establecido en DISABLED, el activador se creará satisfactoriamente igualmente. El activador se marcará como no válido y se volverá a validar la siguiente vez que se invoque.

- El resultado de una selección completa especificada en una *sentencia-procedimiento-SQL* no está disponible dentro ni fuera del activador.
- Un procedimiento que se llama desde una sentencia compuesta activada no debe emitir las sentencias COMMIT ni ROLLBACK (SQLSTATE 42985).
- No se da soporte a un procedimiento que contiene una referencia a un apodo en una sentencia UPDATE de búsqueda, una sentencia DELETE de búsqueda o una sentencia INSERT (SQLSTATE 25000).
- **Restricciones de acceso a las tablas:** si un procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del procedimiento puede acceder a una tabla que la sentencia compuesta que ha invocado el procedimiento esté modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del procedimiento puede modificar una tabla que la sentencia compuesta que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).
- Un activador BEFORE DELETE definido en una tabla implicada en un ciclo de restricciones de referencia en cascada no debe 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 activador son datos dependientes y, por lo tanto, tal vez no produzcan resultados coherentes.

En su forma más simple, significa que un activador BEFORE DELETE de una tabla con una restricción de referencia a sí misma y una norma de supresión CASCADE no debe incluir ninguna referencia a la tabla en la *acción-activada*.

- La creación de un activador hace que se marquen determinados paquetes como no válidos:
 - Si se crea un activador UPDATE sin una lista de columnas explícita, se invalidan los paquetes que utilizan la actualización en la tabla o la vista de destino.
 - Si se crea un activador UPDATE con una lista de columnas, los paquetes que utilizan la actualización en la tabla de destino sólo se invalidan si el paquete también utiliza la actualización en como mínimo una columna de la lista *nombre-columna* de la sentencia CREATE TRIGGER.
 - Si se crea un activador INSERT, se invalidan los paquetes que utilizan la inserción en una tabla o vista de destino.
 - Si se crea un activador de supresión, los paquetes que hagan uso de la supresión en la tabla o vista de destino se invalidarán.
- 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.
- **Activadores no operativos:** un *activador no operativo* es un activador que ya no está disponible y que, por lo tanto, nunca se activará. Un activador deja de ser operativo si:
 - se revoca un privilegio que el creador del activador debe tener para que se ejecute el activador
 - se descarta un objeto, por ejemplo una tabla, una vista o un alias, del que depende la acción activada
 - deja de estar operativa una vista de la que depende la acción activada
 - se descarta un alias que es la tabla sujeto del activador.

En otras palabras, un activador no operativo es aquél en el que se ha descartado la definición de un activador a consecuencia de las normas en cascada de las sentencias DROP y REVOKE. Por ejemplo, cuando se descarta una vista, los

CREATE TRIGGER

activadores que tienen una *sentencia-procedimiento-SQL* que contiene una referencia a esa vista se establecen como no operativos.

Cuando un activador deja de ser operativo, todos los paquetes con sentencias que realicen operaciones y que estuvieran activando el activador quedarán marcados como no válidos. Cuando el paquete se vuelve a enlazar (explícita o implícitamente), se ignora por completo el activador no operativo. De igual forma, las aplicaciones con sentencias de SQL dinámico que realicen operaciones y que estuvieran activando el activador también pasarán por alto por completo todos los activadores que no sean operativos.

El nombre del activador puede seguirse especificando en las sentencias DROP TRIGGER y COMMENT ON TRIGGER.

Es posible volver a crear un activador no operativo emitiendo una sentencia CREATE TRIGGER y utilizando el texto de definición del activador no operativo. Este texto de definición de activador se almacena en la columna TEXT de la vista de catálogo SYSCAT.TRIGGERS. Observe que no es necesario descartar de manera explícita el activador no operativo para volver a crearlo. La emisión de una sentencia CREATE TRIGGER con el mismo *nombre-activador* que un activador no operativo hará que se sustituya dicho activador no operativo con un aviso (SQLSTATE 01595).

Los activadores no operativos se señalan con una X en la columna VALID de la vista de catálogo SYSCAT.TRIGGERS.

- **Errores que se producen al ejecutar activadores:** los errores que se producen durante la ejecución de sentencias de SQL activadas se devuelven mediante SQLSTATE 09000 a menos 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 activador, 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, la SQLSTATE devuelta es la especificada en la sentencia SIGNAL SQLSTATE o la condición RAISE_ERROR.

- Crear un activador con un nombre de esquema que todavía no existe dará como resultado la creación implícita del esquema siempre y cuando el ID de autorización de la sentencia tenga la autoridad IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- **Columna DB2SECURITYLABEL:** Se puede hacer referencia a una columna DB2SECURITYLABEL en el cuerpo de activador de BEFORE TRIGGER, pero no se puede cambiar en el cuerpo de un activador BEFORE (SQLSTATE 42989).
- **Columnas del período BUSINESS_TIME:** Las columnas de inicio y de fin de un período BUSINESS_TIME no se pueden cambiar en el cuerpo de un activador BEFORE UPDATE (SQLSTATE 42808).
- **Vistas de sólo lectura:** añadir el activador INSTEAD OF para una vista influye en las características de sólo lectura de la vista. Si una vista de sólo lectura tiene una relación de dependencia con un activador INSTEAD OF, el tipo de operación que se define para el activador INSTEAD OF define si se puede suprimir, insertar o actualizar la vista.
- **Valores de las variables de transición y activadores INSTEAD OF:** los valores iniciales de las nuevas variables de transición o de las nuevas columnas de la tabla de transición que se pueden ver en un activador INSTEAD OF INSERT se establecen como se indica a continuación:
 - Si se especifica explícitamente un valor para una columna en la operación de inserción, la variable de transición nueva correspondiente es dicho valor especificado explícitamente.

- Si no se especifica explícitamente un valor para una columna en la operación de inserción o se especifica la cláusula DEFAULTS, la variable de transición nueva correspondiente es:
 - El valor por omisión de la columna de tabla subyacente si la columna de vista se puede actualizar (sin el activador INSTEAD OF)
 - De lo contrario, será el valor nulo

Los valores iniciales de las nuevas variables de transición que se pueden ver en un activador INSTEAD OF UPDATE se establecen como se indica a continuación:

- Si se especifica un valor explícitamente para una columna de la operación de actualización, la nueva variable de transición correspondiente es dicho valor especificado explícitamente.
 - Si se especifica explícitamente la cláusula DEFAULT para una columna de la operación de actualización, la nueva variable de transición correspondiente es:
 - El valor por omisión de la columna de tabla subyacente si la columna de vista se puede actualizar (sin el activador INSTEAD OF)
 - De lo contrario, será el valor nulo
 - De lo contrario, la nueva variable de transición correspondiente es el valor existente de la columna en la fila.
- **Activadores y tablas con tipo:** Un activador BEFORE o AFTER puede asociarse a una tabla con tipo en cualquier nivel de una jerarquía de tablas. Si una sentencia de SQL activa varios activadores, estos se ejecutarán en el orden en que fueron creados, aunque estén asociados a tablas diferentes de la jerarquía de tablas con tipo.

Cuando se activa un activador, 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:

- **Activadores de fila:** Cuando se utiliza una sentencia de SQL para insertar, actualizar o suprimir una fila de tabla, la sentencia activa activadores de fila asociados a la tabla más específica donde reside la fila, y los activadores asociados a todas las supertablas de esa tabla. Esta norma 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 activadores de fila asociados a EMP y a sus supertablas. Para las filas de MGR, se activan los activadores de fila asociados a MGR y a sus supertablas.
- **Activadores de sentencia:** Las sentencias INSERT, UPDATE o DELETE activan activadores de sentencia asociados a las tablas (y a sus supertablas) que podrían estar afectados por la sentencia. Esta norma 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 activadores de sentencia para EMP y sus supertablas. Otro ejemplo: En un mandato UPDATE EMP o DELETE EMP, se activan los activadores 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 activadores de sentencia para EMP y sus supertablas, pero no activadores de sentencia para subtablas.

Efectos de las sentencias DROP TABLE: Una sentencia DROP TABLE ("descartar tabla") no activa ningún activador asociado a la tabla que se descarta. En

CREATE TRIGGER

cambio, si la tabla descartada es una subtabla, todas las filas de la tabla descartada son elegibles para ser suprimidas de sus supertablas. Por lo tanto, para una tabla T:

- Activadores de fila: DROP TABLE T activa activadores de supresión de filas que están asociados a todas las supertablas de T, para cada fila de T.
- Activadores de sentencia: DROP TABLE T activa activadores 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 activadores 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 activadores 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 activadores 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 activadores de sentencia para T1 y para todas sus subtablas y supertablas.
 4. 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 activadores de sentencia sólo para T1 y sus supertablas.
- **Sentencia y activadores MERGE:** La sentencia MERGE puede ejecutar operaciones de actualización, supresión e inserción. Los activadores UPDATE, DELETE o INSERT aplicables se activan para la sentencia MERGE cuando se ejecuta una operación de actualización, supresión o inserción.
 - **Ofuscación:** La sentencia CREATE TRIGGER se puede enviar en formato ofuscado. En una sentencia ofuscada, sólo es legible en nombre del activador. El resto de la sentencia se codifica de tal manera que no es legible, pero que el servidor de bases de datos puede descodificar. Las sentencias ofuscadas se pueden generar mediante una llamada a la función DBMS_DDL.WRAP.
 - **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - OLD_TABLE se puede especificar en lugar de OLD TABLE, y NEW_TABLE se puede especificar en lugar de NEW TABLE
 - Se puede especificar MODE DB2SQL a continuación de FOR EACH ROW o FOR EACH STATEMENT
 - **Creación de un activador con la opción SECURED:** Por lo general los usuarios con autorización SECADM no tienen privilegios para crear objetos de base de datos como funciones definidas por el usuario o activadores. Suelen examinar los datos a los que accede un activador, comprobar que son seguros y, a continuación, otorgar la autorización CREATE_SECURE_OBJECT a quien ya

tenga los privilegios necesarios para crear el activador seguro. Una vez creado el activador, revocarán la autorización CREATE_SECURE_OBJECT al propietario del activador.

El activador se considera seguro. El gestor de bases de datos considera el atributo SECURED como una aserción que declara que el usuario ha establecido un procedimiento de auditoría para todas las actividades del cuerpo del activador. Si un activador seguro hace referencia a funciones definidas por el usuario, el gestor de base de datos presupone que esas funciones son seguras sin validación. Si esas funciones pueden acceder a datos confidenciales, el usuario con autorización SECADM tiene que asegurarse de que dichas funciones puedan acceder a los datos y de que todas las sentencias ALTER FUNCTION posteriores o los cambios en los paquetes externos sean revisados por este proceso de auditoría.

Un activador debe ser seguro si su tabla sujeto tiene el control de acceso de nivel de fila o columna activado. Asimismo, un activador debe ser seguro si su tabla sujeto es una vista y una o más de las tablas subyacentes de esa definición de vista tiene el control de acceso de nivel de fila o columna activado.

- **Creación de un activador con la opción NOT SECURED:** la sentencia CREATE TRIGGER devuelve un error si la tabla sujeto del activador tiene el control de acceso de nivel de fila o columna activado. Asimismo, la sentencia CREATE TRIGGER falla si el activador está definido en una vista y una o más de las tablas subyacentes de esa definición de vista tiene el control de acceso de nivel de fila o columna activado.
- **Control de acceso a filas y columnas que no se implanta para las variables de transición y las tablas de transición:** los activadores se utilizan para la integridad de la base de datos y, de este modo, se hace necesario el equilibrio entre la seguridad y la integridad de la base de datos. Si el control de acceso de nivel de fila o columna se activa en la tabla sujeto o una tabla subyacente de la vista sujeto, no se aplican los permisos de fila ni las máscaras de columna a los valores iniciales de las variables de transición y las tablas de transición. El control de acceso de nivel de fila y columna que se implanta para la tabla sujeto o una tabla subyacente de la vista sujeto también se ignora para las variables de transición y las tablas de transición a las que se hace referencia en el cuerpo del activador o que se pasan como argumentos a las funciones definidas por el usuario que se invocan en el cuerpo del activador. Para que no exista ningún contratiempo de seguridad para las sentencias de SQL en la acción que el activador realiza para acceder a datos confidenciales en las variables de transición y las tablas de transición, el activador debe crearse con la opción SECURED. Si un activador no es seguro, la sentencia CREATE TRIGGER devuelve un error.
- **Consideraciones para columnas ocultas implícitamente:** existe una variable de transición para cualquier columna definida como implícitamente oculta. En el cuerpo de un activador, se puede hacer referencia a una variable de transición que se corresponda con una columna oculta implícitamente.

Ejemplos

- *Ejemplo 1:* cree dos activadores que den como resultado un seguimiento automático del número de empleados que gestiona una empresa. Los activadores 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.

CREATE TRIGGER

El primer activador 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
UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

El segundo activador 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
UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

- *Ejemplo 2:* cree un activador que asegure que siempre que se actualice un registro de piezas se lleve a cabo la siguiente acción de comprobación y de 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 activador 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 un formulario de pedido de piezas adicionales a la empresa adecuada.

```
CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW AS N
FOR EACH ROW
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:* Repita el supuesto del Ejemplo 2, pero utilice una sentencia completa en lugar de una sentencia VALUES para invocar la función definida por el usuario. Este ejemplo también muestra cómo definir el activador como un activador de sentencia en lugar de un activador de fila. Para cada fila de la tabla de transición que se evalúa como verdadera para la cláusula WHERE, se emite una petición de entrega para la pieza.

```
CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW TABLE AS NTABLE
FOR EACH STATEMENT
BEGIN ATOMIC
SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
FROM NTABLE
WHERE (ON_HAND < 0.10 * MAX_STOCKED);
END
```

- *Ejemplo 4:* cree un activador que provoque 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
WHEN (N.SALARY > 1.1 * O.SALARY)
SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT='Incremento de salario>10%'
```

- *Ejemplo 5:* suponga una aplicación que registre y haga seguimientos 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)

Cuando la columna QUOTE de CURRENTQUOTE se actualiza, la nueva oferta (quote en inglés) debe copiarse con una indicación de fecha y hora, 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 activador 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
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
       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 activador 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
BEGIN ATOMIC
  INSERT INTO QUOTEHISTORY
  VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT_TIMESTAMP);
END
```

- *Ejemplo 6:* cree un activador que altere temporalmente los cambios en el campo de ubicación del registro de empleados de la tabla org. Este activador puede ser útil si se procesan nuevos registros de empleados obtenidos cuando se adquirió una empresa más pequeña y la ubicación de destino asignada al empleado es 'Toronto' y la nueva ubicación de destino es 'Los Angeles'. El activador before asegurará que independientemente del valor que la aplicación asigne para este campo, el valor resultante final será 'Los Angeles'.

CREATE TRIGGER

```
CREATE TRIGGER LOCATION_TRIGGER
NO CASCADE
BEFORE UPDATE ON ORG
REFERENCING
  OLD AS PRE
  NEW AS POST
FOR EACH ROW
WHEN (POST.LOCATION = 'Toronto')
  SET POST.LOCATION = 'Los Angeles';
END
```

- *Ejemplo 7:* cree un activador BEFORE que valide automáticamente los documentos XML que contengan descripciones de productos nuevos antes de que se inserten en la tabla PRODUCT de la base de datos SAMPLE:

```
CREATE TRIGGER NEWPROD NO CASCADE BEFORE INSERT ON PRODUCT
REFERENCING NEW AS N
FOR EACH ROW
BEGIN ATOMIC
  SET (N.DESCRPTION) = XMLVALIDATE(N.DESCRPTION
  ACCORDING TO XMLSCHEMA ID producto);
END
```

CREATE TRUSTED CONTEXT

La sentencia CREATE TRUSTED CONTEXT define un contexto fiable en el servidor actual.

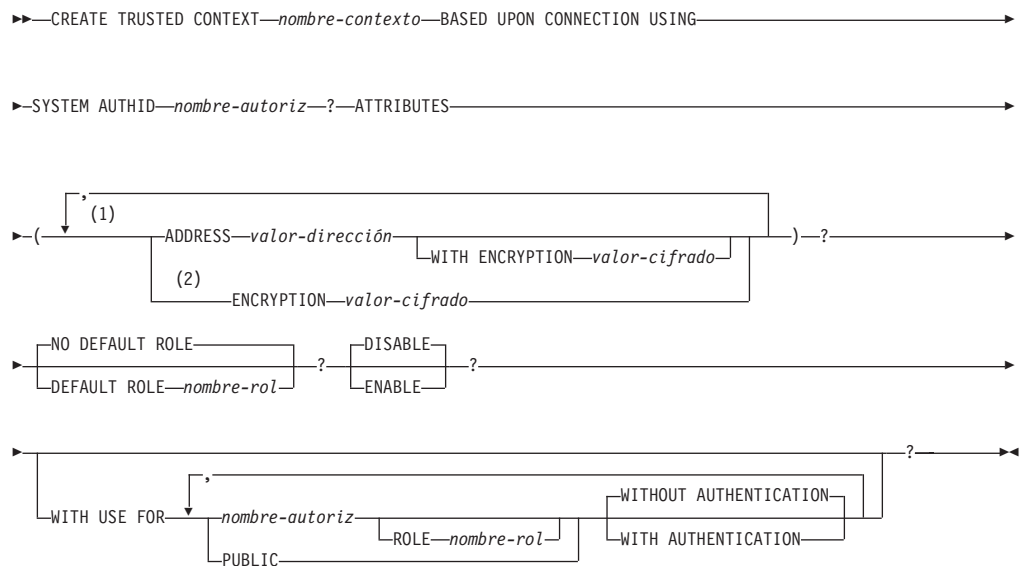
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



Notas:

- 1 Cada combinación de un nombre de atributo y su valor correspondiente, en forma de par, debe ser exclusiva (SQLSTATE 4274D).
- 2 ENCRYPTION no se puede especificar más de una vez (SQLSTATE 42614); sin embargo, WITH ENCRYPTION puede especificarse para cada una de las veces que se especifique ADDRESS.

Descripción

nombre-contexto

Da nombre al contexto fiable. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar un contexto fiable que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

CREATE TRUSTED CONTEXT

BASED UPON CONNECTION USING SYSTEM AUTHID *nombre-autoriz*

Especifica que el contexto es una conexión establecida por medio del ID de autorización del sistema *nombre-autorización*, que no debe asociarse con un contexto fiable existente (SQLSTATE 428GL). No puede ser el ID de autorización de la sentencia (SQLSTATE 42502).

ATTRIBUTES (...)

Especifica una lista de uno o más atributos de confianza de conexión sobre la que se define el contexto fiable.

ADDRESS *valor-dirección*

Especifica la dirección de comunicación real que utiliza el cliente para comunicar con el servidor de bases de datos. El único protocolo soportado es TCP/IP. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.
- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de contexto fiable si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio. Al crear un objeto de contexto fiable, resulta ventajoso proporcionar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP). Con DHCP, un dispositivo puede disponer de una dirección de IP diferente cada vez que se conecte a la red. Por tanto, si se

proporciona una dirección IP estática para el atributo ADDRESS de un objeto de contexto fiable, es posible que algún dispositivo adquiera alguna conexión fiable de modo no intencional. Proporcionar nombres de dominio para el atributo ADDRESS de un objeto de contexto fiable evita este problema en entornos DHCP.

WITH ENCRYPTION *valor-cifrado*

Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser uno de los valores siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autenticación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse un cifrado SSL (Secure Sockets Layer) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 si una conexión entrante debe coincidir con el valor de cifrado para esta dirección específica

ENCRYPTION *valor-cifrado*

Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red. El valor por omisión es NONE.

valor-cifrado

Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser uno de los valores siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico para una conexión de entrada que coincida con el atributo ENCRYPTION de este objeto de contexto fiable
- LOW, se necesita un cifrado ligero mínimo; el tipo de autenticación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable
- HIGH, debe utilizarse un cifrado SSL (Secure Sockets Layer) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 si una conexión de entrada debe coincidir con el atributo ENCRYPTION de este objeto de contexto fiable

La tabla siguiente resume el momento en que puede utilizarse un contexto fiable, en función del cifrado utilizado por la conexión existente. Si el contexto fiable no puede utilizarse para la conexión, se devuelve un aviso (SQLSTATE 01679) y el campo SQLWARN8 de SQLCA se establece en 'Y', el cual indica que la conexión es una conexión regular (no fiable).

CREATE TRUSTED CONTEXT

Tabla 27. Cifrado y contextos fiables

Cifrado utilizado por la conexión existente	Valor de ENCRYPTION para el contexto fiable	¿Puede utilizarse el contexto fiable para la conexión?
No hay cifrado	'NONE'	Sí
No hay cifrado	'LOW'	No
No hay cifrado	'HIGH'	No
Cifrado bajo (DATA_ENCRYPT)	'NONE'	Sí
Cifrado bajo (DATA_ENCRYPT)	'LOW'	Sí
Cifrado bajo (DATA_ENCRYPT)	'HIGH'	No
Cifrado alto (SSL)	'NONE'	Sí
Cifrado alto (SSL)	'LOW'	Sí
Cifrado alto (SSL)	'HIGH'	Sí

NO DEFAULT ROLE o **DEFAULT ROLE** *nombre-rol*

Especifica si se asocia o no un rol por omisión con una conexión fiable basada en este contexto fiable. El valor por omisión es NO DEFAULT ROLE.

NO DEFAULT ROLE

Especifica que el contexto fiable no tiene un rol por omisión.

DEFAULT ROLE *nombre-rol*

Especifica que *nombre-rol* es el rol por omisión para el contexto fiable. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). Este rol se utiliza con el usuario de una conexión fiable, basada en este contexto fiable, cuando el usuario no tenga definido un rol específico de usuario como parte de la definición del contexto fiable.

DISABLE o **ENABLE**

Especifica si se crea o no el contexto fiable en el estado habilitado o inhabilitado. El valor por omisión es DISABLE.

DISABLE

Especifica que se cree el contexto fiable en el estado inhabilitado. Un contexto fiable que esté inhabilitado no se tomará en consideración cuando se establezca una conexión fiable.

ENABLE

Especifica que se cree el contexto fiable en el estado habilitado.

WITH USE FOR

Especifica los usuarios que pueden utilizar una conexión fiable basada en este contexto fiable.

nombre-autoriz

Especifica que la conexión fiable puede utilizarse por medio del *nombre-autorización* especificado. El *nombre-autorización* no debe especificarse más de una vez con la cláusula WITH USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502). Si la definición de un contexto fiable permite el acceso tanto por parte de PUBLIC como por parte de una lista de usuarios, las especificaciones para un usuario alterarán temporalmente las especificaciones para PUBLIC. Por ejemplo, suponga que se define un contexto fiable que permite el acceso

tanto de PUBLIC WITH AUTHENTICATION como de JOE WITHOUT AUTHENTICATION. Si JOE utiliza el contexto fiable, no se necesitará la autenticación. Sin embargo, si GEORGE utiliza el contexto fiable, se necesitará la autenticación.

ROLE *nombre-rol*

Especifica que *nombre-rol* es el rol que ha de utilizarse para el usuario cuando una conexión fiable está utilizando el contexto fiable. El *nombre-rol* debe identificar un rol que exista en el servidor actual (SQLSTATE 42704). El rol explícitamente especificado para el usuario altera temporalmente los roles por omisión asociados con el contexto fiable.

PUBLIC

Especifica que cualquier usuario puede utilizar una conexión fiable basada en este contexto fiable. PUBLIC no debe especificarse más de una vez (SQLSTATE 428GM). Todos los usuarios que utilicen dicha conexión fiable hacen uso de los privilegios asociados con el rol por omisión para el contexto fiable asociado. Si no se define un rol por omisión para el contexto fiable, no habrá ningún rol asociado con los usuarios que usen una conexión fiable basada en este contexto fiable.

WITHOUT AUTHENTICATION o **WITH AUTHENTICATION**

Especifica si conmutar el usuario en una conexión fiable requiere autenticación del usuario o no. El valor por omisión es WITHOUT AUTHENTICATION.

WITHOUT AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable para este usuario no requiere autenticación.

WITH AUTHENTICATION

Especifica que conmutar el usuario actual en una conexión fiable para este usuario requiere autenticación.

Normas

- Una sentencia de SQL exclusiva del contexto fiable debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas del contexto fiable son:
 - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT o DROP (TRUSTED CONTEXT)
- Una sentencia de SQL exclusiva del contexto fiable no puede emitirse en una transacción global; por ejemplo, una transacción XA o una transacción global que se inicie como parte de una confirmación en dos fases para las transacciones federadas (SQLSTATE 51041).

Notas

- Al proporcionar una dirección IP como parte de una definición de contexto fiable, la dirección debe estar en el formato que esté vigente para la red. Por ejemplo, proporcionar una dirección en un formato IPv6 cuando la red sea IPv4 no dará como resultado una coincidencia. En un entorno mixto, resulta ventajoso especificar las representaciones IPv4 e IPv6 de la dirección, o mejor aún, especificar un nombre de dominio seguro (por ejemplo, corona.torolab.ibm.com), que oculta los detalles de formato de dirección.
- *Especificar un rol en la definición de un contexto fiable:* La definición de un contexto fiable puede designar un rol para un ID de autorización específico y un rol por omisión a utilizar para los ID de autorización para los que no se haya

CREATE TRUSTED CONTEXT

especificado un rol específico en la definición del contexto fiable. Este rol puede utilizarse con una conexión fiable basada en el contexto fiable, pero no hará que el rol esté disponible fuera de una conexión fiable basada en el contexto fiable.

- Al emitir una sentencia de SQL de lenguaje de manipulación de datos (DML) que utilice una conexión fiable, se tomarán en consideración los privilegios que mantenga un rol asignado por el contexto vigentes para el ID de autorización en la definición del contexto fiable asociado, además de otros privilegios que mantenga directamente el ID de autorización de la sentencia, u otros roles que mantenga indirectamente el ID de autorización de la sentencia.
- No se tomarán en consideración, para las sentencias de SQL de lenguaje de definición de datos (DDL), los privilegios que mantenga un rol asignado por el contexto vigentes para el ID de autorización en la definición del contexto fiable asociado. Por ejemplo, para crear un objeto, el ID de autorización de la sentencia debe poder efectuar dicha acción sin incluir los privilegios que mantenga el rol asignado por el contexto.
- Al instalar una aplicación nueva que autentifique en DB2 utilizando las mismas credenciales que una aplicación existente en la misma máquina y que saque partido de un contexto fiable, es posible que la aplicación nueva también saque partido del mismo objeto de contexto fiable (por ejemplo, heredando el rol de contexto fiable). Es posible que no sea esta la intención del administrador de seguridad. Es posible que el administrador de seguridad desee activar el recurso de auditoría de DB2 para averiguar que aplicaciones están sacando partido de los objetos de contexto fiable.
- Sólo se permite una sentencia de SQL exclusiva del contexto fiable sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva del contexto fiable sin confirmar, las siguientes sentencias de SQL exclusivas del contexto fiable esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva del contexto fiable actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

- *Ejemplo 1:* Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a dos ID de usuario diferentes. Cuando el usuario actual de la conexión se conmuta al ID de usuario JOE, no se requiere la autenticación. Sin embargo, la autenticación se requiere cuando el usuario actual de la conexión se conmuta al ID de usuario BOB. Tenga en cuenta que el contexto fiable tiene un rol por omisión llamado *rol-contexto*. Esto implica que los usuarios que trabajen dentro de los confines de este contexto fiable heredarán los privilegios asociados con el rol *rol-contexto*.

```
CREATE TRUSTED CONTEXT APPSERVER
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  DEFAULT ROLE CONTEXT_ROLE
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR JOE WITHOUT AUTHENTICATION
  BOB WITH AUTHENTICATION
```

- *Ejemplo 2:* Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a cualquier ID de usuario sin autenticación.

```
CREATE TRUSTED CONTEXT SECUREROLE
  BASED UPON CONNECTION USING SYSTEM AUTHID PBIRD
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

CREATE TRUSTED CONTEXT

- *Ejemplo 3:* Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a cualquier ID de usuario sin autenticación. La diferencia entre este contexto fiable y el contexto fiable creado en el ejemplo 2, es que este contexto fiable tiene un atributo adicional llamado ENCRYPTION. El valor del atributo ENCRYPTION para el contexto fiable SECUREROLEENCRYPT indica que el valor de cifrado que utilice una conexión debe tener como mínimo un "cifrado bajo" (consulte la Tabla 27 en la página 868) para coincidir con este atributo de contexto fiable.

```
CREATE TRUSTED CONTEXT SECUREROLEENCRYPT
  BASED UPON CONNECTION USING SYSTEM AUTHID SHARPER
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204'
    ENCRYPTION 'LOW')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

- *Ejemplo 4:* Cree un contexto fiable tal que las conexiones efectuadas por el usuario WRJAIBI desde las direcciones 9.26.146.201 y 9.26.146.203 resulten fiables cuando no se utilice ningún cifrado, pero en el que una conexión efectuada por el usuario WRJAIBI desde la dirección 9.26.146.202 requiera un nivel de cifrado LOW para resultar fiable.

```
CREATE TRUSTED CONTEXT WALIDLOCSENSITIVE
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.146.201',
    ADDRESS '9.26.146.202' WITH ENCRYPTION 'LOW',
    ADDRESS '9.26.146.203'
    ENCRYPTION 'NONE')
```

CREATE TYPE

La sentencia CREATE TYPE define un tipo de datos definido por el usuario en el servidor actual.

Mediante la utilización de esta sentencia pueden crearse cinco tipos distintos de tipos de datos definidos por el usuario. Cada uno de los tipos se describe por separado.

- **Matriz.** Tipo de datos definido por el usuario que es una matriz ordinaria o una matriz asociativa. Los elementos de un tipo de matriz se basan en uno de los tipos de datos incorporados o en un tipo definido por el usuario que no sea un tipo de cursor ni un tipo estructurado.
- **Cursor.** Tipo de datos definido por el usuario que es un tipo de cursor.
- **Diferenciado.** Un tipo de datos definido por el usuario que deriva de uno de los tipos de datos incorporados y que puede definirse para utilizar normas de tipificación firme o normas de tipificación no firme.. Cuando se crea un tipo diferenciado con tipo firme, se generan funciones que realizan la conversión entre el tipo diferenciado definido por el usuario y el tipo de datos incorporado fuente. Opcionalmente, al crearse el tipo diferenciado definido por el usuario, puede generarse soporte para las operaciones de comparación que han de utilizarse con el tipo diferenciado con tipo firme.
- **Fila.** Tipo de datos definido por el usuario que representa una fila. Incluye uno o varios campos con los tipos de datos asociados que componen una fila de datos.
- **Estructurado.** Tipo de datos definido por el usuario que representa un objeto y los métodos asociados. Puede incluir cero o más atributos y puede ser un subtipo que permita que los atributos puedan heredarse de un supertipo. Algunos métodos se generan al crearse el tipo estructurado definido por el usuario y otros pueden especificarse como parte de la definición.

CREATE TYPE (matriz)

La sentencia CREATE TYPE (matriz) define un tipo de matriz. Los elementos de un tipo de matriz se basan en uno de los tipos de datos incorporados o en un tipo diferenciado definido por el usuario.

Invocación

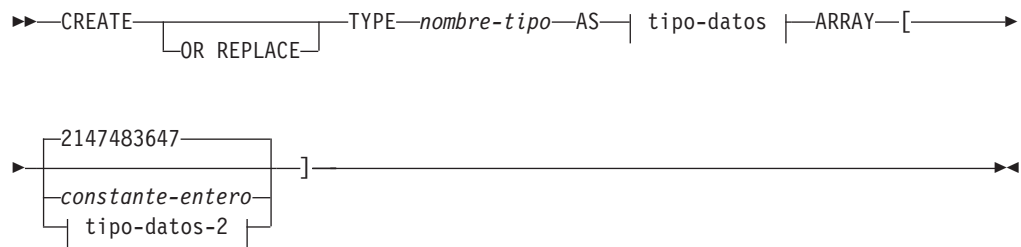
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo de matriz no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo de matriz hace referencia a un esquema existente
- Autorización DBADM

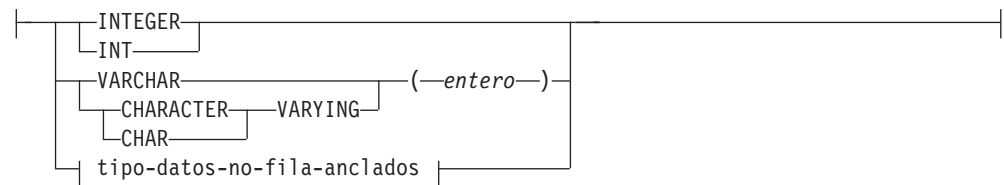
Sintaxis



tipo-datos:

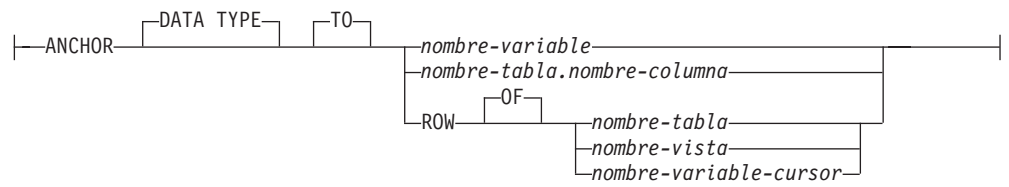


tipo-datos-2:



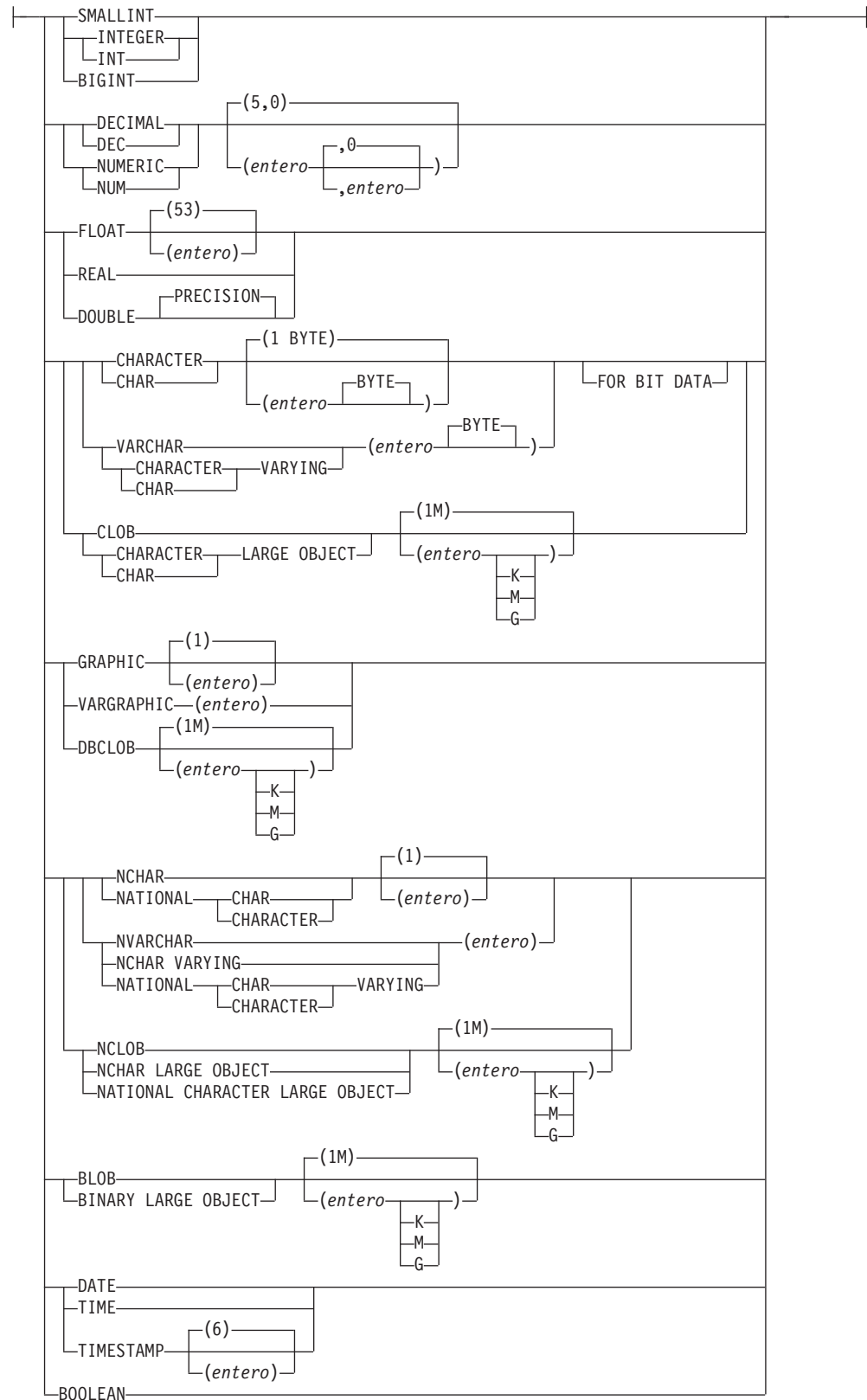
tipo-datos-anclados:

CREATE TYPE (matriz)



tipo-incorporado:

CREATE TYPE (matriz)



tipo-datos-no-fila-anclados:

CREATE TYPE (matriz)

```
|—ANCHOR—DATA TYPE—TO—| nombre-variable ————— |  
|                               | nombre-tabla.nombre-columna |
```

Descripción

OR REPLACE

Especifica que ha de sustituirse la definición del tipo de datos si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que las funciones y métodos se invalidan en lugar de descartarse cuando éstos tienen parámetros o un valor de retorno definido con el tipo de datos que se sustituye. La definición existente no debe ser de un tipo estructurado (SQLSTATE 42809). Esta opción se pasa por alto si no existe una definición para el tipo de datos en el servidor actual.

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe designar ningún otro tipo (incorporado o definido por el usuario) que ya exista en el servidor actual. El nombre sin calificar no debe ser el mismo que un tipo de datos incorporado o BOOLEAN, BINARY o VARBINARY (SQLSTATE 42918).

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). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Si se especifica un nombre compuesto de dos partes *nombre-tipo*, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos

Especifica el tipo de datos de los elementos de matriz.

tipo-incorporado

Especifica un tipo de datos incorporado. Consulte en "CREATE TABLE" la descripción de los tipos de datos incorporados. Los tipos incorporados incluyen los tipos de datos descritos en "CREATE TABLE", que no sean los tipos de referencia, SYSPROC.DB2SECURITYLABEL, XML o los tipos definidos por el usuario (SQLSTATE 429C2).

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. Si se especifica un *nombre-tipo-fila* sin un nombre de esquema, el *nombre-tipo-fila* se resuelve buscando los esquemas en la vía de acceso de SQL. Los tipos de fila se pueden anidar como elementos en otros tipos de matriz con un nivel máximo de anidamiento de dieciséis.

nombre-tipo-matriz

Especifica un tipo de matriz. Si se especifica un *nombre-tipo-matriz* sin un nombre de esquema, el *nombre-tipo-matriz* se resuelve buscando los esquemas en la vía de acceso de SQL. Los tipos de matriz se pueden anidar como elementos en otros tipos de matriz con un nivel máximo de anidamiento de dieciséis.

tipo-datos-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos. El tipo de datos del objeto de anclaje se ve afectado por las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente o, en el caso de una fila, cuando se crea un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable global se utiliza como tipo de datos para los elementos de matriz.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para los elementos de matriz.

ROW OF *nombre-tabla* o *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. El tipo de datos de los elementos de matriz es un tipo de fila sin nombre.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los elementos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula **CONSTANT** especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable de cursor no es de un tipo firme que utiliza un tipo de fila con nombre, el tipo de datos de los elementos de matriz es un tipo de fila sin nombre.

tipo-datos-no-fila-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos. El tipo de datos del objeto de anclaje se ve afectado por las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global con un tipo de datos **INTEGER** o **VARCHAR**. El tipo de datos de la variable global se utiliza como tipo de datos para el índice de matriz.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente con un tipo de datos **INTEGER** o **VARCHAR**. El tipo de datos de la columna se utiliza como tipo de datos para el índice de matriz.

ARRAY [*constante-entero*]

Especifica que el tipo es una matriz con una cardinalidad máxima de *constante-entero*. El valor debe ser un entero positivo (diferente de cero) e inferior al mayor valor entero positivo (SQLSTATE 42820). El valor por omisión

CREATE TYPE (matriz)

es el mayor valor entero positivo (2 147 483 647). La cardinalidad del valor de una matriz se determina mediante la posición de elemento más alta que se ha asignado al valor de la matriz.

La cardinalidad máxima de una matriz de un determinado sistema está limitada por la cantidad total de memoria disponible para las aplicaciones DB2. como tal, aunque puedan crearse matrices con grandes cardinalidades, es posible que no puedan utilizarse todos los elementos.

ARRAY[*tipo2-datos*]

Especifica que el tipo es una matriz asociativa que está indexada según los valores de tipo de datos *tipo2-datos*. El tipo de datos debe ser el tipo de datos INTEGER o VARCHAR (SQLSTATE 429C2). Los valores especificados como índice al asignar un elemento de matriz deben poder asignarse a un valor de *tipo2-datos*. La cardinalidad del valor de una matriz se determina mediante el número de valores de índice exclusivos que se utilizan al asignar elementos de matriz.

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.

Notas

- **Uso del tipo de matriz:** el tipo de matriz sólo puede utilizarse como el tipo de datos de:
 - Una variable local en una sentencia de SQL compuesto (compilado)
 - Un parámetro de una rutina de SQL
 - Un parámetro de procedimiento Java (sólo matrices comunes no anidadas)
 - El tipo de retorno de una función de SQL
 - Una variable global
- Una variable o un parámetro definido con un tipo de matriz sólo se puede utilizar en sentencias de SQL compuesto (compilado)

Ejemplos

Ejemplo 1: Cree un tipo de matriz denominado PHONENUMBERS con un máximo de 50 elementos que tengan el tipo de datos DECIMAL(10, 0).

```
CREATE TYPE
PHONENUMBERS AS DECIMAL(10,0)
ARRAY[50]
```

Ejemplo 2: Cree un tipo de matriz denominado NUMBERS con valor por omisión número de elementos en el esquema GENERIC.

```
CREATE TYPE GENERIC.NUMBERS AS DECFLOAT(34)
ARRAY[]
```

Ejemplo 3: Crear una matriz asociativa denominada PERSONAL_PHONENUMBERS con elementos que sean DECIMAL(16, 0) indexados por series como 'Casa', 'Trabajo' o 'Mamá'.

```
CREATE TYPE PERSONALPHONENUMBERS AS DECIMAL(16, 0) ARRAY[VARCHAR(8)]
```

CREATE TYPE (matriz)

Ejemplo 4: Cree un tipo de matriz asociativa donde los índices son nombres de provincias, territorios o países y los elementos son capitales:

```
CREATE TYPE CAPITALSARRAY AS VARCHAR(30) ARRAY[VARCHAR(20)]
```

Ejemplo 5: Cree un tipo de matriz asociativa para descripciones de producto de un máximo de 40 caracteres de longitud, donde los índices son números de producto de un máximo de 12 caracteres de longitud:

```
CREATE TYPE PRODUCTS AS VARCHAR(40) ARRAY[VARCHAR(12)]
```

CREATE TYPE (cursor)

La sentencia CREATE TYPE (cursor) define un tipo de cursor definido por el usuario.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

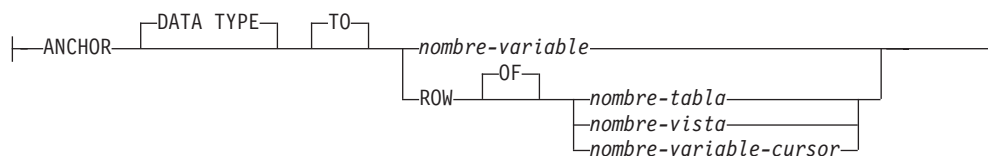
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo de cursor no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo de cursor hace referencia a un esquema existente
- Autorización DBADM

Sintaxis



tipo-datos-fila-anclados:



Descripción

OR REPLACE

Especifica que ha de sustituirse la definición del tipo de datos si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que las funciones y métodos se invalidan en lugar de descartarse cuando éstos tienen parámetros o un valor de retorno definido con el tipo de datos que se sustituye. La definición existente no debe ser de un tipo estructurado (SQLSTATE 42809). Esta opción se pasa por alto si no existe una definición para el tipo de datos en el servidor actual.

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado o definido por el usuario) que ya exista en el servidor actual. El nombre sin calificar no debe ser el mismo que un tipo de datos incorporado o BOOLEAN, BINARY o VARBINARY (SQLSTATE 42918).

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). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación. Si se especifica un nombre de tipo compuesto de dos partes, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos-fila-anclados

Identifica la información de fila de otro objeto que se utiliza para determinar el tipo de fila asociado con el tipo de cursor. El tipo de datos del objeto de anclaje está sujeto a las mismas limitaciones que se aplican a la creación de un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable a la que se hace referencia debe ser un tipo de fila y se utiliza como el tipo de fila asociado al tipo de cursor.

ROW OF *nombre-tabla* o *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. Los tipos de datos de las columnas de objeto de anclaje tienen las mismas limitaciones que se aplican a los tipos de datos de campo. El tipo de fila asociado con el tipo de cursor es un tipo de fila sin nombre.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los objetos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula **CONSTANT** especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable de cursor no es de un tipo firme que utiliza un tipo de fila con nombre, el tipo de fila asociado con el tipo de cursor es un tipo de fila sin nombre.

nombre-tipo-fila

Especifica el tipo de fila que se utilizará para comprobar el tipo de fila de la tabla de resultados de la *sentencia-select* asignada a una variable del tipo de cursor. La asignación falla si la comprobación de tipo falla (SQLSTATE 42821). Si se especifica el *nombre-tipo-fila* sin un nombre de esquema, el tipo de fila se resuelve buscando en los esquemas de la vía de acceso de SQL.

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un

CREATE TYPE (cursor)

cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.

Notas

- *Uso del tipo de cursor:* el tipo de cursor solo puede utilizarse como el tipo de datos de:
 - Una variable local en una sentencia de SQL compuesto (compilado)
 - Un parámetro de una rutina de SQL
 - El tipo de retorno de una función de SQL
 - Una variable global
- Una variable o un parámetro definido con un tipo de cursor sólo se puede utilizar en sentencias de SQL compuesto (compilado)
- Una variable o un parámetro que tiene un tipo de cursor con tipo firme no se debe utilizar para asignar valores de cursor basados en un *nombre-sentencia*, en lugar de en una *sentencia-select*
- Un tipo de cursor definido por el usuario con un tipo de fila asociado es un tipo de cursor con tipo firme; de lo contrario, es un tipo de cursor con tipo no firme.

Ejemplos

- *Ejemplo 1:* Crear un tipo de cursor que pueda utilizarse con cualquier cursor.
CREATE TYPE EMPCURSOR AS CURSOR
- *Ejemplo 2:* Crear un tipo de cursor con tipo firme que esté basado en el tipo de datos de fila DEPTROW:
CREATE TYPE DEPTCURSOR AS DEPTROW CURSOR

CREATE TYPE (diferenciado)

La sentencia CREATE TYPE (diferenciado) define un tipo diferenciado. El tipo diferenciado siempre deriva de uno de los tipos de datos incorporados y puede definirse para utilizar normas de tipificación firme o tipificación no firme..

La ejecución satisfactoria de la sentencia que define un tipo diferenciado con tipo firme también genera funciones para la conversión entre el tipo diferenciado y su tipo fuente y, opcionalmente, genera soporte para utilizar los operadores de comparación (=, <>, <, <=, > y >=) con el tipo diferenciado. La ejecución satisfactoria de la sentencia que define un tipo diferenciado con tipo no firme no genera ninguna función.

Invocación

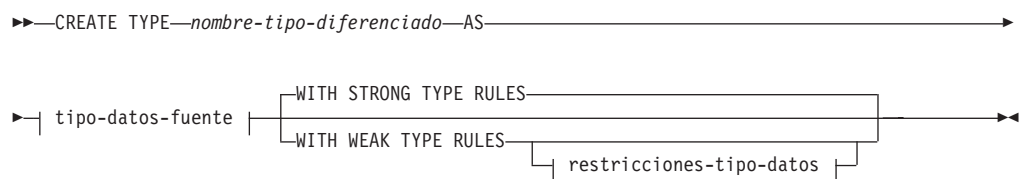
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- 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
- Autorización DBADM

Sintaxis

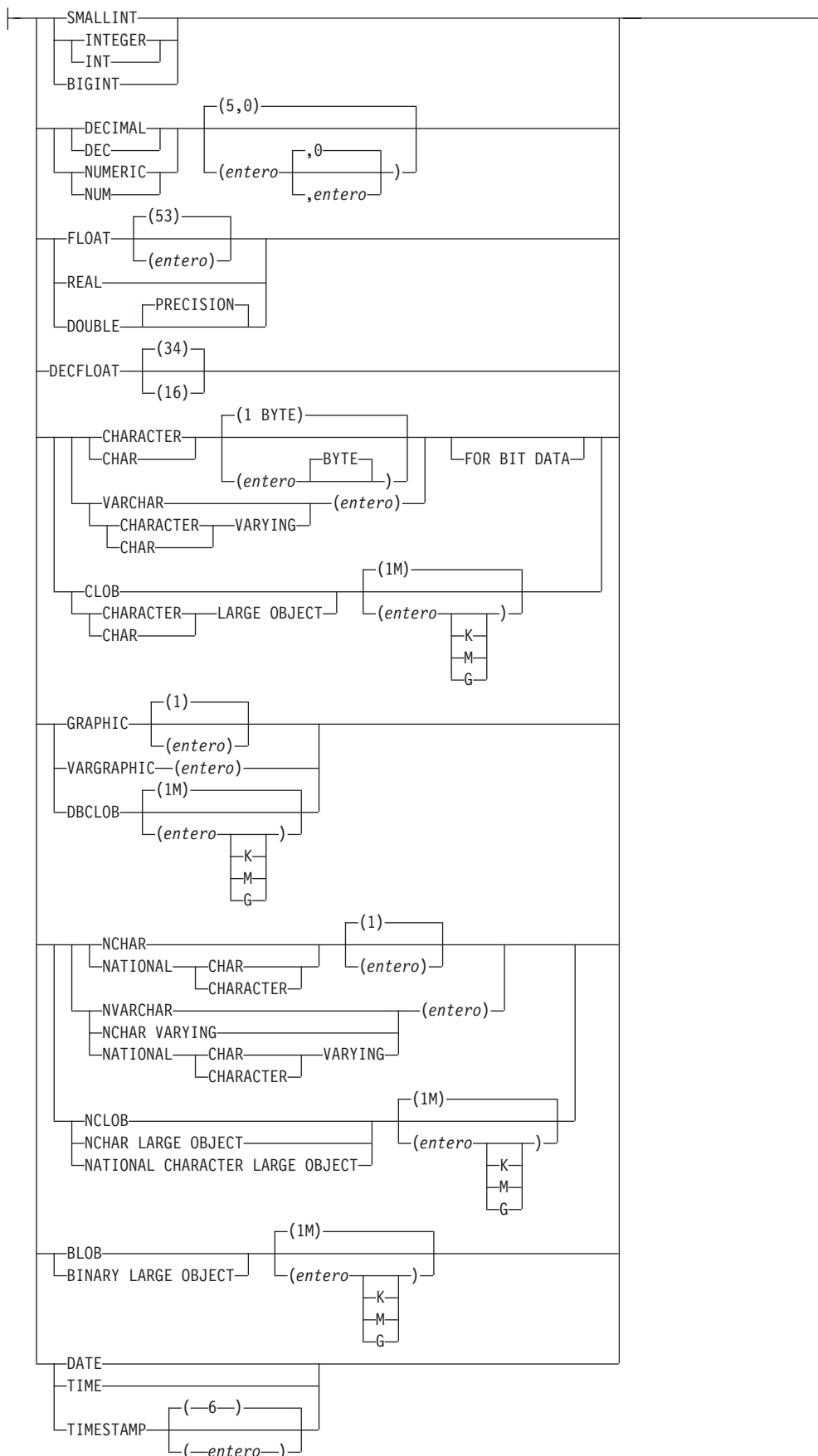


tipo-datos-fuente:



tipo-incorporado:

CREATE TYPE (diferenciado)



tipo-datos-anclados:

```

|—ANCHOR—|—DATA TYPE—|—TO—|—nombre-variable—|
|—nombre-tabla.nombre-columna—|

```

Restricciones-tipo-datos:

```

|—NOT NULL—|—CHECK—(—nombre-comprobación—)|

```

Descripción*nombre-tipo-diferenciado*

Indica el nombre del tipo diferenciado. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado o definido por el usuario) que ya exista en el servidor actual. El nombre no calificado no debe ser el mismo que el nombre de un tipo de datos incorporado ni BOOLEAN, BINARY o VARBINARY (SQLSTATE 42918). El nombre no calificado tampoco debe ser ARRAY, INTERVAL ni ROWID.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación 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 de SQL.

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no se pueden utilizar como *nombre-tipo-diferenciado* (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.

Si se especifica un *nombre-tipo-diferenciado* de dos partes, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos-fuente

Especifica el tipo de datos utilizado como base para la representación interna del tipo diferenciado. El tipo de datos debe ser un tipo de datos incorporado. Para obtener más información sobre los tipos de datos incorporados, consulte la sección "CREATE TABLE". El tipo de datos fuente no puede ser de tipo XML o un tipo ARRAY (SQLSTATE 42601). Para la portabilidad de las aplicaciones entre plataformas, utilice los siguientes nombres de tipos de datos recomendados:

- DOUBLE o REAL en lugar de FLOAT
- DECIMAL en lugar de NUMERIC
- VARCHAR, BLOB o CLOB en lugar de LONG VARCHAR
- VARGRAPHIC o DBCLOB en lugar de LONG VARGRAPHIC

tipo-datos-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos. El tipo de datos del objeto de anclaje se ve afectado por las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

CREATE TYPE (diferenciado)

nombre-variable

Identifica una variable global con un tipo de datos que es un tipo interno distinto de ROW o CURSOR. El tipo de datos de la variable global se utiliza como tipo de datos fuente para el tipo diferenciado.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente con un tipo de datos que debe especificarse como tipo-incorporado. El tipo de datos de la columna se utiliza como tipo de datos fuente para el tipo diferenciado.

WITH STRONG TYPE RULES

Especifica que las normas de tipificación firme se utilizan para operaciones donde este tipo de datos es un operando, incluidas las asignaciones y comparaciones. Es el valor por omisión.

WITH WEAK TYPE RULES

Especifica que las normas de tipificación no firme se utilizan para operaciones donde este tipo de datos es un operando, incluidas las asignaciones, las comparaciones y la resolución de función. Cuando se utilizan valores de un tipo diferenciado con tipo no firme, el tipo de datos se trata efectivamente como el *tipo-datos-fuente* especificado al procesarse la operación.

Restricciones-tipo-datos

Define restricciones para el tipo diferenciado que se aplican al asignar valores al tipo diferenciado o al realizar la conversión en el tipo diferenciado.

NOT NULL

Impide que un valor con este tipo diferenciado tenga un valor nulo. Si no se especifica NOT NULL, un valor con este tipo diferenciado puede tener el valor nulo.

CHECK (condición-comprobación)

Define una restricción de comprobación de tipo de datos. En cualquier momento, la condición de comprobación deberá ser verdadera o desconocida para cada valor que tenga este tipo de datos. La *condición-comprobación* es una forma de la *condición-búsqueda* que se ajusta a las normas de restricciones de comprobación de tabla (SQLSTATE 42621), con la adición de que la palabra clave VALUE se utiliza para hacer referencia a un valor que se asigna o convierte en el tipo diferenciado, de la misma forma que se hace referencia a un nombre de columna en una restricción de comprobación de tabla. Tenga en cuenta que la *condición-comprobación* no puede hacer referencia a variables globales.

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.

Notas

- **Privilegios:** el definidor del tipo definido por el usuario siempre recibe el privilegio WITH GRANT OPTION de EXECUTE en todas las funciones generadas automáticamente por el tipo diferenciado.

El privilegio EXECUTE sobre todas las funciones generadas automáticamente durante la sentencia CREATE TYPE (diferenciado) se otorga a PUBLIC.

- 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre el esquema se otorga a PUBLIC.
- *Funciones generadas adicionales:* cuando se crea un tipo diferenciado con tipo firme, se generan las siguientes funciones para realizar la conversión del tipo fuente:
 - Una función para convertir del tipo diferenciado al tipo de fuente
 - Una función para convertir del tipo de fuente al tipo diferenciado
 - Una función para convertir de INTEGER al tipo diferenciado si el tipo de fuente es SMALLINT
 - Una función para convertir de VARCHAR al tipo diferenciado si el tipo de fuente es CHAR
 - Una función para convertir de VARGRAPHIC al tipo diferenciado si el tipo de 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 de fuente es un tipo con parámetros, la función para convertir el tipo diferenciado al tipo de fuente tendrá como nombre de función el nombre del tipo de fuente sin los parámetros (consulte los detalles en la Tabla 28). El tipo del valor de retorno de esta función incluirá los parámetros dados en la sentencia CREATE TYPE (diferenciado). La función para convertir del tipo de fuente al tipo diferenciado tendrá un parámetro de entrada cuyo tipo es el tipo de fuente incluyendo sus parámetros. Por ejemplo,

```
CREATE TYPE T_SHOESIZE AS CHAR(2)
WITH COMPARISONS
```

```
CREATE 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 la conversión del tipo diferenciado al tipo de fuente y del tipo de fuente al tipo diferenciado para todos los tipos de datos predefinidos.

Tabla 28. Funciones CAST en tipos diferenciados

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
SMALLINT	<i>nombre-tipo-diferenciado</i>	SMALLINT	<i>nombre-tipo-diferenciado</i>
SMALLINT	<i>nombre-tipo-diferenciado</i>	INTEGER	<i>nombre-tipo-diferenciado</i>

CREATE TYPE (diferenciado)

Tabla 28. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
SMALLINT	SMALLINT	<i>nombre-tipo-diferenciado</i>	SMALLINT
INTEGER	<i>nombre-tipo-diferenciado</i>	INTEGER	<i>nombre-tipo-diferenciado</i>
INTEGER	INTEGER	<i>nombre-tipo-diferenciado</i>	INTEGER
BIGINT	<i>nombre-tipo-diferenciado</i>	BIGINT	<i>nombre-tipo-diferenciado</i>
BIGINT	BIGINT	<i>nombre-tipo-diferenciado</i>	BIGINT
DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)	<i>nombre-tipo-diferenciado</i>
DECIMAL	DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)
NUMERIC	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)	<i>nombre-tipo-diferenciado</i>
NUMERIC	DECIMAL	<i>nombre-tipo-diferenciado</i>	DECIMAL (p,s)
REAL	<i>nombre-tipo-diferenciado</i>	REAL	<i>nombre-tipo-diferenciado</i>
REAL	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
REAL	REAL	<i>nombre-tipo-diferenciado</i>	REAL
FLOAT(n) donde $n \leq 24$	<i>nombre-tipo-diferenciado</i>	REAL	<i>nombre-tipo-diferenciado</i>
FLOAT(n) donde $n \leq 24$	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
FLOAT(n) donde $n \leq 24$	REAL	<i>nombre-tipo-diferenciado</i>	REAL
FLOAT(n) donde $n > 24$	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
FLOAT(n) donde $n > 24$	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
FLOAT	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
FLOAT	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
DOUBLE	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DOUBLE PRECISION	<i>nombre-tipo-diferenciado</i>	DOUBLE	<i>nombre-tipo-diferenciado</i>
DOUBLE PRECISION	DOUBLE	<i>nombre-tipo-diferenciado</i>	DOUBLE
DECFLOAT	<i>nombre-tipo-diferenciado</i>	DECFLOAT(n)	<i>nombre-tipo-diferenciado</i>
DECFLOAT	DECFLOAT	<i>nombre-tipo-diferenciado</i>	DECFLOAT(n)
CHAR	<i>nombre-tipo-diferenciado</i>	CHAR (n)	<i>nombre-tipo-diferenciado</i>
CHAR	CHAR	<i>nombre-tipo-diferenciado</i>	CHAR (n)
CHAR	<i>nombre-tipo-diferenciado</i>	VARCHAR (n)	<i>nombre-tipo-diferenciado</i>
VARCHAR	<i>nombre-tipo-diferenciado</i>	VARCHAR (n)	<i>nombre-tipo-diferenciado</i>
VARCHAR	VARCHAR	<i>nombre-tipo-diferenciado</i>	VARCHAR (n)
CLOB	<i>nombre-tipo-diferenciado</i>	CLOB (n)	<i>nombre-tipo-diferenciado</i>
CLOB	CLOB	<i>nombre-tipo-diferenciado</i>	CLOB (n)
GRAPHIC	<i>nombre-tipo-diferenciado</i>	GRAPHIC (n)	<i>nombre-tipo-diferenciado</i>
GRAPHIC	GRAPHIC	<i>nombre-tipo-diferenciado</i>	GRAPHIC (n)
GRAPHIC	<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	VARGRAPHIC	<i>nombre-tipo-diferenciado</i>	VARGRAPHIC (n)
DBCLOB	<i>nombre-tipo-diferenciado</i>	DBCLOB (n)	<i>nombre-tipo-diferenciado</i>
DBCLOB	DBCLOB	<i>nombre-tipo-diferenciado</i>	DBCLOB (n)
BLOB	<i>nombre-tipo-diferenciado</i>	BLOB (n)	<i>nombre-tipo-diferenciado</i>

Tabla 28. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
BLOB	BLOB	<i>nombre-tipo-diferenciado</i>	BLOB (<i>n</i>)
DATE	<i>nombre-tipo-diferenciado</i>	DATE	<i>nombre-tipo-diferenciado</i>
DATE	DATE	<i>nombre-tipo-diferenciado</i>	DATE
TIME	<i>nombre-tipo-diferenciado</i>	TIME	<i>nombre-tipo-diferenciado</i>
TIME	TIME	<i>nombre-tipo-diferenciado</i>	TIME
TIMESTAMP	<i>nombre-tipo-diferenciado</i>	TIMESTAMP(<i>p</i>)	<i>nombre-tipo-diferenciado</i>
TIMESTAMP	TIMESTAMP	<i>nombre-tipo-diferenciado</i>	TIMESTAMP(<i>p</i>)

Nota: NUMERIC y FLOAT no son aconsejables para crear un tipo de datos definido por el usuario para una aplicación portátil. Deben utilizarse DECIMAL y DOUBLE en su lugar.

Las funciones que se describen en la tabla anterior y las funciones de operador de comparación son las únicas funciones que se generan automáticamente al definirse los tipos diferenciados. En consecuencia, ninguna de las funciones incorporadas (AVG, MAX, LENGTH, etcétera) recibe soporte para los tipos diferenciados con tipo firme hasta que se utiliza la sentencia CREATE FUNCTION para registrar las funciones definidas por el usuario para el tipo diferenciado con tipo firme, y esas funciones definidas por el usuario deben derivar de 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 con tipo firme, se crean operadores de comparación generados por el sistema cuando el tipo fuente da soporte a las comparaciones. La creación de esos operadores de comparación generará entradas en la vista de catálogo SYSCAT.ROUTINES de las nuevas funciones.

El nombre de esquema del tipo diferenciado debe estar incluido en la vía de acceso de SQL o en la opción FUNCPATH BIND para que la utilización de estos operadores y de las funciones de conversión en las sentencias de SQL sea satisfactoria.

- Cuando se crea un tipo diferenciado con tipo no firme, no es necesario que se generen o creen funciones adicionales porque las normas de tipificación no firme permiten utilizar un tipo diferenciado con tipo no firme en el mismo contexto en el que puede utilizarse el tipo fuente.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - CREATE DISTINCT TYPE puede especificarse en vez de CREATE TYPE
 - Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC y las funciones de conversión están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro. La cláusula WITH COMPARISONS sigue sin dar soporte a los tipos de datos LONG VARCHAR y LONG VARGRAPHIC.
 - La cláusula WITH COMPARISONS, que especifica que han de crearse operadores de comparación generados por el sistema para comparar dos instancias del tipo diferenciado, puede especificarse como la última cláusula de la sentencia si no se ha especificado WITH WEAK TYPE RULES. Utilice WITH COMPARISONS solo si es necesario para mantener la compatibilidad con versiones anteriores de los productos de la familia de DB2. Si el tipo de datos fuente es BLOB, CLOB o DBCLOB y se especifica WITH COMPARISONS, se genera un aviso como en los releases anteriores.

CREATE TYPE (diferenciado)

- ALLOW NULL, o solo NULL, pueden especificarse como valor opuesto a NOT NULL. Esta es la característica de capacidad de nulos por omisión del tipo diferenciado si no se han especificado la cláusula ALLOW NULL ni la cláusula NOT NULL. La especificación de ALLOW NULL no se considera para definir una restricción de tipo de datos para el tipo diferenciado.

Ejemplos

- *Ejemplo 1:* Crear un tipo diferenciado con tipo firme denominado SHOESIZE basado en un tipo de datos INTEGER.

```
CREATE TYPE SHOESIZE AS INTEGER
```

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:* Crear un tipo diferenciado con tipo firme denominado MILES basado en un tipo de datos DOUBLE.

```
CREATE TYPE MILES AS DOUBLE
```

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.

- *Ejemplo 3:* Crear un tipo diferenciado con tipo no firme denominado BONUS basado en un tipo de datos INTEGER y que represente un porcentaje que no puede exceder de 100.

```
CREATE TYPE BONUS AS INTEGER WITH WEAK TYPE RULES  
CHECK(VALUE >= 0 AND VALUE <= 100)
```

Puesto que se define con normas de tipificación no firme, no se generan funciones de comparación y de conversión para el tipo diferenciado con tipo no firme denominado BONUS.

- *Ejemplo 4:* Crear un tipo diferenciado con tipo no firme denominado SALARY basado en un tipo de datos DOUBLE que no pueda ser NULL y donde el rango superior esté limitado a menos de cien mil.

```
CREATE TYPE SALARY AS DOUBLE WITH WEAK TYPE RULES  
NOT NULL CHECK(VALUE < 100000)
```


CREATE TYPE (fila)

La sentencia CREATE TYPE (fila) define un tipo de fila. Un tipo de fila incluye uno o más campos con tipos de datos asociados que conforman una fila de datos.

Invocación

Esta sentencia puede incorporarse 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 que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo de fila no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo de fila hace referencia a un esquema existente
- Autorización DBADM

Sintaxis

```

▶▶ CREATE [OR REPLACE] TYPE nombre-tipo AS ROW

```

```

(
  definición-campo
  tipo-datos-fila-anclados
)

```

definición-campo:

```

nombre-campo tipo-datos

```

tipo-datos:

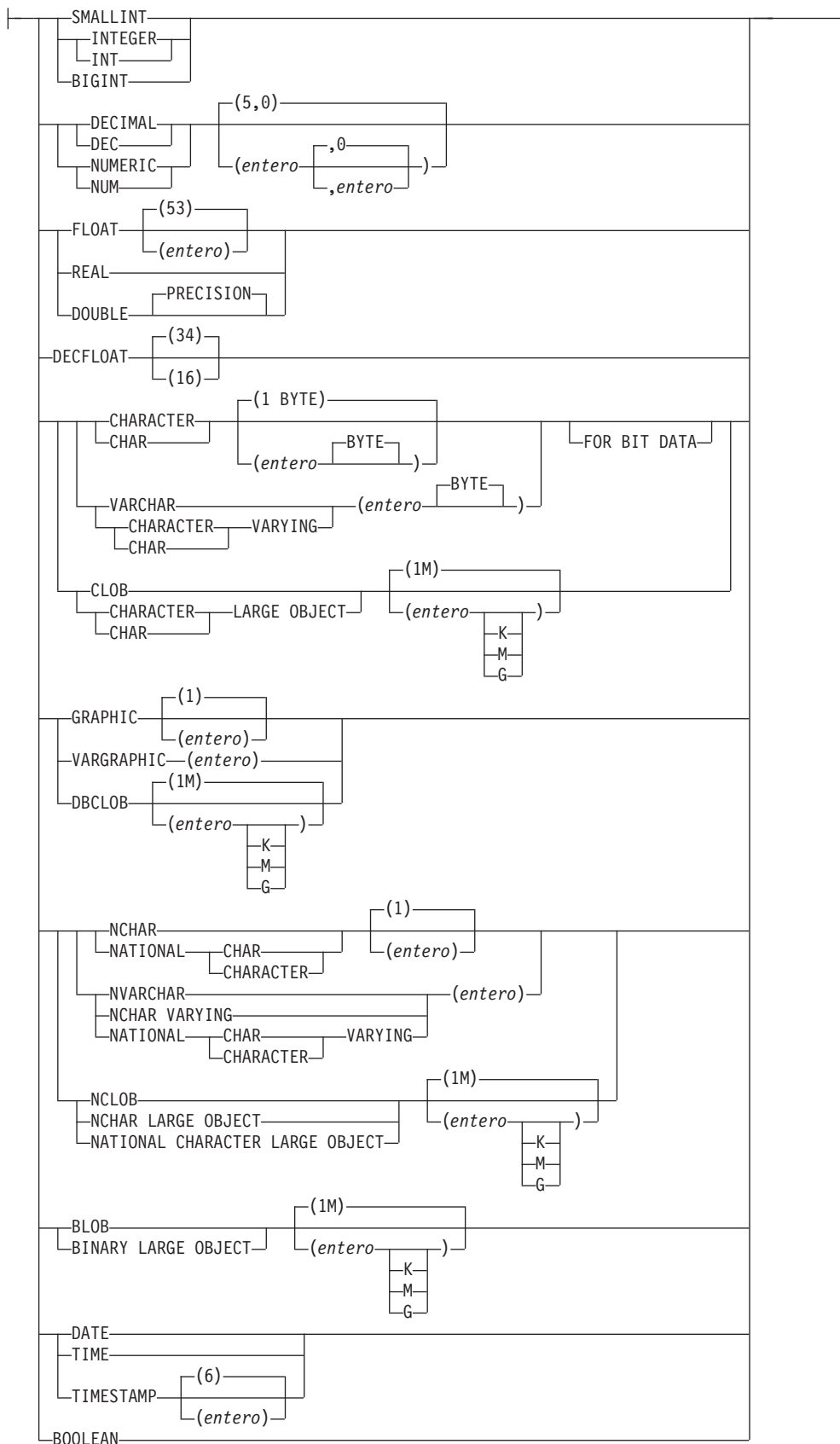
```

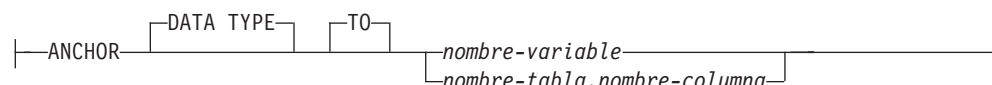
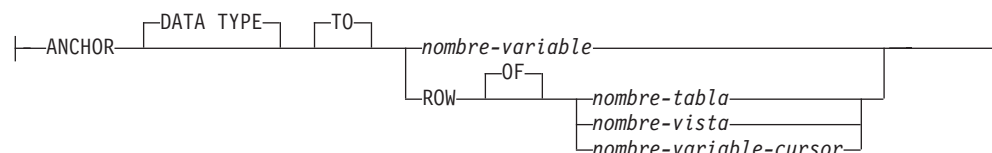
tipo-incorporado
tipo-datos-no-fila-anclados
tipo-datos-fila-anclados
nombre-tipo-fila
nombre-tipo-matriz
nombre-tipo-diferenciado

```

tipo-incorporado:

CREATE TYPE (fila)



tipo-datos-no-fila-anclados:**tipo-datos-fila-anclados:****Descripción****OR REPLACE**

Especifica que ha de sustituirse la definición del tipo de datos si existe uno en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que las funciones y métodos se invalidan en lugar de descartarse cuando éstos tienen parámetros o un valor de retorno definido con el tipo de datos que se sustituye. La definición existente no debe ser de un tipo estructurado (SQLSTATE 42809). Esta opción se pasa por alto si no existe una definición para el tipo de datos en el servidor actual.

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado, estructurado, de matriz, fila o diferenciado) que ya se describa en el catálogo. El nombre sin calificar no debe ser el mismo que un tipo de datos incorporado o BOOLEAN (SQLSTATE 42918).

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). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Si se especifica un *nombre-tipo* compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS'; de lo contrario, se devuelve un error (SQLSTATE 42939).

definición-campo

Define los campos del tipo de fila.

nombre-campo

Especifica el nombre de un campo dentro del tipo de fila. El nombre no puede ser el mismo que el de cualquier otro campo de este tipo de fila (SQLSTATE 42711).

tipo-datos

Especifica el tipo de datos del campo.

tipo-incorporado

Especifica un tipo de datos incorporado. Consulte en "CREATE TABLE" la descripción de los tipos de datos incorporados. Los tipos incorporados incluyen los tipos de datos descritos en "CREATE

CREATE TYPE (fila)

TABLE", que no sean los tipos de referencia, SYSPROC.DB2SECURITYLABEL, XML o los tipos definidos por el usuario (SQLSTATE 429C2).

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. Si se especifica un *nombre-tipo-fila* sin un nombre de esquema, el *nombre-tipo-fila* se resuelve buscando los esquemas en la vía de acceso de SQL. Los tipos de fila se pueden anidar como tipos de campo de un tipo de fila con un nivel máximo de anidamiento de dieciséis.

nombre-tipo-matriz

Especifica un tipo de matriz. Si se especifica un *nombre-tipo-matriz* sin un nombre de esquema, el *nombre-tipo-matriz* se resuelve buscando los esquemas en la vía de acceso de SQL. Los tipos de matriz se pueden anidar como tipos de campo de un tipo de fila con un nivel máximo de anidamiento de dieciséis.

nombre-tipo-diferenciado

Especifica el tipo de datos diferenciado definido por el usuario. El tipo diferenciado especificado no puede tener restricciones de tipo de datos (SQLSTATE 429C5).

tipo-datos-no-fila-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global con un tipo de datos que sea de un tipo de datos de campo de fila soportado. El tipo de datos de la variable global se utiliza como tipo de datos para el campo.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente con un tipo de datos incorporado o diferenciado. El tipo de datos de la columna se utiliza como tipo de datos para el campo.

tipo-datos-fila-anclados

Identifica la información de fila de otro objeto que se debe utilizar como campos de la fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable a la que se hace referencia debe ser un tipo de fila.

ROW OF *nombre-tabla* **o** *nombre-vista*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por *nombre-tabla* o la vista identificada por *nombre-vista*. Los tipos de datos de las columnas de objeto de anclaje tienen las mismas limitaciones que se aplican a los tipos de datos de campo.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los objetos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula **CONSTANT** especificando una *sentencia-select* en la que todas las columnas de resultados tienen nombre.

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.

Notas

- **Uso del tipo de fila:** el tipo de fila solo puede utilizarse como el tipo de datos de:
 - Una variable local en una sentencia de SQL compuesto (compilado)
 - Un parámetro de una rutina de SQL
 - El tipo de retorno de una función de SQL
 - El elemento de un tipo de matriz
 - Un tipo de cursor definido por el usuario
 - Una variable global
- Una variable o un parámetro definido con un tipo de fila sólo se puede utilizar en sentencias de SQL compuesto (compilado)

Ejemplo

- Crear un tipo de fila según las columnas de la tabla DEPARTMENT.

```
CREATE TYPE DEPTROW AS ROW (DEPTNO  VARCHAR(3),
                           DEPTNAME VARCHAR(29),
                           MGRNO   CHAR(6),
                           ADMRDEPT CHAR(3),
                           LOCATION CHAR(16))
```

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 incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

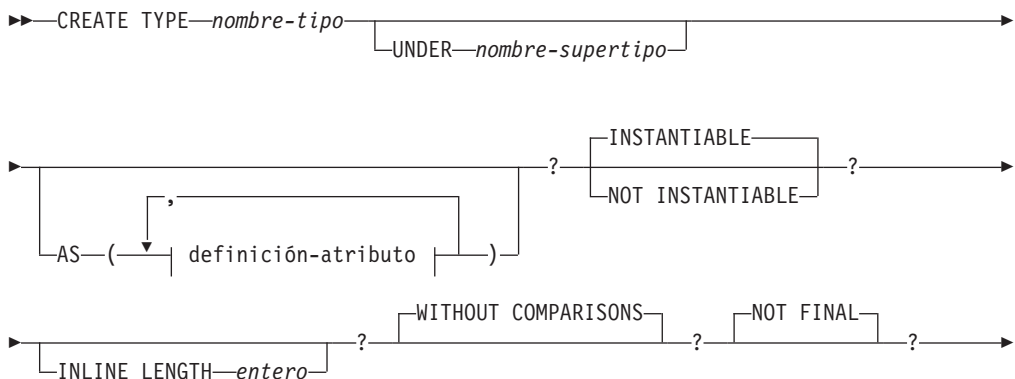
Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

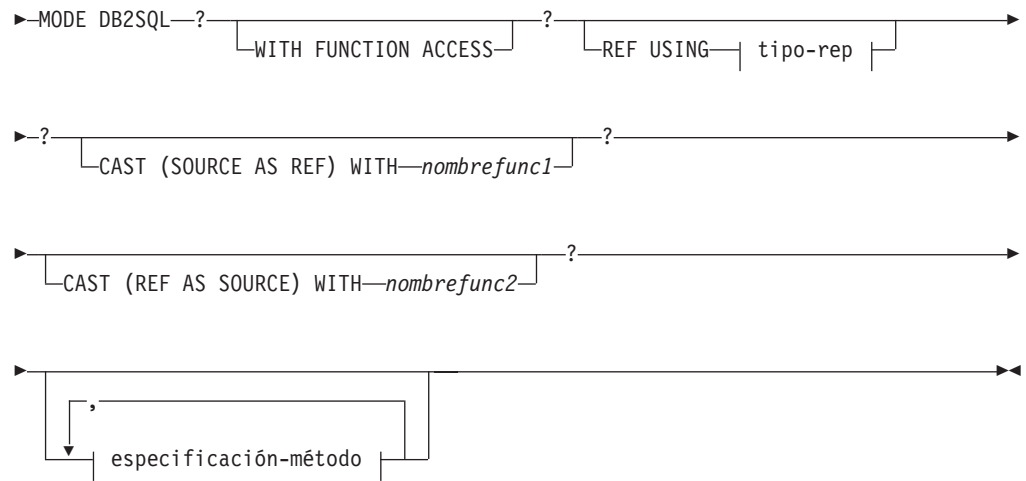
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo hace referencia a un esquema existente
- Autorización DBADM

Si se especifica UNDER y el ID de autorización de la sentencia no es el mismo que el del propietario del tipo raíz de la jerarquía de tipos, se necesita autorización DBADM.

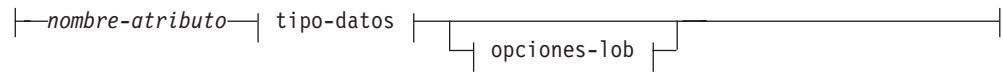
Sintaxis



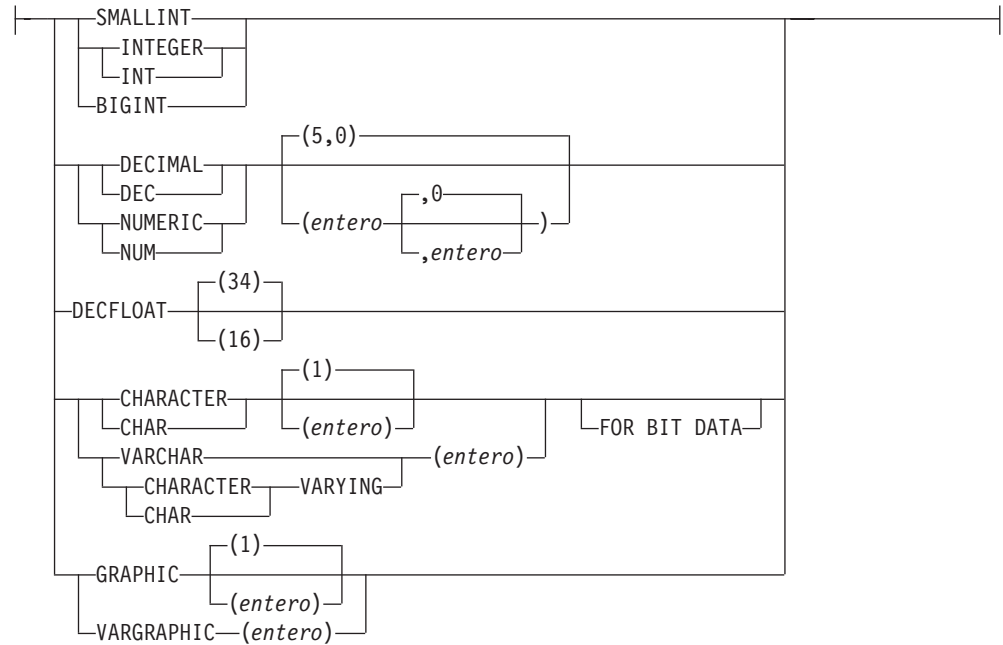
CREATE TYPE (estructurado)



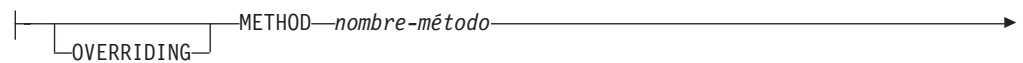
definición-atributo:



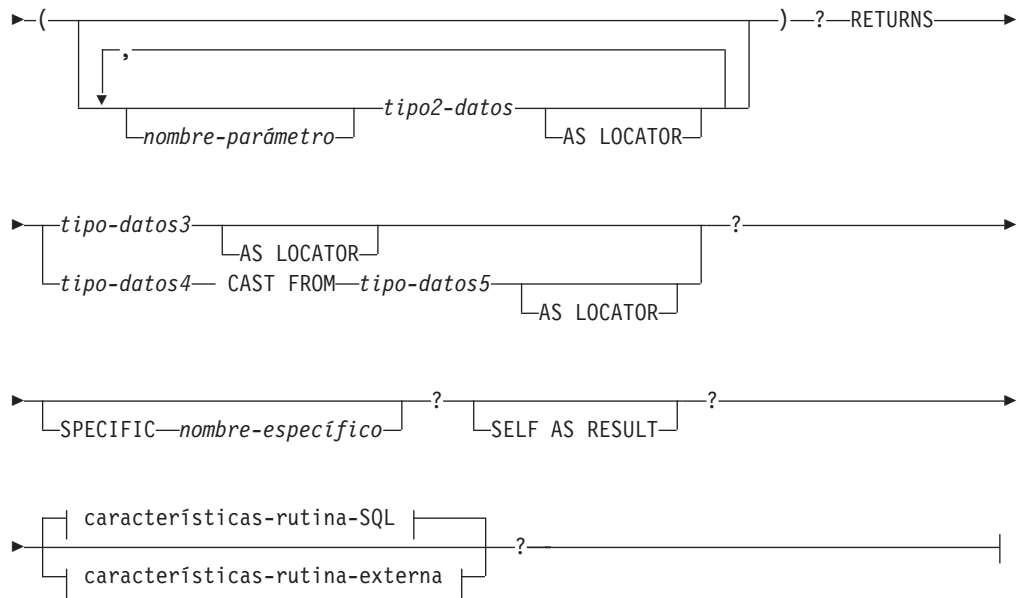
tipo-rep:



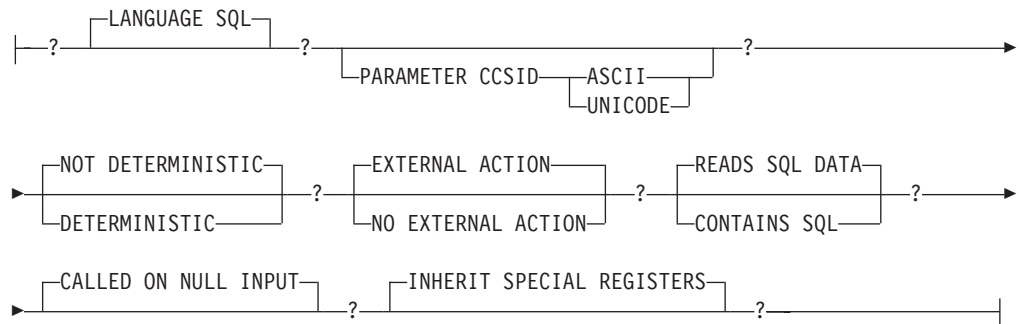
especificación-método:



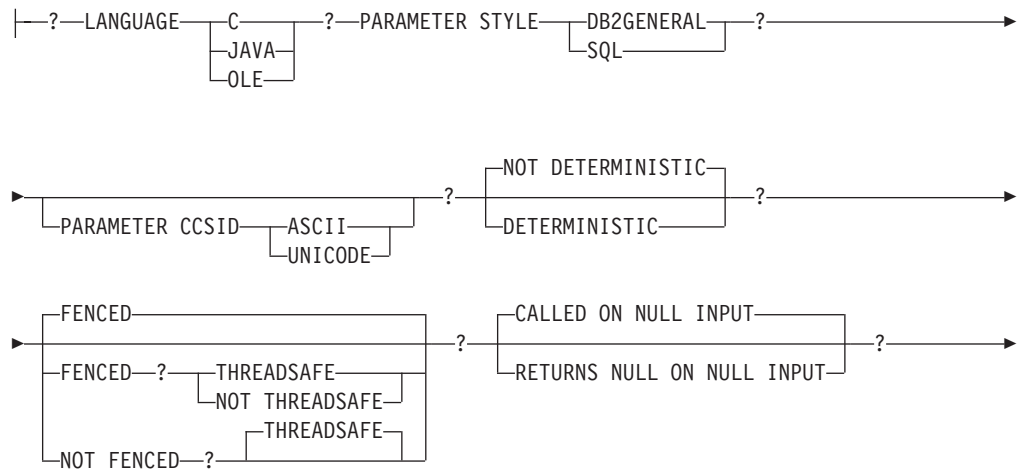
CREATE TYPE (estructurado)

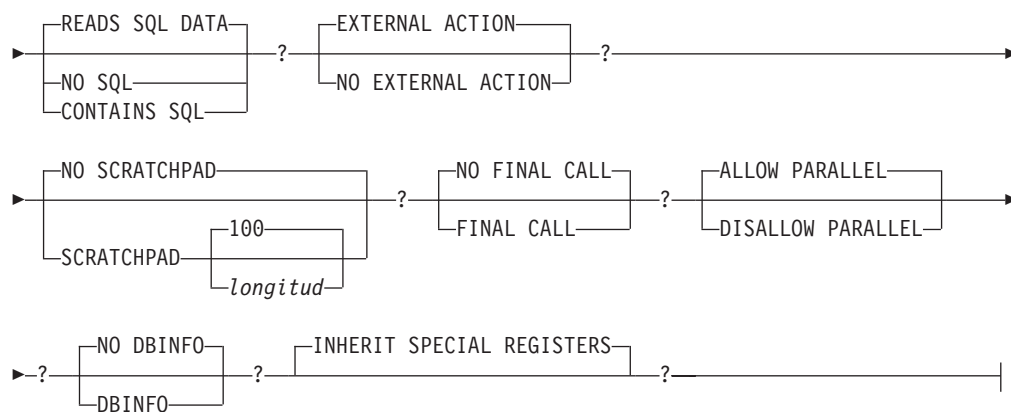


características-rutina-SQL:



características-rutina-externa:





Descripción

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado, estructurado o diferenciado) que ya exista en el servidor actual. El nombre no calificado no debe ser el mismo que el nombre de un tipo de datos incorporado, BINARY, VARBINARY o BOOLEAN (SQLSTATE 42918). El nombre no calificado tampoco debe ser ARRAY, INTERVAL ni ROWID. 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 o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados.

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). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Si se especifica un nombre compuesto de dos partes *nombre-tipo*, el nombre de esquema no puede empezar por los caracteres 'SYS' (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 que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como un *nombre-atributo* (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.

CREATE TYPE (estructurado)

tipo-datos

El tipo de datos del atributo. Es uno de los tipos de datos que se listan en "CREATE TABLE", que no son XML o un tipo diferenciado con tipo no firme (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. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". 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).

Para evitar las definiciones de tipo que, durante la ejecución, permitirían que una instancia del tipo contuviera directa o indirectamente otra instancia del mismo tipo o uno de sus subtipos, un tipo no puede definirse de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP).

opciones-lob

Especifica las opciones asociadas a tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de *opciones-lob*, consulte "CREATE TABLE".

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 alterar temporalmente 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 las fórmulas mostradas 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:

$$\text{espacio_para_atributos_no_cortos} = \text{SUM}(\text{longitudatributo} + n)$$

n se define como:

- 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 29.

2. Calcula la longitud total "inline" por omisión, mediante esta fórmula:

$$\text{longitud_por_omisión}(\text{tipo_estructurado}) = (\text{número_de_atributos} * 10) + 32 + \text{espacio_para_atributos_no_cortos}$$

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 29. Número de bytes para los tipos de datos de atributos

Tipo de datos de atributo	Número de bytes
DECIMAL	La parte integral de $(p/2)+1$, donde p es la precisión
DECFLOAT(n)	Si n es 16, el número de bytes es 8; si n es 34, el número de bytes es 16
CHAR (n)	n
VARCHAR (n)	n
GRAPHIC (n)	$n * 2$
VARGRAPHIC (n)	$n * 2$
TIMESTAMP	10
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 en función de la longitud máxima definida para el atributo LOB (consulte la Tabla 30 en la página 902).
Tipo diferenciado	Longitud del tipo de fuente del tipo diferenciado
Tipo de referencia	Longitud del tipo de datos incorporado en la que se basa el tipo de referencia.
Tipo estructurado	<code>inline_length(tipo_atributo)</code>

CREATE TYPE (estructurado)

Tabla 30. Tamaño del descriptor LOB como una función de la longitud máxima LOB

Longitud máxima de LOB	Tamaño del descriptor de LOB
1024	68
8192	92
65.536	116
524.000	140
4.190.000	164
134.000.000	196
536.000.000	220
1.070.000.000	252
1.470.000.000	276
2.147.483.647	312

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 incorporado utilizados como representación (tipo 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 REAL, FLOAT, DECFLOAT, BLOB, CLOB, DBCLOB, tipo matriz o tipo estructurado y debe tener una longitud igual o inferior a 32 672 bytes (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 debe especificarse 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)* (SQLSTATE 42710).

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).

OVERRIDING

Especifica que el método que está definiéndose altera temporalmente a un método de un supertipo del tipo que está definiéndose. La alteración temporal permite volver a implementar métodos en los subtipos y, por lo tanto, ofrece funciones más específicas. La alteración temporal no recibe soporte para los siguientes tipos de métodos:

- Métodos de tabla y fila
- Métodos externos declarado con PARAMETER STYLE JAVA
- Métodos que puede utilizarse como predicados en una extensión de índice
- Métodos de mutación o de observador generados por el sistema

Si se intenta alterar temporalmente uno de estos métodos, se generará un error (SQLSTATE 42745).

Si un método es un método de alteración temporal válido, deberá existir un método original para uno de los supertipos adecuados del tipo que está definiéndose y deberán existir las relaciones siguientes entre el método de alteración temporal y el método original:

- El nombre de método del método que está definiéndose y el método original son equivalentes.
- El método que está definiéndose y el método original tienen el mismo número de parámetros.
- El tipo de datos de cada parámetro del método que está definiéndose y el tipo de datos de los parámetros correspondientes del método original son idénticos. Este requisito excluye al parámetro SELF implícito.

Si no existe un método original con estas características, se devolverá un error (SQLSTATE 428FV).

El método de alteración temporal hereda los siguientes atributos del método original:

- Lenguaje
- Indicación de determinismo
- Indicación de acción externa
- Una indicación que especifique si este método debe o no llamarse si alguno de sus argumentos es el valor nulo
- Conversión del resultado (si se ha especificado en el método original)
- Indicación SELF AS RESULT
- La indicación de acceso a los datos de SQL o CONSTAINS SQL
- Para los métodos externos:
 - Estilo del parámetro

CREATE TYPE (estructurado)

- Indicación del localizador de los parámetros y del resultado (si se ha especificado en el método original)
- Indicación FENCED, SCRATCHPAD, FINAL CALL, ALLOW PARALLEL y DBINFO
- Indicación INHERIT SPECIAL REGISTER y THREADSAFE

nombre-método

Designa el método que se está definiendo. Debe ser un identificador de SQL no calificado (SQLSTATE 42601). El nombre de método está calificado implícitamente por el esquema utilizado para CREATE TYPE.

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como un *nombre-método* (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.

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). Si el método que está declarándose altera temporalmente a otro método, el nombre del parámetro deberá ser exactamente el mismo nombre que el del correspondiente parámetro del método alterado temporalmente; de lo contrario, se devolverá un error (SQLSTATE 428FV).

tipo2-datos

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).

Puede especificar abreviaturas y tipos de datos SQL que se pueden especificar como un tipo de columna en la sentencia CREATE TABLE y que tienen equivalentes en el lenguaje en el que se está utilizando para escribir el método. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos del lenguaje principal, consulte el tema que pertenece a su lenguaje de la lista de temas relacionados que aparece a continuación.

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 sistema 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 sistema principal.

DECIMAL (o NUMERIC) y coma flotante decimal no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).

No se pueden utilizar tipos de datos XML (SQLSTATE 42815).

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 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.

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).

Si el método que está declarándose altera temporalmente a otro método, la indicación AS LOCATOR del parámetro deberá coincidir exactamente con la indicación AS LOCATOR del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

Si el método que está declarándose altera temporalmente a otro método, la indicación FOR BIT DATA de cada parámetro deberá coincidir exactamente con la indicación FOR BIT DATA del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

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, especificadas en la descripción de 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).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si el método altera temporalmente a otro método, *tipo-datos3* debe ser un subtipo del tipo de datos del resultado del método alterado temporalmente si este tipo de datos es un tipo estructurado; de lo contrario, ambos tipos de datos deben ser idénticos (SQLSTATE 428FV).

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*

CREATE TYPE (estructurado)

debe ser convertible al parámetro *tipo-datos4*. Si no es convertible, se devuelve un error (SQLSTATE 42880).

Puesto que la longitud, la precisión o la escala de *tipo-datos4* puede inferirse de *tipo-datos5*, no es necesario (pero está permitido) especificar la longitud, la precisión o la escala de los tipos con parámetros especificados para *tipo-datos4*. En cambio, se pueden utilizar paréntesis vacíos como, por ejemplo, VARCHAR(). FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE).

Un tipo distinto no es válido como el tipo especificado en *tipo-datos5* (SQLSTATE 42815). XML no es válido como el tipo especificado en *tipo-datos4* o *tipo-datos5* (SQLSTATE 42815).

La operación de conversión también está sujeta a comprobaciones durante la ejecución, que pueden dar como resultado 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).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse la cláusula FOR BIT DATA (SQLSTATE 428FV).

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 descartar el método. No puede utilizarse nunca para invocar el método. La forma no calificada de *nombre-específico* es un identificador de SQL (cuya longitud máxima es 18). La forma calificada es un nombre de esquema seguido de un punto y un identificador de SQL. El nombre (incluido el calificador implícito o explícito) no debe designar 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: SQLaammddhhmmssxxx.

SELF AS RESULT

Identifica este método como un método de conservación de tipo, que se define de la forma siguiente:

- El tipo de retorno declarado debe ser el mismo que el tipo sujeto declarado (SQLSTATE 428EQ).
- 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.
- El método debe implementarse de forma que el tipo dinámico del resultado sea el mismo que el tipo dinámico del argumento sujeto (SQLSTATE 2200G), y el resultado no puede ser NULL (SQLSTATE 22004).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

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.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al método de SQL y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

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. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. 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).

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. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

CREATE TYPE (estructurado)

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 existen valores de argumento nulos.

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del método heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para un método que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno en el que se ha abierto el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

características-rutina-externa

LANGUAGE

Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje para la que 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 llamará al método definido por el usuario en 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 OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

Sólo se da soporte a LANGUAGE OLE para métodos definidos por el usuario almacenados en Windows de 32 bits. THREADSAFE no puede especificarse para los métodos que se han definido con LANGUAGE OLE (SQLSTATE 42613).

PARAMETER STYLE

Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros al método y obtener el resultado del método.

DB2GENERAL

Se utiliza para especificar las convenciones para pasar parámetros desde métodos externos y devolver el valor desde estos, que están definidos 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.

SQL

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 métodos expuestos por objetos de automatización OLE. Esta opción debe especificarse cuando se utiliza LANGUAGE C o LANGUAGE OLE.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al método externo y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

Esta cláusula no se puede especificar con LANGUAGE OLE (SQLSTATE 42613).

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. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de

CREATE TYPE (estructurado)

que las entradas idénticas siempre producen los mismos resultados. Un ejemplo de un tipo que no es determinante es aquel que hace referencia a registros especiales, variables globales o funciones no determinantes de un modo que afecta al tipo de resultado.

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 ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios) 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.

PRECAUCIÓN:

La utilización de NOT FENCED para métodos no probados debidamente puede comprometer la integridad de una base de datos DB2. Las bases de datos DB2 toman algunas precauciones ante varios de los tipos comunes de anomalías inadvertidas que puedan producirse, pero no pueden garantizar la integridad completa si se utilizan métodos NOT FENCED definidos por el usuario.

Para un método con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si el método es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar un método como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

THREADSAFE o NOT THREADSAFE

Especifica si se considera que el método es "seguro" para ejecutarse en el mismo proceso que otras rutinas (THREADSAFE) o no (NOT THREADSAFE).

Si el método se ha definido con un LANGUAGE distinto de OLE:

- Si el método se ha definido como THREADSAFE, el gestor de bases de datos puede invocar el método en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, un método no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Los métodos FENCED y NOT FENCED pueden ser THREADSAFE.
- Si el método se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará el método en el mismo proceso que otra rutina.

Para los métodos FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si el método se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para los métodos NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

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 existen 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.

NO SQL, CONTAINS SQL, READS SQL DATA

Indica si el método debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que el método no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL

Indica que el método puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ningún método devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA

Indica que en el método pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ningún método devuelven un error distinto (SQLSTATE 38003 ó 42985).

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.

CREATE TYPE (estructurado)

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 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 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 que se ha mostrado anteriormente. Si el método se ejecuta en varias particiones de base de datos, se asignará un área reutilizable en cada partición de base de datos en la que se procese el método, para cada referencia al método en la sentencia de SQL. De forma 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 llamada son:

- Llamada normal: Se pasan los 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 las siguientes circunstancias:
 - Fin de sentencia: Esta situación se produce cuando se cierra el cursor, en el caso de sentencias orientadas a cursor, 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 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 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.
- ¿Existe alguna acción externa que realice el método que sólo se deba llevar a cabo en una partición de base de datos? 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 todos los métodos externos debe encontrarse en un directorio disponible en todas las particiones de base de datos.

CREATE TYPE (estructurado)

El diagrama de sintaxis indica que el valor por omisión es ALLOW PARALLEL. Sin embargo, el valor por omisión es DISALLOW PARALLEL si en la sentencia se especifican una o más de las opciones siguientes:

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 al método como un 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). Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si se especifica DBINFO, al método se le envía una estructura que contiene la siguiente información:

- 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 las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - si, y sólo 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, contiene el nombre no calificado de la tabla que se desea actualizar o insertar; de otro modo, 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.
- 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.
- Números de columna del resultado del método de tabla - no es aplicable a métodos.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales del método heredarán sus valores iniciales de la sentencia que realiza la llamada. Para los cursores, los valores iniciales se heredan del valor de tiempo que corresponde a la apertura del cursor.

Al proceso que realiza la llamada del método no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Algunos registros especiales, como los registros especiales de hora y fecha, reflejan una propiedad de la sentencia que se está ejecutando y, por consiguiente, nunca se heredan de quien llama.

Notas

- La creación de un tipo estructurado 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre 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).
- Ningún método original puede alterar temporalmente a otro método, o ser alterado temporalmente por un método original (SQLSTATE 42745). Además, en una relación de alteración temporal no puede haber una función y un método. Esto significa que si se ha considerado que la función es un método, siendo su primer parámetro el sujeto S, no deberá alterar temporalmente a otro método de ningún supertipo de S y no podrá alterarlo temporalmente otro método de ningún subtipo de S (SQLSTATE 42745).
- 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 altera temporalmente la signatura de una función existente en este esquema, la sentencia falla (SQLSTATE 42710). Las funciones o métodos generados no pueden descartarse 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 han definido para que tengan el mismo comportamiento que los operadores de comparación para el tipo de datos subyacente de REF(*nombre-tipo*). (Todas las referencias de una jerarquía de tipo tienen el mismo tipo de representación de referencia. Esto permite que REF(S) y REF(T) puedan compararse, siempre que S y T tengan un supertipo común. Debido a que la exclusividad de la columna de OID sólo se aplica 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 distintas.)

CREATE TYPE (estructurado)

El ámbito del tipo de referencia no se toma en consideración 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:

```
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 estar incluido en la vía de acceso de SQL para que la utilización de estos operadores y de las funciones de conversión en las sentencias de SQL sea satisfactoria.

- Función constructora

La función constructora se genera para permitir la creación de una nueva instancia del tipo. Esta nueva instancia tendrá un 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.

- 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 asunto es nulo, el método observador devuelve un valor nulo del tipo de atributo.

CREATE TYPE (estructurado)

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 de método del método observador generado es como si la sentencia siguiente se hubiera ejecutado:

```
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 asunto modificada mediante 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 términos generales, puesto que es el conjunto de funciones de transformación que se asocia a un tipo determinado (o jerarquía de tipo) que permite a UDF y a la aplicación cliente acceder al tipo estructurado, las funciones de transformación deben escribirse de modo que se dé soporte a *todos* los atributos de una jerarquía compuesta determinada (es decir, incluyendo el cierre transitivo de todos los subtipos y sus tipo estructurados anidados).

CREATE TYPE (estructurado)

Cuando se crea un nuevo subtipo de un tipo existente, se invalidan todos los paquetes que dependen de los métodos que se han definido en los supertipos del tipo que está creándose y que pueden elegirse para la alteración temporal.

- **Restricciones de acceso a las tablas:** si un método se ha definido como READS SQL DATA, ninguna sentencia del método podrá acceder a una tabla que la sentencia que ha invocado el método está modificando (SQLSTATE 57053). Por ejemplo, supongamos que el método BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE DEPTINFO SET SALARY = SALARY + EMP.BONUS(), no podrá leerse ninguna sentencia de SQL del método BONUS desde la tabla EMPLOYEE.
- **Privilegios:** el definidor del tipo definido por el usuario siempre recibe el privilegio WITH GRANT OPTION de EXECUTE en todos los métodos y funciones generados automáticamente por el tipo estructurado. El privilegio EXECUTE no se otorgará para ninguno de los métodos explícitamente especificados en la sentencia CREATE TYPE hasta que se haya definido un cuerpo de método mediante la utilización de la sentencia CREATE METHOD. El usuario que define el tipo definido por el usuario no dispone del derecho para descartar la especificación de método mediante la utilización de la sentencia ALTER TYPE. El privilegio EXECUTE sobre todos los métodos y funciones generados automáticamente durante la sentencia CREATE TYPE (estructurado) se otorga a PUBLIC.

Cuando en una sentencia de SQL se utiliza un método externo, el usuario que define el método debe disponer de privilegio EXECUTE para cualquiera de los paquetes que el método utiliza.

- En un entorno de base de datos particionada, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- **Métodos EXTERNAL ACTION:** Si se invoca un método EXTERNAL ACTION en otra lista que no sea la lista de selección más externa, los resultados son imprevisibles, ya que el número de veces que se invoca el método variará en función del plan de acceso utilizado.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
 - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
 - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
 - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
 - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL

Se acepta la sintaxis siguiente como comportamiento por omisión para los métodos externos:

- ASUTIME NO LIMIT
- NO COLLID
- PROGRAM TYPE SUB
- STAY RESIDENT NO

- CCSID UNICODE en una base de datos Unicode
- CCSID ASCII en una base de datos que no es Unicode si no se especifica
PARAMETER CCSID UNICODE

Se acepta la sintaxis siguiente como comportamiento por omisión para los métodos de SQL:

- CCSID UNICODE en una base de datos Unicode
- CCSID ASCII en una base de datos que no es Unicode

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)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  NO EXTERNAL ACTION,

  METHOD DISTANCE (address_t)
  RETURNS FLOAT
  LANGUAGE C
  DETERMINISTIC
  PARAMETER STYLE SQL
  NO SQL
  NO EXTERNAL ACTION
```

```
CREATE TYPE germany_addr_t UNDER address_t AS
  (FAMILY_NAME VARCHAR(30))
  NOT FINAL
  MODE DB2SQL
```

CREATE TYPE (estructurado)

```
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

La sentencia CREATE TYPE MAPPING define una correlación entre tipos de datos.

La correlación puede definirse entre los tipos de datos siguientes:

- El tipo de datos de una columna de una tabla o una vista de fuente de datos que se va a definir para una base de datos federada
- Un tipo de datos correspondiente que ya está definido para la base de datos federada

La correlación puede asociar el tipo de datos de base de datos federada con un tipo de datos de:

- Una fuente de datos especificada
- Un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular

Una correlación de tipo de datos sólo debe crearse si una correlación existente no es adecuada.

Si se pueden aplicar varias correlaciones de tipos al crear un apodo o una tabla (DDL transparente), se aplicará la más reciente.

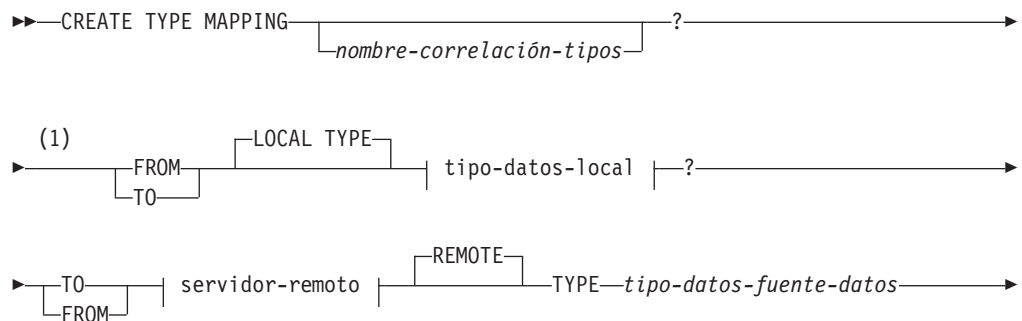
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

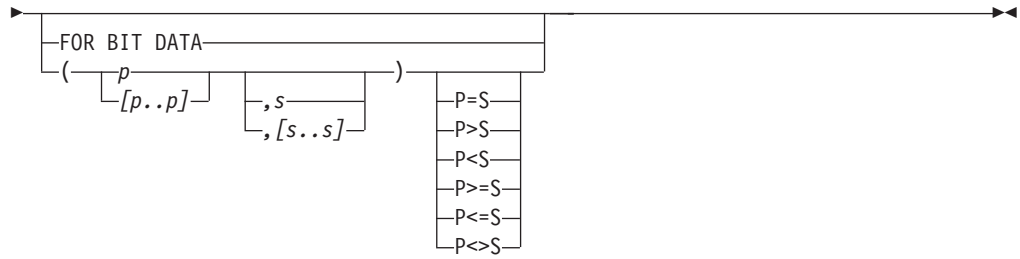
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

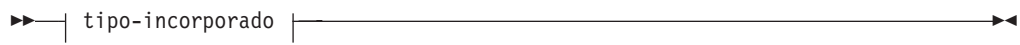
Sintaxis



CREATE TYPE MAPPING

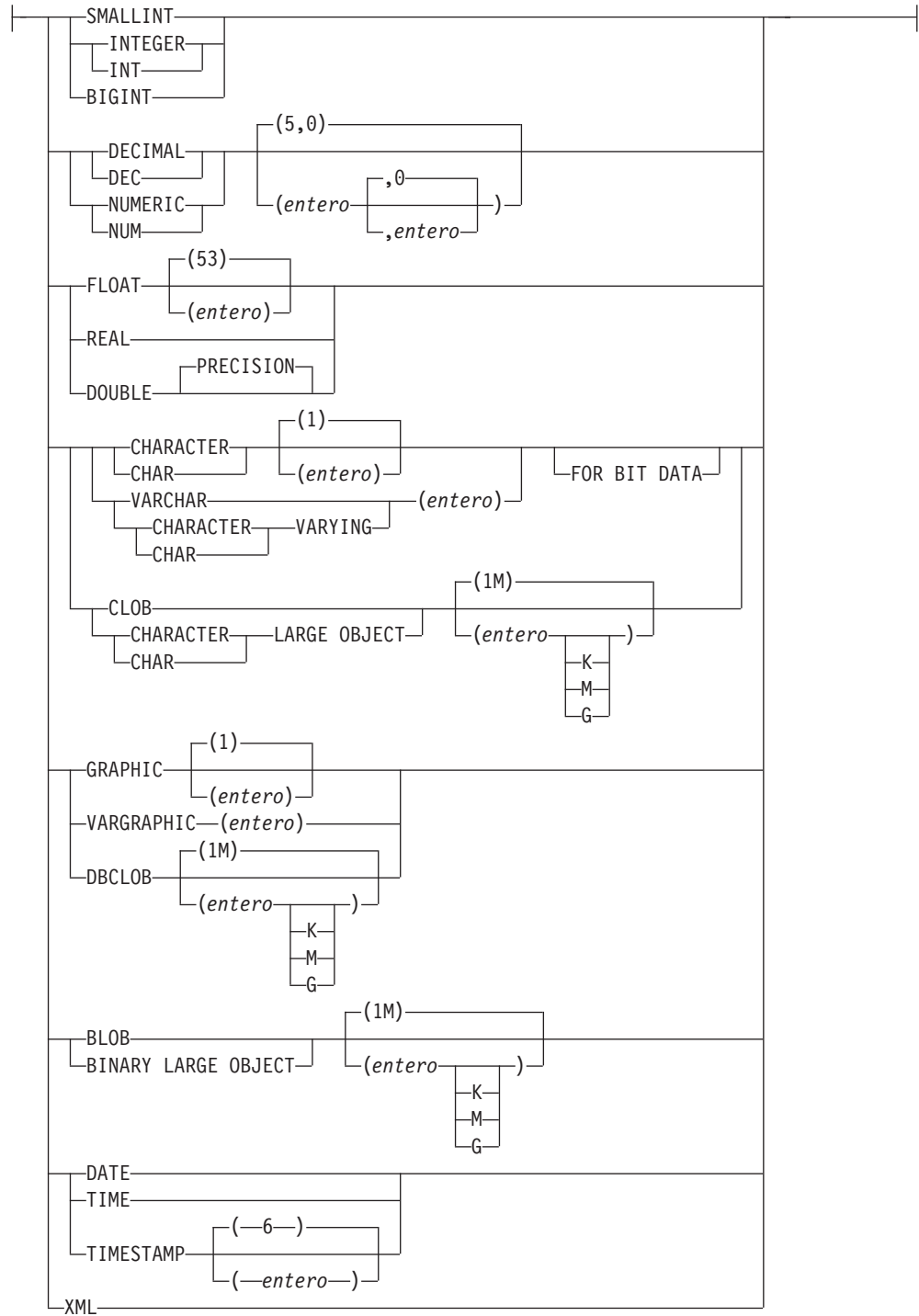


tipo-datos-locales:

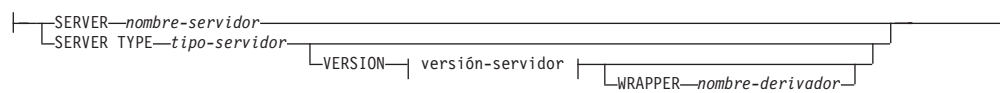


tipo-incorporado:

CREATE TYPE MAPPING

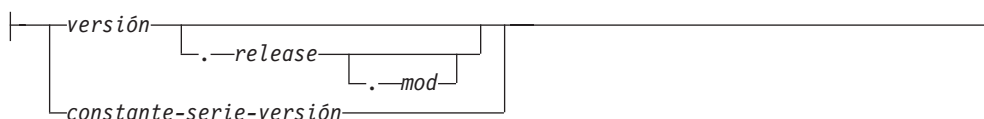


servidor-remoto:



CREATE TYPE MAPPING

versión-servidor:



Notas:

- 1 En la sentencia CREATE TYPE MAPPING, deberán existir una palabra clave TO y una palabra clave FROM.

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*.

FROM o TO

Especifica una correlación de tipos invertida o en avance.

FROM

Especifica una correlación de tipos en avance cuando va seguida del *tipo-datos-local* o una correlación de tipos invertida cuando va seguida del *servidor-remoto*.

TO Especifica una correlación de datos en avance cuando va seguida del *servidor-remoto* o una correlación de tipo de datos invertida cuando va seguida del *tipo-datos-local*.

tipo-datos-locales

Identifica un tipo de datos que se define en una base de datos federada. Si se especifica el *tipo-datos-local* sin un nombre de esquema, el nombre de tipo se resuelve buscando los esquemas en la vía de acceso de SQL.

Se pueden utilizar paréntesis vacíos para los tipos de datos con parámetros. Un tipo de datos con parámetros es cualquiera de los tipos de datos que se puede definir con una longitud, una escala o una precisión específica. Si se especifican paréntesis vacíos en una correlación de tipos en avance como, por ejemplo, CHAR(), la longitud se determina a partir de la longitud de la columna de la tabla remota. Si se especifican paréntesis vacíos en una correlación de tipo invertida, la correlación de tipos se aplica al tipo de datos con cualquier longitud. Si se omiten los paréntesis, se utiliza la longitud por omisión para el tipo de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE). NUMBER() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (DECFLOAT o DECIMAL).

DECFLOAT puede aceptarse solamente como el *tipo-datos-locales* por el derivador Oracle, el derivador DB2 para IBM DB2 Versión 9.5 para Linux, UNIX y Windows o posterior.

El *tipo-datos-locales* no puede ser un tipo definido por el usuario (SQLSTATE 42611).

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 fuente de datos para el que se define el *tipo-datos-fuente-datos*.

VERSION

Identifica la versión de la fuente de datos para el que se define el *tipo-datos-fuente-datos*.

versión

Especifica el número de versión. El valor debe ser un entero.

release

Especifica el número del release de la versión indicada por *versión*. El valor debe ser un entero.

mod

Especifica el número de la modificación del release indicado por *release*. El valor 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 estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-derivador*

Especifica el nombre del derivador 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 fuente de datos que se está correlacionando con el tipo de datos local.

Se pueden utilizar paréntesis vacíos para los tipos de datos con parámetros. Si se especifican paréntesis vacíos en una correlación de tipos en avance como, por ejemplo, CHAR(), la correlación de tipos se aplica al tipo de datos con cualquier longitud. Si se especifican paréntesis vacíos en una correlación de tipo de datos invertida, la longitud se determina a partir de la longitud de columna especificada en el DDL transparente. Si se omiten los paréntesis, se utiliza la longitud por omisión para el tipo de datos.

El *tipo-datos-fuente-datos* debe ser un tipo de datos incorporado. No están permitidos los tipos definidos por el usuario.

Si se especifica el *nombre-servidor* con una correlación de tipos o los servidores existentes se ven afectados por la correlación de tipos, *tipo-datos-fuente-datos*, *p* y *s* se verifican al crear la correlación de tipos (SQLSTATE 42611).

p Si se especifica *p*, el tipo de correlación sólo afecta al tipo de datos cuya longitud o precisión sea igual a *p*.

[*p1*..*p2*]

Sólo para la correlación de tipos en avance. Para un tipo de datos decimal, *p1* y *p2* especifican el número mínimo y máximo de dígitos que un valor puede tener. Para tipos de datos de serie, *p1* y *p2* especifican el número mínimo y máximo de caracteres que un valor puede tener. En todos los casos, el máximo debe ser igual o exceder del mínimo; y ambos números deben ser válidos con respecto al tipo de datos.

s Si se especifica *s*, sólo el tipo de datos cuya escala sea igual a *s* se ve afectado por la correlación de tipos.

CREATE TYPE MAPPING

[*s1..s2*]

Sólo para la correlación de tipos en avance. Para un tipo de datos decimal, *s1* y *s2* especifican el número mínimo y máximo de dígitos permitidos a la derecha de la coma decimal. El máximo debe ser igual o superior al mínimo y ambos números deben ser válidos con respecto al tipo de datos.

P [operando] S

El tipo de datos decimal, P [*operando*] S especifica una comparación entre la precisión y el número de dígitos permitidos a la derecha de la coma decimal. Por ejemplo, el operando = indica que se aplica la correlación de tipos si la precisión y el número de dígitos permitidos en la fracción decimal son iguales.

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 para un tipo de datos de caracteres.

Notas

- Una sentencia CREATE TYPE MAPPING de una unidad de trabajo determinada (UOW) no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
 - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
 - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o 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 uno de los elementos siguientes:
 - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
 - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
 - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- Cuando se pueden aplicar varias correlaciones de tipos, se utilizará la más reciente. Puede recuperar la hora de creación para una correlación de tipos consultando la columna CREATE_TIME de la vista del catálogo SYSCAT.TYPEMAPPINGS.

Ejemplos

- *Ejemplo 1:* Cree una correlación de tipos en avance entre el tipo de datos DATE de Oracle y el tipo de datos SYSIBM.DATE. Para todos los apodos que se creen después de haber definido esta correlación, las columnas del tipo de datos DATE de Oracle se correlacionarán con columnas DB2 del tipo de datos DATE.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
FROM LOCAL TYPE SYSIBM.DATE
TO SERVER TYPE ORACLE
REMOTE TYPE DATE
```

- *Ejemplo 2:* Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.DECIMAL(10,2) y el tipo de datos NUMBER([10..38],2) en la fuente de datos ORACLE1. Si hay una columna en la tabla Oracle del tipo de datos NUMBER(11,2), se correlacionará con una columna de datos de tipo DECIMAL(10,2), porque 11 está entre 10 y 38.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  FROM LOCAL TYPE SYSIBM.DECIMAL(10,2)
  TO SERVER ORACLE1
  REMOTE TYPE NUMBER([10..38],2)
```

- *Ejemplo 3:* Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.VARCHAR(*p*) y el tipo de datos CHAR(*p*) de Oracle en la fuente de datos ORACLE1 (*p* es cualquier longitud). Si hay una columna en la tabla Oracle del tipo de datos CHAR(10), se correlacionará con una columna del tipo de datos VARCHAR(10).

```
CREATE TYPE MAPPING MY_ORACLE_CHAR
  FROM LOCAL TYPE SYSIBM.VARCHAR()
  TO SERVER ORACLE1
  REMOTE TYPE CHAR()
```

- *Ejemplo 4:* Cree una correlación de tipos invertida del tipo de datos NUMBER(10,2) de Oracle en la fuente de datos ORACLE2 y el tipo de datos SYSIBM.DECIMAL(10,2). Si utiliza un DDL transparente para crear una tabla Oracle y especifica una columna de tipo de datos DECIMAL(10,2), DB2 creará la tabla Oracle con una columna de tipo de datos NUMBER(10,2).

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  TO LOCAL TYPE SYSIBM.DECIMAL(10,2)
  FROM SERVER ORACLE2
  REMOTE TYPE NUMBER(10,2)
```

CREATE USAGE LIST

La sentencia CREATE USAGE LIST define una lista de uso. Una lista de uso es un objeto de base de datos para supervisar todas las secciones exclusivas (sentencias DML) que han hecho referencia a una tabla o índice determinados durante la ejecución.

Invocación

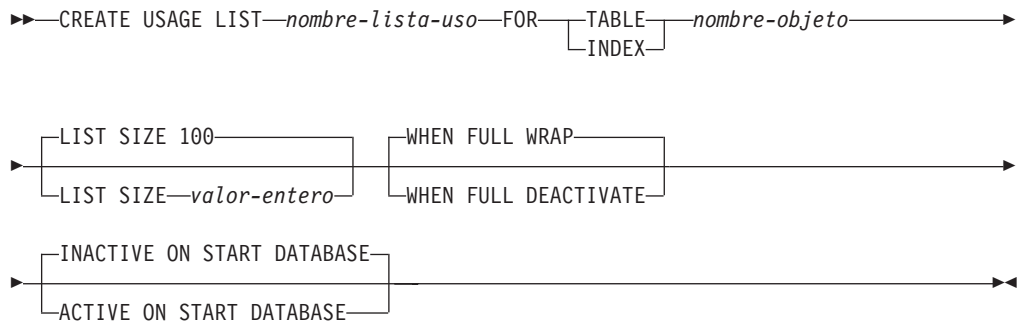
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SQLADM

Sintaxis



Descripción

nombre-lista-uso

Asigna un nombre a la lista de uso. El *nombre-lista-uso*, incluyendo el calificador implícito o explícito, no debe identificar una lista de uso descrita en el catálogo (SQLSTATE 42710). Si la lista de uso está calificada de forma explícita con un nombre de esquema, éste no debe empezar con los caracteres 'SYS' (SQLSTATE 42939).

TABLE *nombre-objeto*

Designa la tabla para la cual se ha definido la lista de uso. El *nombre-objeto*, incluyendo el calificador implícito o explícito, debe especificar una tabla definida en el catálogo (SQLSTATE 42704). El nombre no debe especificar alias, tablas de catálogo, tablas temporales creadas, tablas de jerarquía, tablas desenlazadas, apodos, tablas con tipo o vistas (SQLSTATE 42809).

INDEX *nombre-objeto*

Designa el índice para el cual se ha definido la lista de uso. El *nombre-objeto*, incluyendo el calificador implícito o explícito, debe especificar un índice definido en el catálogo (SQLSTATE 42704). Los índices definidos en tablas que no sean tablas sin tipo o tablas de consulta materializada no se soportan (SQLSTATE 42809). El nombre debe especificar un índice físico; índices de bloque (BLOK), índices de clústeres (CLUS), índices de bloque de dimensión

(DIM), índices regulares (REG) e índices físicos sobre columna XML (XVIP). El resto de tipos de índices no se soportan (SQLSTATE 42809).

LIST SIZE *valor-entero*

Especifica que el tamaño de esta lista son *valor-entero* entradas. El tamaño mínimo que se puede especificar es de 10 y el máximo de 5000 (SQLSTATE 428B7). El tamaño por omisión es de 100 entradas.

WHEN FULL

Especifica qué acción ha de realizarse cuando una lista de uso activa se llene. El valor por omisión es recortar cuando la lista está llena.

WRAP

Especifica que la lista de uso se ajusta y sustituye las entradas mas antiguas.

DEACTIVATE

Especifica que la lista de uso se desactiva.

INACTIVE ON START DATABASE

Especifica que la lista de uso no está activada para la supervisión siempre que la base de datos está activada. La colección se debe iniciar de forma explícita utilizando la sentencia SET USAGE LIST. Esta cláusula es el valor por omisión.

ACTIVE ON START DATABASE

Especifica que la lista de uso se activa automáticamente para la supervisión siempre que la base de datos está activada.

Notas

- **Secciones de rastreo con claves únicas:** una lista de uso conserva rastreos de todas las secciones exclusivas (únicamente sentencias DML) que han referenciado un objeto concreto. Las referencias se agregan a la lista con una única clave de ID ejecutable, representando la sección que hace la referencia, y el ID del intervalo del supervisor en el momento de la referencia. Cada entrada de la lista mantiene un recuento de las ejecuciones de sección relacionadas con esa entrada y un conjunto de estadísticas que describe en términos generales el efecto que ha tenido la sección en el objeto durante esas ejecuciones.
- **Hora de publicación de la lista de uso:** una lista de uso se establece como publicada cuando se confirma la sentencia CREATE USAGE LIST.
- **Asignación de memoria:** la memoria se asigna la primera vez que una sección hace referencia al objeto para el que se ha definido la lista de uso.
- **Asignación de memoria en un entorno de base de datos particionada o un entorno DB2 pureScale:** Si el estado de una lista de uso para una tabla o índice particionados se establece en activo, la memoria se asigna a cada partición de datos cuando la sección hace referencia por primera vez a la partición de datos. De forma similar, en un entorno de base de datos particionada o en un entorno DB2 pureScale, la memoria se asigna a cada miembro activo. Si un miembro no está disponible en el momento de la activación, la memoria se asigna cuando el miembro se vuelve a activar (si el estado de la lista de uso todavía está establecido en activo). Esto también se aplica cuando se añade un miembro al clúster.
- **Estado de la lista de uso al especificar WHEN FULL DEACTIVATE:** si la lista de uso se ha creado con la cláusula WHEN FULL DEACTIVATE, el estado de la lista de uso de cada miembro se establece en inactivo de forma independiente. De forma similar, para las tablas e índices particionados, el estado de la lista de uso para cada partición de datos se establece en inactivo de forma independiente.

CREATE USAGE LIST

- *Reactivación implícita de una lista de uso activa:* Si el estado de una lista de uso INACTIVE ON START DATABASE se establece en activo en un entorno de base de datos particionada o entorno DB2 pureScale, su comportamiento será similar al de la cláusula ACTIVE ON START DATABASE hasta que el estado de la lista de uso se establezca explícitamente en inactivo o hasta que se recicle la instancia. Es decir, si el estado de una lista de uso está activo cuando se desactiva un miembro de base de datos y el miembro de la base de datos se vuelve a activar posteriormente, la lista de uso para este miembro también se vuelve a activar de forma implícita.
- *Las listas de uso inactivas permanecen inactivas después de volver a activar el miembro de la base de datos:* Si el estado de una lista de uso ACTIVE ON START DATABASE se establece en inactivo en un entorno de base de datos particionada o DB2 pureScale, su comportamiento es similar al de la cláusula INACTIVE ON START DATABASE hasta que el estado de la lista de uso se establece en activo explícitamente o hasta que se recicle la instancia. Es decir, si el estado de una lista de uso está inactivo cuando un miembro de la base de datos se desactiva o está fuera de línea y un miembro de la base de datos se vuelve a activar posteriormente, el estado de la lista de uso para este miembro permanecerá inactivo.
- *Varias listas de uso:* se pueden crear varias listas de uso para la misma tabla o índice, sin embargo, se recomienda que sólo se active una. La activación de todas las listas afecta al rendimiento de la base de datos y al uso de memoria.
- *Activación y desactivación de las listas de uso:* consulte la sección Notes para consultar la sentencia SET USAGE LIST STATE referente a la activación y desactivación de la lista de uso.
- *Consideraciones sobre el tamaño de la lista de uso:* cuando el estado de una lista de uso se establece en activo, la memoria de la lista de uso se asigna desde el almacenamiento dinámico de supervisión. El valor máximo del tamaño de lista de uso es de 2 MB aproximadamente. Para las tablas e índices particionados, la memoria se asigna a cada partición de datos. Por ejemplo, si una tabla particionada tiene definidas tres particiones de datos, se asignarán aproximadamente 6 MB de memoria. Por tanto, la activación de varias listas de uso impone más requisitos de memoria en el almacenamiento dinámico de supervisión. Por tanto, se recomienda seleccionar un tamaño de lista razonable o establecer el parámetro de configuración **mon_heap_sz** en AUTOMATIC para que el gestor de bases de datos gestione el tamaño del almacenamiento dinámico de supervisión.
- *Consideraciones sobre el rendimiento:* para mantener un rendimiento alto, cree listas de uso limitándose a la cantidad necesaria para recabar la información que necesite. Cada lista de uso requiere memoria del sistema; el rendimiento del sistema puede verse afectado si se activan listas de uso adicionales.

Ejemplos

- *Ejemplo 1:* cree una lista de uso USL_ACC para la tabla SAYYID.ACCOUNTS con un tamaño de lista por omisión de 100 entradas.

```
CREATE USAGE LIST USL_ACC FOR TABLE SAYYID.ACCOUNTS
```
- *Ejemplo 2:* cree una lista de uso USL_SHOPPING_IND para el índice BIRD.SHOPPINGIND con una lista de 50 entradas que se ajuste cuando la lista se llene.

```
CREATE USAGE LIST USL_SHOPPING_IND FOR INDEX BIRD.SHOPPINGIND
LISTSIZE 50
WHEN FULL WRAP
```


- *Ejemplo 3:* cree una lista de uso USL_PAYROLL para la tabla MIKE.PAYROLL con un tamaño de lista de 200 entradas que se desactivará cuando la lista esté llena y que iniciará automáticamente la recopilación cuando se active la base de datos.

```
CREATE USAGE LIST USL_PAYROLL FOR TABLE MIKE.PAYROLL  
LISTSIZE 200  
WHEN FULL DEACTIVATE  
ACTIVE ON START DATABASE
```

- *Ejemplo 4:* cree una lista de uso USL_EMP para la tabla particionada JACOBO.EMPLOYEES con un tamaño de lista de 500 entradas que se desactivará cuando la lista esté llena.

```
CREATE USAGE LIST USL_EMP FOR TABLE JACOBO.EMPLOYEES  
LIST SIZE 500  
WHEN FULL DEACTIVATE
```

Cuando la lista de uso esté desactivada para la supervisión, se asignará una lista de 500 entradas a cada partición de datos.

- *Ejemplo 5:* cree una lista de uso USL_PARTS para la tabla SHAKTI.PARTS con un tamaño de lista de 20 entradas que se activará manualmente cuando se active la base de datos y se recortará cuando esté llena.

```
CREATE USAGE LIST USL_PARTS FOR TABLE SHAKTI.PARTS  
LIST SIZE 20  
INACTIVE ON START DATABASE  
WHEN FULL WRAP
```

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 deben utilizar en una fuente de datos especificada.

Invocación

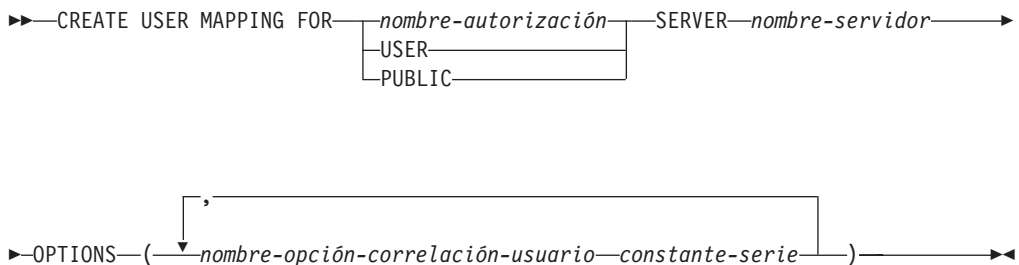
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de la autorización que se está correlacionando a la fuente de datos, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o autorizaciones.

Cuando se crea una correlación de usuarios públicos, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM.

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. El *nombre-autorización* se correlaciona con la opción de correlación de usuarios REMOTE_AUTHID.

USER

El valor del registro especial USER. Cuando se especifica USER, el ID de autorización que emite la sentencia CREATE USER MAPPING se correlaciona con la opción de correlación de usuarios REMOTE_AUTHID.

PUBLIC

Especifica que cualquier ID de autorización válido de la base de datos federada local se correlacione con el ID de autorización de la fuente de datos especificado en la opción de usuario REMOTE_AUTHID.

SERVER *nombre-servidor*

Nombra el objeto de servidor para la fuente de datos a la que el *nombre-autorización* puede acceder. El *nombre-servidor* es el nombre local para el servidor remoto que se ha registrado en la base de datos federada.

OPTIONS

Indica las opciones que están habilitadas cuando se crea la correlación de usuarios.

nombre-opción-correlación-usuario

Especifica el nombre de la opción.

constante-serie

Especifica el valor para el *nombre-opción-correlación-usuario* como una constante de serie de caracteres.

Notas

- Las correlaciones de usuarios sólo se necesitan para las siguientes fuentes de datos: la familia de productos DB2, Documentum, Informix, Microsoft SQL Server, ODBC, Oracle, Sybase y Teradata.
- Siempre se requiere la opción REMOTE_PASSWORD para una correlación de usuarios.
- No pueden coexistir en el mismo servidor federado correlaciones de usuarios públicos y correlaciones de usuarios no públicos. Esto significa que, si ha creado correlaciones de usuarios públicos, no podrá crear correlaciones de usuarios no públicos en el mismo servidor federado. Asimismo, si ha creado correlaciones de usuarios no públicos, no podrá crear correlaciones de usuarios públicos en el mismo servidor federado.
- **Alternativas de sintaxis:** se soporta la sintaxis siguiente para mantener la compatibilidad con versiones anteriores de DB2:
 - ADD se puede especificar antes de *nombre-opción-correlación-usuario* *constante-serie*.

Ejemplos

- *Ejemplo 1:* Registre una correlación de usuarios para el objeto de servidor de fuente de datos DB2 para z/OS SERVER390. Correlacione el nombre de autorización para la base de datos federada local con el ID de usuario y contraseña para SERVER390. El nombre de autorización es RSPALTEN. El ID de usuario para SERVER390 es SYSTEM. La contraseña para SERVER390 es MANAGER.

```
CREATE USER MAPPING FOR RSPALTEN
  SERVER SERVER390
  OPTIONS
  (REMOTE_AUTHID 'SYSTEM',
   REMOTE_PASSWORD 'MANAGER')
```

- *Ejemplo 2:* Registre una correlación de usuarios para el objeto de servidor de fuente de datos Oracle ORACLE1. MARCR es el nombre de autorización para la base de datos federada local y el ID de usuario para ORACLE1. Puesto que el nombre de autorización y el ID de usuario son iguales, no es necesario especificar la opción REMOTE_AUTHID en la correlación de usuarios. La contraseña para MARCR en ORACLE1 es NZXCZY.

```
CREATE USER MAPPING FOR MARCR
  SERVER ORACLE1
  OPTIONS
  (REMOTE_PASSWORD 'NZXCZY')
```

- *Ejemplo 3:* Cree un derivador de DRDA y un servidor de fuente de datos DB2 para z/OS SERVER390. A continuación, registrar una correlación de usuarios públicos con el objeto de servidor SERVER390. PUBLIC indica cualquier ID de autorización válido para la base de datos federada local. El ID de usuario para SERVER390 es APP_USER. La contraseña para SERVER390 es secret.

CREATE USER MAPPING

```
CREATE WRAPPER DRDA;  
CREATE SERVER SERVER390  
  TYPE db2/udb VERSION 9.7 WRAPPER DRDA  
  AUTHORIZATION "APP_USER" PASSWORD "secret"  
  OPTIONS (DBNAME 'remotedb');  
CREATE USER MAPPING FOR PUBLIC SERVER SERVER390  
  OPTIONS (REMOTE_AUTHID 'APP_USER', REMOTE_PASSWORD 'secret');
```

CREATE VARIABLE

La sentencia CREATE VARIABLE define una variable global de sesión.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la variable no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la variable hace referencia a un esquema existente
- Autorización DBADM

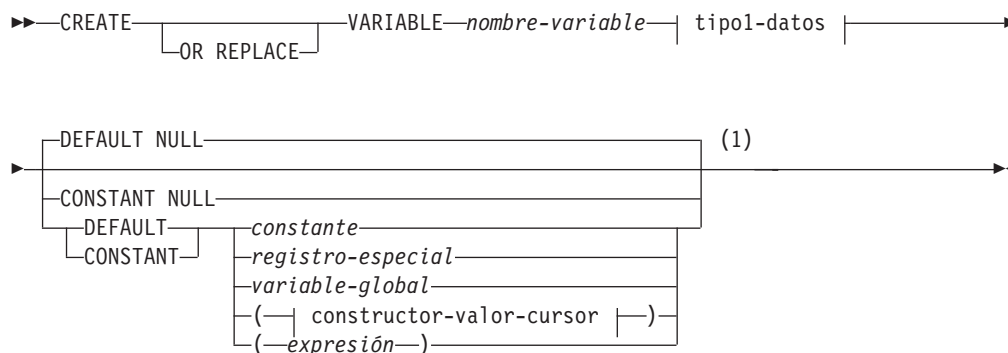
y cualesquiera privilegios que sean necesarios para ejecutar la expresión por omisión.

Para ejecutar esta sentencia con un *constructor-valor-cursor* que utiliza una *sentencia-select*, los privilegios del ID de autorización de la sentencia deben incluir los privilegios necesarios para ejecutar la *sentencia-select*. Consulte la sección Autorización en "Consultas de SQL".

Los privilegios de grupo no se tienen en cuenta a la hora de comprobar la autorización para los objetos a los que se hace referencia en la sentencia.

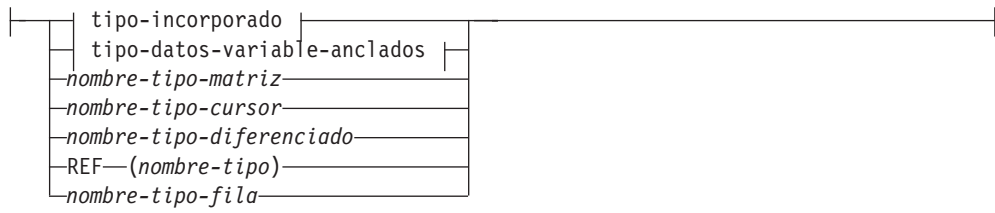
Para sustituir una variable existente, el ID de autorización de la sentencia debe ser el propietario de la variable existente (SQLSTATE 42501).

Sintaxis

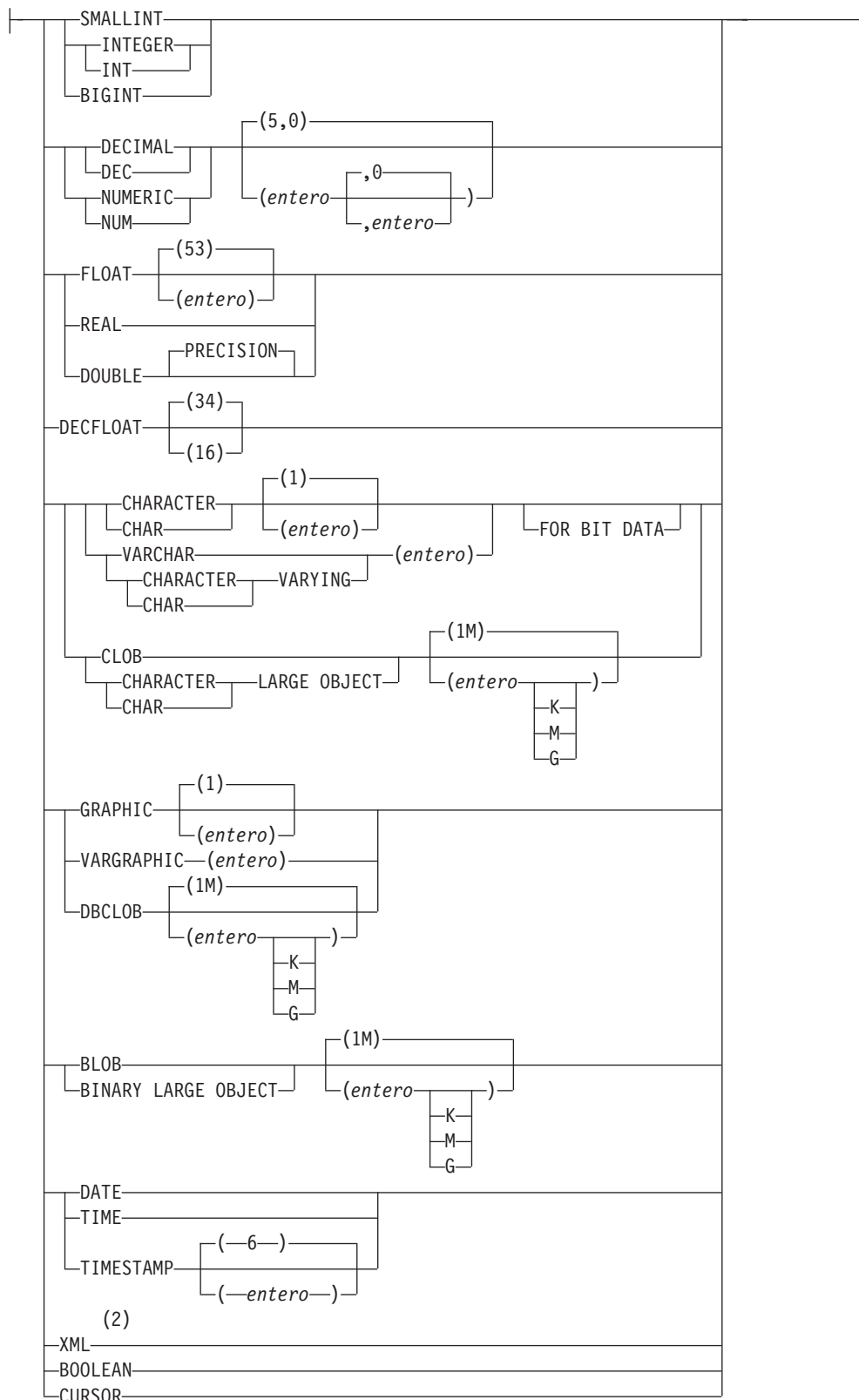


tipo-datos1:

CREATE VARIABLE

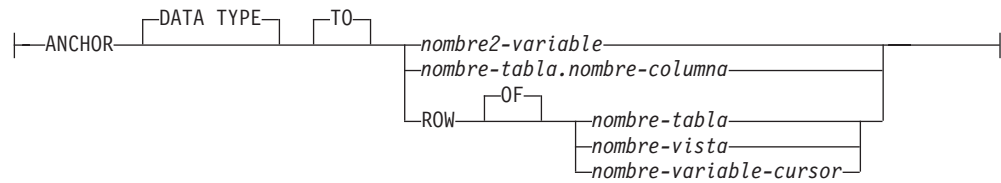


tipo-incorporado:

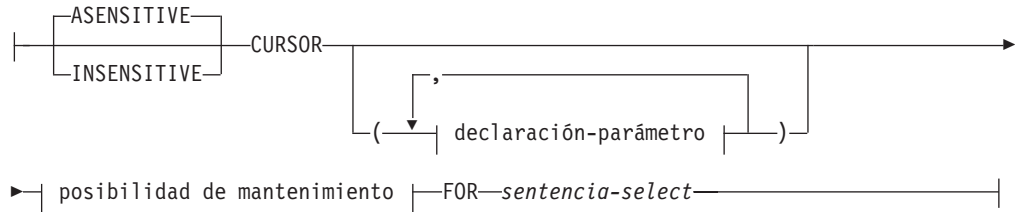


tipo-datos-variable-anclados:

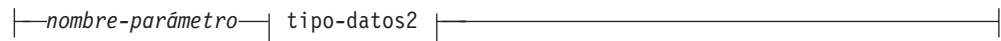
CREATE VARIABLE



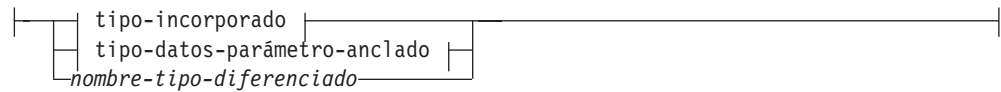
constructor-valor-cursor:



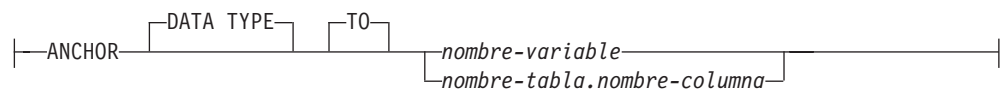
declaración-parámetro:



tipo-datos-2:



tipo-datos-parámetro-anclado:



posibilidad de mantenimiento:



Notas:

- 1 Si `tipo-datos1` especifica un tipo incorporado **CURSOR** o un `nombre-tipo-cursor`, solamente pueden especificarse **NULL** o `constructor-valor-cursor`. Únicamente se puede especificar **DEFAULT NULL** explícitamente para el `nombre-tipo-matriz` o `nombre-tipo-fila`.
- 2 Para Versión 10.1, puede utilizar el tipo de datos XML sólo como tipo de datos de parámetro en un constructor de valor de cursor. Para Versión 10.1 Fixpack 1 o releases de fixpack posteriores, también puede utilizar el tipo de datos XML para crear variables globales.

Descripción

OR REPLACE

Especifica que se debe sustituir la definición de la variable si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre la variable no se ven afectados por ello. Esta opción se ignora si no existe una definición para la variable en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

nombre-variable

Da nombre a la variable global. El nombre, incluyendo un calificador implícito o explícito, no debe identificar una variable global que ya exista en el servidor actual (SQLSTATE 42710). Si no se especifica el calificador, se asigna el esquema actual de forma implícita. Si el nombre de variable global está calificado de forma explícita con un nombre de esquema, éste no debe empezar con los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos1

Especifica el tipo de datos de la variable global. No puede especificarse un tipo estructurado (SQLSTATE 42611).

tipo-incorporado

Especifica un tipo de datos incorporado. BOOLEAN y CURSOR no pueden especificarse para una tabla. Para Versión 10.1, no puede especificarse un tipo de datos XML (SQLSTATE 42611). El soporte del tipo de datos XML empieza en Versión 10.1 Fixpack 1. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

Se puede especificar FOR BIT DATA como parte de los tipos de datos de serie de caracteres.

BOOLEAN

Para un booleano.

CURSOR

Para una referencia a un cursor subyacente.

tipo-datos-variable-anclados

Identifica otro objeto que se utiliza para determinar el tipo de datos de la variable global. El tipo de datos del objeto de anclaje tiene las mismas limitaciones que se aplican a la especificación del tipo de datos directamente o, en el caso de una fila, a la creación de un tipo de fila.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre2-variable

Identifica una variable global. El tipo de datos de la variable a la que se hace referencia se utiliza como tipo de datos para la variable global.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para la variable global.

ROW OF *nombre-tabla* **o** *nombre-vista*

Especifica que la variable global es una fila de campos con nombres y tipos de datos que se basan en los nombres de columna y los tipos de datos de columna de la tabla identificada por

CREATE VARIABLE

nombre-tabla o la vista identificada por *nombre-vista*. El tipo de datos de la variable global es un tipo de fila sin nombre.

ROW OF *nombre-variable-cursor*

Especifica una fila de campos con nombres y tipos de datos que se basan en los nombres de campo y los tipos de datos de campos de la variable de cursor identificada por *nombre-variable-cursor*. La variable de cursor especificada debe ser uno de los elementos siguientes (SQLSTATE 428HS):

- Una variable global con un tipo de datos de cursor con tipo firme.
- Una variable global con un tipo de datos de cursor con tipo no firme que se creó o declaró con una cláusula **CONSTANT** especificando una sentencia-select en la que todas las columnas de resultados tienen nombre.

Si el tipo de cursor de la variable global no es de un tipo firme que utiliza un tipo de fila con nombre, el tipo de datos de la variable global es un tipo de fila sin nombre.

nombre-tipo-matriz

Especifica el nombre de un tipo de matriz definido por el usuario. Si se especifica el *nombre-tipo-matriz* sin un nombre de esquema, el tipo de matriz se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-cursor

Especifica el nombre de un tipo de cursor. Si se especifica el *nombre-tipo-cursor* sin un nombre de esquema, el tipo de cursor se resuelve buscando en los esquemas de la vía de acceso de SQL.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. La longitud, la precisión y la escala de la variable declarada son, respectivamente, la longitud, la precisión y la escala del tipo fuente del tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

REF (*nombre-tipo*)

Especifica un tipo de referencia. Si se especifica un nombre de tipo sin un nombre de esquema, el *nombre-tipo* se resuelve buscando los esquemas en la vía de acceso de SQL.

nombre-tipo-fila

Especifica el nombre de un tipo de fila definido por el usuario. Los campos de la variable son los campos del tipo de fila. Si se especifica el *nombre-tipo-fila* sin un nombre de esquema, el tipo de fila se resuelve buscando en los esquemas de la vía de acceso de SQL.

DEFAULT o CONSTANT

Especifica un valor para la variable global cuando se hace referencia a ella por primera vez en la sesión. El valor de la cláusula **DEFAULT** o **CONSTANT** se determina en esta primera referencia. Si no se especifica ninguna, el valor por omisión para la variable global es el valor nulo. Sólo puede especificarse de forma explícita **DEFAULT NULL** si está especificado *nombre-tipo-matriz* o *nombre-tipo-fila*.

DEFAULT

Define el valor por omisión de la variable global. El valor por omisión debe ser una asignación compatible con el tipo de datos de la variable.

CONSTANT

Especifica que la variable global tiene un valor fijo que no se puede cambiar. Una variable global que se defina mediante CONSTANT no puede utilizarse como destino de una operación de asignación. El valor fijo debe ser compatible con la asignación al tipo de datos de la variable.

NULL

Especifica NULL como valor por omisión para la variable global. Si se especifica el *nombre-tipo-fila*, el valor de la variable global es una fila donde cada campo tiene el valor nulo.

constante

Especifica el valor de una constante como valor por omisión para la variable global. Si *tipo-datos1* especifica un tipo incorporado CURSOR o un *nombre-tipo-cursor*, no puede especificarse la *constante* (SQLSTATE 42601).

registro-especial

Especifica el valor de un registro especial como valor por omisión para la variable global. Si *tipo1-datos* especifica un tipo incorporado CURSOR o un *nombre-tipo-cursor*, no se puede especificar el *registro-especial* (SQLSTATE 42601).

variable-global

Especifica el valor de una variable global como valor por omisión para la variable global. Si *tipo1-datos* especifica un tipo incorporado CURSOR o un *nombre-tipo-cursor*, no se puede especificar la *variable-global* (SQLSTATE 42601).

constructor-valor-cursor

Un *constructor-valor-cursor* especifica la *sentencia-select* asociada con la variable global. La asignación de un *constructor-valor-cursor* a una variable de cursor define el cursor subyacente de esa variable de cursor.

ASENSITIVE o INSENSITIVE

Especifica si el cursor es sensible o insensible a los cambios. Consulte "DECLARE CURSOR" para obtener más información. El valor por omisión es ASENSITIVE.

ASENSITIVE

Especifica que el cursor debe ser todo lo sensible que sea posible para las operaciones de inserción, actualización o supresión realizadas en las filas subyacentes de la tabla de resultados, en función de cómo se haya optimizado *sentencia-select*. Esta opción es el valor por omisión.

INSENSITIVE

Especifica que el cursor no debe contar con sensibilidad para las operaciones de inserción, actualización o supresión que se han realizado en las filas subyacentes de la tabla de resultados. Si se especifica INSENSITIVE, el cursor es de sólo lectura y la tabla de resultados se materializa al abrirse el cursor. Como resultado de ello, el tamaño de la tabla de resultados, el orden de las filas y los valores de cada fila no cambian tras abrirse el cursor. La sentencia SELECT no puede contener una cláusula FOR UPDATE, y el cursor no puede utilizarse para las actualizaciones o supresiones posicionadas.

(declaración-parámetro, ...)

Especifica los parámetros de entrada del cursor, incluido el nombre y el tipo de datos de cada parámetro.

CREATE VARIABLE

nombre-parámetro

Asigna un nombre al parámetro que se debe utilizar como variable de SQL dentro de la *sentencia-select*. El nombre no puede ser igual que ningún otro nombre de parámetro del cursor. Los nombres deben elegirse también evitando nombres de columna que se puedan utilizar en la *sentencia-select*, ya que los nombres de columna se resuelven antes que los nombres de parámetro.

tipo2-datos

Especifica el tipo de datos del parámetro de cursor utilizado dentro de una *sentencia-select*.

tipo-incorporado

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Los tipos incorporados BOOLEAN y CURSOR no pueden especificarse (SQLSTATE 429BB).

tipo-datos-parámetro-anclado

Identifica otro objeto que se utiliza para determinar el tipo de datos del parámetro de cursor. El tipo de datos del objeto de anclaje se ve afectado por las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

ANCHOR DATA TYPE TO

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

nombre-variable

Identifica una variable global. El tipo de datos de la variable a la que se hace referencia se utiliza como tipo de datos para el parámetro de cursor.

nombre-tabla.nombre-columna

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el parámetro de cursor.

nombre-tipo-diferenciado

Especifica el nombre de un tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

posibilidad de mantenimiento

Especifica si se impedirá que el cursor se cierre como consecuencia de una operación de confirmación. Consulte "DECLARE CURSOR" para obtener más información. El valor por omisión es WITHOUT HOLD.

WITHOUT HOLD

No impide que el cursor se cierre como consecuencia de una operación de confirmación.

WITH HOLD

Mantiene recursos en varias unidades de trabajo. Impide que el cursor se cierre como consecuencia de una operación de confirmación.

sentencia-select

Especifica la sentencia SELECT del cursor. Consulte "sentencia-select" para obtener más información.

nombre-sentencia

Especifica la *sentencia-select* preparada del cursor. Consulte "PREPARE" para obtener una explicación de las sentencias preparadas. La variable del cursor de destino no debe tener un tipo de datos que sea un tipo de cursor definido por el usuario con tipo firme (SQLSTATE 428HU).

expresión

Especifica el valor de una expresión como valor por omisión para la variable global. La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". La expresión debe ser compatible con la asignación al tipo de datos de la variable. El tamaño máximo de la expresión es de 64K. La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL) ni realizar acciones externas (SQLSTATE 42845). Si *tipo1-datos* especifica un tipo incorporado CURSOR o un *nombre-tipo-cursor*, no se puede especificar la *expresión* (SQLSTATE 42601).

Normas

- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.

Notas

- Las variables globales de sesión tienen un ámbito de sesión. Esto significa que, aunque están disponibles para todas las sesiones activas en la base de datos, su valor es privado para cada una de las sesiones.
- **Contextos para variables globales de fila, cursor, booleanas y de matriz:** las variables globales que son variables de matriz, variables booleanas o variables de fila sólo se pueden utilizar en sentencias de SQL compuesto (compilado) o en sentencias de variable SET. Las variables globales que son variables de cursor solamente se pueden utilizar en sentencias de SQL compuesto (compilado).
- **Creación con errores:** si un objeto al que se hace referencia en la expresión por omisión no existe o se ha marcado como no válido o si el definidor no tiene temporalmente los privilegios para acceder al objeto, y si el parámetro de configuración de la base de datos **auto_reval** no se ha establecido en DISABLED, la variable seguirá creándose satisfactoriamente. La variable se marcará como no válida y se volverá a validar la próxima vez que se invoque.
- **Ámbito de valores de variable global:** Los valores de las variables globales se mantienen hasta que se actualizan en la sesión actual, hasta que se descarta o modifica la variable global o hasta que finaliza una sesión de la aplicación. El valor no se ve afectado por las sentencias COMMIT o ROLLBACK. El valor por omisión para una variable global puede ser no determinista y dependiente de cuándo se calcula el valor por omisión para la variable global (por ejemplo, una referencia a la hora del día o una referencia a algunos datos almacenados en una tabla).

Una técnica utilizada habitualmente, sobre todo para cuestiones de rendimiento, es que una aplicación o producto gestione un conjunto de conexiones y dirija transacciones a una conexión arbitraria. En estas situaciones, el valor que no sea por omisión de una variable global o el valor por omisión inicial no determinista para una variable global solamente deberían ser dependientes hasta que finalice la transacción. Algunos ejemplos de dónde puede producirse este tipo de situación incluyen aplicaciones que utilicen protocolos XA, usen la

CREATE VARIABLE

agrupación de conexiones, empleen el concentrador de conexiones y utilicen HADR para lograr la migración tras error.

- **Privilegios para utilizar una variable global:** Todo intento de leer o grabar en una variable global creada mediante esta sentencia requiere que el ID de autorización que intente esta acción detente el privilegio correspondiente sobre la variable global. Al definidor de la variable se le otorgan de modo implícito todos los privilegios sobre la variable.
- **Establecimiento del valor por omisión:** La primera vez que se hace referencia a una variable global creada dentro de un determinado ámbito se crea una instancia de la misma en su valor por omisión. Tenga en cuenta que si se hace referencia a una variable global en una sentencia, se creará una instancia de la misma con independencia del flujo de control para dicha sentencia.
- **Utilización de una variable global de sesión recién creada:** Si se crea una variable global en una sesión, no podrán utilizarla otras sesiones hasta que se haya confirmado la unidad de trabajo. Sin embargo, la variable global nueva puede utilizarse en la sesión que creó la variable antes de la confirmación de la unidad de trabajo.

Ejemplos

- *Ejemplo 1:* Cree una variable global de sesión para indicar la impresora que ha de utilizarse para la sesión.

```
CREATE VARIABLE MYSCHEMA.MYJOB_PRINTER VARCHAR(30)
DEFAULT 'Impresora por omisión'
```

- *Ejemplo 2:* Cree una variable global de sesión para indicar el departamento en el que trabaja el empleado.

```
CREATE VARIABLE SCHEMA1.GV_DEPTNO INTEGER
DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
WHERE EMPUSER = SESSION_USER))
```

- *Ejemplo 3:* Cree una variable global de sesión para indicar el nivel de seguridad del usuario actual.

```
CREATE VARIABLE SCHEMA2.GV_SECURITY_LEVEL INTEGER
DEFAULT (GET_SECURITY_LEVEL (SESSION_USER))
```

- *Ejemplo 4:* Cree una variable global de sesión como un cursor en la tabla STAFF que devuelva los nombres de cada empleado para cada tipo de trabajo especificado. Ordene los resultados por número de departamento.

```
CREATE VARIABLE STAFFJOBS CURSOR
CONSTANT (CURSOR (WHICHJOB CHAR(5))
FOR SELECT NAME, DEPT FROM STAFF WHERE JOB = WHICHJOB
ORDER BY DEPT)
```

- *Ejemplo 5:* Crear una variable global del tipo de datos XML:

```
CREATE VARIABLE MYSCHEMA.CUSTOMER_HISTORY_VAR XML
```

CREATE VIEW

La sentencia CREATE VIEW define una vista para una o más tablas, vistas o apodos.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la vista no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la vista hace referencia a un esquema existente
- Autorización DBADM

y, como mínimo, una de las autorizaciones siguientes para cada tabla, vista o apodo que se identifica en cualquier selección completa:

- Privilegio CONTROL para esa tabla, vista o apodo
- Privilegio SELECT para esa tabla, vista o apodo
- Autorización DATAACCESS

Si se crea una subvista:

- El ID de autorización de la sentencia debe ser el mismo que el del definidor de la tabla raíz de la jerarquía de tablas; o bien
- Los privilegios que tiene el ID de autorización deben incluir la autorización DBADM

y

- El ID de autorización de la sentencia debe tener el privilegio SELECT WITH GRANT en la tabla subyacente de la subvista, o bien la supervista no debe tener el privilegio SELECT otorgado a ningún usuario distinto del definidor de la vista; o bien
- Autorización ACCESSCTRL y una de las autorizaciones siguientes:
 - Privilegio SELECT en la tabla subyacente de la subvista
 - Autorización DATAACCESS

Si se especifica WITH ROW MOVEMENT, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio UPDATE en la tabla o vista
- Autorización DATAACCESS

No se tienen en cuenta los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE VIEW.

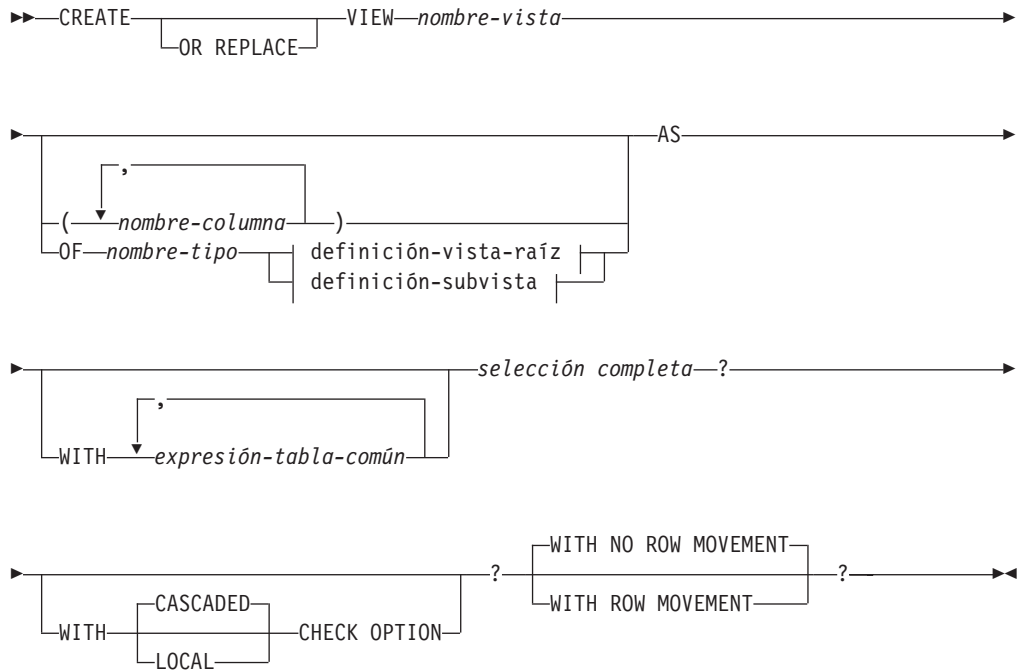
No se tienen en cuenta los privilegios al definir una vista para un apodo de base de datos federada. Los requisitos de autorización de la fuente de datos para la

CREATE VIEW

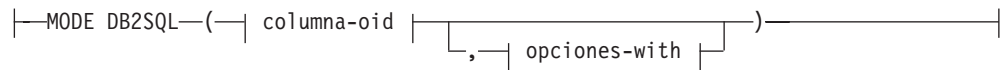
tabla o vista referenciada por el apodo se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización remoto distinto.

Para sustituir una vista existente, el ID de autorización de la sentencia debe ser el propietario de la vista existente (SQLSTATE 42501).

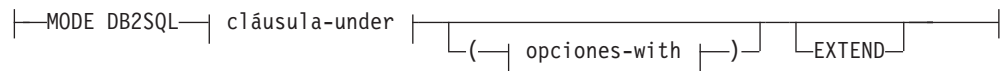
Sintaxis



definición-vista-raíz:

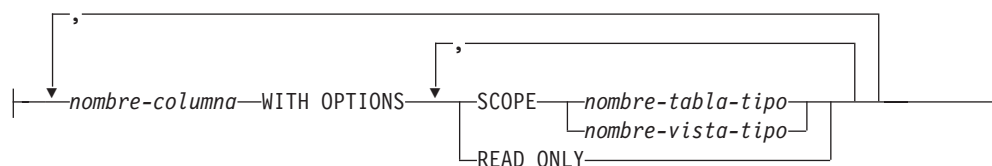


definición-subvista:



columna-oid:



opciones-with:**cláusula-under:****Descripción****OR REPLACE**

Especifica que se debe sustituir la definición de la vista si existe una en el servidor actual. La definición existente se descarta de forma efectiva antes de que la nueva definición se sustituya en el catálogo, con la excepción de que los privilegios que se han otorgado sobre la vista no se ven afectados por ello. Esta opción se ignora si no existe una definición para la vista en el servidor actual. Esta opción sólo puede especificarla el propietario del objeto.

nombre-vista

Indica el nombre de la vista. El nombre (incluido el calificador implícito o explícito) no debe designar una tabla, vista, apodo ni alias que se haya descrito 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 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, esta debe constar de tantos nombres como columnas haya en la tabla de resultados de la selección completa. Cada *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 de resultados de la selección completa.

Debe especificarse una lista de nombres de columna si la tabla de resultados 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.

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

CREATE VIEW

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 una 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 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 eliminación de referencia 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 953 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 ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en el *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.

CREATE VIEW

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.

selección completa

Define la vista. En todo momento, la vista consta de las filas que se generarían si se ejecutara la sentencia SELECT. El tipo de datos de las columnas de la vista no puede ser un tipo diferenciado con restricciones de tipo de datos, tipo de matriz, tipo de cursor o tipo de fila. La selección completa no debe hacer referencia a variables del lenguaje principal, a marcadores de parámetro ni a tablas temporales declaradas. En cambio, una vista parametrizada se puede crear como función de tabla SQL.

La selección completa no puede incluir una sentencia de cambio de datos de SQL en la cláusula FROM (SQLSTATE 428FL).

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 DBPARTITIONNUM o HASHEDVALUE, a las funciones no deterministas o a funciones que se han definido para que exista una acción externa.
- El cuerpo de la vista debe consistir en una sola subselección o en UNION ALL de dos o más 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 (es decir, una vista no puede tener varias ramas 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, la vista de la raíz en la jerarquía debe especificar la opción UNCHECKED para su columna de 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.
- 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 se da alguna de las condiciones siguientes:

- La vista es de sólo lectura (SQLSTATE 42813). Si se ha especificado WITH CHECK OPTION para una vista que puede actualizarse y que no admite inserciones, la restricción se aplicará sólo a las actualizaciones.
- La vista hace referencia a la función DBPARTITIONNUM o HASHEDVALUE, a una función no determinante o a una función con acción externa (SQLSTATE 42997).
- Un apodo es el destino de actualización de la vista.
- Una vista que tiene definido un activador INSTEAD OF es el destino de actualización de la vista (SQLSTATE 428FQ).

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):

CREATE VIEW

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
```

La siguiente sentencia INSERT que utiliza V1 será satisfactoria porque V1 no tiene WITH CHECK OPTION y V1 no 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)
```

WITH NO ROW MOVEMENT o WITH ROW MOVEMENT

Especifica la acción a emprender para una vista UNION ALL actualizable cuando se actualiza una fila de modo que viola una restricción de comprobación de la tabla subyacente. El valor por omisión es WITH NO ROW MOVEMENT.

WITH NO ROW MOVEMENT

Especifica que se devolverá un error (SQLSTATE 23513) si se actualiza una fila mediante un método que infrinja restricción de comprobación de la tabla subyacente.

WITH ROW MOVEMENT

Especifica que se debe mover una fila actualizada a la tabla subyacente adecuada, aunque viole una restricción de comprobación en dicha tabla.

El movimiento de filas implica la supresión de las filas que violan la restricción de comprobación y la inserción de dichas filas de nuevo en la

vista. La cláusula WITH ROW MOVEMENT sólo se puede especificar para vistas UNION ALL cuyas columnas sean actualizables (SQLSTATE 429BJ). Si se inserta una fila (quizás después de la activación de un activador) en la misma tabla subyacente desde la que se ha suprimido, se devuelve un error (SQLSTATE 23524). Una vista definida utilizando la cláusula WITH ROW MOVEMENT no debe contener operaciones UNION ALL anidadas, excepto en la selección completa más externa (SQLSTATE 429BJ). Una vista definida utilizando la cláusula WITH ROW MOVEMENT no puede contener ninguna referencia a una tabla temporal de período de sistema, tabla temporal de período de aplicación o tabla bitemporal.

Notas

- La creación de una vista 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 disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN sobre 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 referencial y control) si es que hay alguna definida en la tabla base.
- No se puede crear una nueva vista si esta utiliza en su definición una vista no operativa. (SQLSTATE 51024).
- Si un objeto al que se hace referencia en el cuerpo de la vista no existe o se ha marcado como no válido o el definidor no tiene temporalmente los privilegios para acceder al objeto y, si el parámetro de configuración de la base de datos **auto_reval** no se ha establecido en DISABLED, la vista seguirá creándose satisfactoriamente. La vista se marcará como no válida y se volverá a validar la siguiente vez que se referencie.
- Esta sentencia no permite utilizar tablas temporales declaradas (SQLSTATE 42995).
- **Vistas que pueden suprimirse:** Una vista *podrá suprimirse* si para la vista se ha definido un activador INSTEAD OF para la operación de supresión o si se dan todas las condiciones siguientes:
 - Cada cláusula FROM de la selección completa externa sólo identifica una 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 a un apodo). Además, cualquier especificación de período especificada para la tabla base o la vista suprimible no hace referencia al período SYSTEM_TIME.
 - 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 agregadas en la lista de selección.
 - La selección completa externa 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 la misma tabla y cada operando debe poder suprimirse.
 - La lista de selección de la selección completa externa no incluye DISTINCT.

CREATE VIEW

- **Vistas que pueden actualizarse:** Una columna de una vista *podrá actualizarse* si para la vista se ha definido un activador INSTEAD OF para la operación de actualización o si se dan todas las condiciones siguientes:
 - La vista puede suprimirse (con independencia de la existencia de un activador INSTEAD OF para la supresión), la columna se resuelve en una columna de una tabla base (sin utilizarse una operación de eliminación de referencia) y no se ha especificado 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 puede actualizarse si puede actualizarse *cualquier* columna de la vista.

- **Vistas que se pueden insertar::** una vista se puede insertar si se ha definido un activador INSTEAD OF para la operación de inserción de la vista o, como mínimo, una columna de la vista se puede actualizar (de modo independiente respecto a un activador INSTEAD OF para actualizar), y la selección completa de la vista no incluye UNION ALL.

Una fila determinada se puede insertar en una vista (incluida una UNION ALL) solamente si cumple las restricciones de comprobación de exactamente una de las tablas base subyacentes.

Para insertarla en una vista que incluya columnas que no se pueden actualizar, estas columnas deben omitirse de la lista de columnas.

- **Vistas de sólo lectura:** una vista es de *sólo-lectura* si *no* es suprimible, actualizable o insertable.

La columna READONLY de la vista de catálogo SYSCAT.VIEWS indica si una vista es de sólo lectura sin tenerse en cuenta las especificaciones de período o los activadores INSTEAD OF.

- Las expresiones de tabla comunes y las expresiones de tablas anidadas deben seguir el mismo conjunto de normas para determinar si se pueden suprimir, actualizar, insertar o si son de sólo lectura.
- **Registros especiales para soporte temporal:** Los valores de los registros especiales CURRENT TEMPORAL SYSTEM_TIME y CURRENT TEMPORAL BUSINESS_TIME no tienen ningún impacto en la expresión de consulta que define una vista mientras se está definiendo. Cuando se utiliza una vista en una sentencia de SQL, los valores de los registros especiales CURRENT TEMPORAL SYSTEM_TIME y CURRENT TEMPORAL BUSINESS_TIME se aplican a la vista en la sesión que procesa la sentencia de SQL.
- **Vistas no operativas:** una *vista no operativa* es una vista que ya no está disponible para sentencias de SQL. Una vista deja de ser operativa si:
 - Se revoca un privilegio del que depende la definición de dicha vista.
 - Se descarta un objeto, como una tabla, un apodo, un alias 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 descartado involuntariamente la definición de vista. Por ejemplo, al descartar un alias, cualquier vista definida con ese alias deja de ser 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 vuelva a crearse o descartarse explícitamente una vista no operativa, no podrá compilarse una sentencia que utilice esa vista no operativa

(SQLSTATE 51024), a excepción de las sentencias CREATE ALIAS, CREATE VIEW, DROP VIEW y COMMENT ON TABLE. Hasta que no se haya descartado explícitamente la vista no operativa, no se puede utilizar su nombre calificado para crear otra tabla o alias (SQLSTATE 42710).

Una vista no operativa podría 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 descartar explícitamente la vista no operativa para volver a crearla. 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 sobre la vista así como el derecho de descartar la vista. La persona que define una vista sólo tiene el privilegio CONTROL sobre la vista si posee el privilegio CONTROL en todas las tablas base, vistas o apodos identificados en la selección completa, o si el usuario que define la vista dispone de todas las autorizaciones siguientes:
 - ACCESSCTRL o SECADM
 - DATAACCESS
 - DBADM

Al definidor de la vista se le otorgan los privilegios INSERT, UPDATE, UPDATE de nivel de columna o DELETE en la vista si esta no es de sólo lectura y el definidor dispone de los privilegios correspondientes en los objetos subyacentes.

Para una vista definida WITH ROW MOVEMENT, la persona que la define adquiere el privilegio UPDATE sobre ella sólo si tiene el privilegio UPDATE sobre todas las columnas de la vista, así como los privilegios INSERT y DELETE sobre todas las tablas o vistas 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 este privilegio. Los privilegios no se tienen en cuenta al definir una vista en 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, los privilegios de SELECT que tiene la supervista inmediata se otorgan automáticamente a la subvista.

- **Columnas REF y de ámbito:** cuando se selecciona una columna de tipo de referencia en la selección completa de una definición de vista, considere el tipo de destino y el ámbito que se necesita.
 - 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.

CREATE VIEW

- 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:** Una columna de una vista se considera una columna de identidad, si el elemento de la columna correspondiente en la selección completa de la definición de la vista es el nombre de la 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 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 que la lista de selecciones de la definición de vista incluye directa o indirectamente el nombre de una columna de identidad de una tabla base, se aplican las mismas normas que si la declaración INSERT hicieran 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 cuando posteriormente se hace referencia a la misma en una consulta.

Cuando se crea la vista, no se realiza ninguna comprobación de privilegios para determinar si el definidor de la vista tiene acceso a la tabla de fuente de datos subyacente o a la vista de un apodo. La comprobación de privilegios de referencias a las tablas o las vistas de la base de datos federada se lleva a cabo del modo habitual, y se requiere que el definidor de la vista tenga al menos el privilegio SELECT en dichos objetos.

Cuando posteriormente se hace referencia en una consulta, a una vista federada, los apodos que resultan de las consultas en la fuente de datos y el ID de autorización que ha emitido la consulta (o el ID de autorización remota con el que se correlaciona) deben tener los privilegios necesarios para acceder a la tabla o vista de la fuente de datos. El ID de autorización que emite la consulta que hace referencia a la vista federada no es necesario que tenga privilegios adicionales en tablas o vistas (sin federar) que existen en el servidor federado.

- **ROW MOVEMENT, activadores y restricciones:** Cuando se actualiza una vista definida mediante la cláusula WITH ROW MOVEMENT, la secuencia de operaciones de activador y restricciones es la siguiente:
 1. Los activadores BEFORE UPDATE se activan para todas las filas que se están actualizando, incluidas las filas que finalmente se moverán.
 2. Se procesa la operación de actualización.
 3. Se procesan las restricciones para todas las filas actualizadas.

4. Los activadores AFTER UPDATE (tanto de nivel de fila como de nivel de sentencia) se activan en el orden de creación, para todas las filas que satisfacen las restricciones tras la operación de actualización. Puesto que de trata de una sentencia UPDATE, todos los activadores del nivel de sentencia UPDATE se activan para todas las tablas subyacentes.
 5. Los activadores BEFORE DELETE se activan para todas las filas que no han cumplidos las restricciones tras la operación de actualización (estas filas son las que debe moverse).
 6. Se procesa la operación de supresión.
 7. Se procesan las restricciones para todas las filas suprimidas.
 8. Los activadores AFTER DELETE (tanto de nivel de fila como de nivel de sentencia) se activan en orden de creación, para todas las filas suprimidas. Los activadores del nivel de sentencia se activan únicamente para las tablas implicadas en la operación de supresión.
 9. Los activadores BEFORE INSERT se activan para todas las filas que se van a insertar (es decir, las filas que se van a mover). Las nuevas tablas de transición para los activadores BEFORE INSERT contienen los datos de entrada que ha proporcionado el usuario. Estos activadores no contienen ninguna operación UPDATE, DELETE o INSERT, ni invocan ninguna rutina que contenga dichas operaciones (SQLSTATE 42987).
 10. Se procesa la operación de inserción.
 11. Se procesan las restricciones para todas las filas insertadas.
 12. Los activadores AFTER INSERT (tanto de nivel de fila como de nivel de sentencia) se activan en el orden de creación para todas las filas insertadas. Los activadores del nivel de sentencia se activan únicamente para las tablas implicadas en la operación de inserción.
- **Vistas UNION ALL anidadas:** Una vista definida con UNION ALL y basada, directa o indirectamente, en una vista que también está definida con UNION ALL no se puede actualizar si alguna de las vistas se ha definido utilizando la cláusula WITH ROW MOVEMENT (SQLSTATE 429BK).
 - **Consideraciones para columnas implícitamente ocultas:** es posible que la tabla resultante de la selección completa incluya una columna de la tabla base que está definida como implícitamente oculta. Esto puede suceder cuando se hace referencia explícitamente a la columna implícitamente oculta en la selección completa de la definición de vista. Sin embargo, la columna correspondiente de la vista no hereda el atributo de implícitamente oculta. Las columnas de una vista no pueden definirse como ocultas.
 - **Subselección:** la *cláusula-isolation* no puede especificarse en la *selección-completa* (SQLSTATE 42601).
 - **Ofuscación:** La sentencia CREATE VIEW se puede enviar en formato ofuscado. En una sentencia ofuscada, sólo es legible en nombre de la vista. El resto de la sentencia se codifica de tal manera que no es legible, pero que el servidor de bases de datos puede descodificar. Las sentencias ofuscadas se pueden generar mediante una llamada a la función DBMS_DDL.WRAP.
 - **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos.
 - La palabra clave FEDERATED puede especificarse entre las palabras clave CREATE y VIEW. Sin embargo, se ignora la palabra clave FEDERATED porque ya no se devuelve un aviso si se utilizan objetos federados en la definición de la vista.

Ejemplos

- *Ejemplo 1:* Crear una vista denominada MA_PROJ basada en la tabla PROJECT que solo contenga las filas cuyo número de proyecto (PROJNO) 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 PROJECT.

```
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
```

Puede evitarse la especificación de la lista de nombres de columna especificando el nombre de la expresión SALARY+BONUS+COMM como 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:

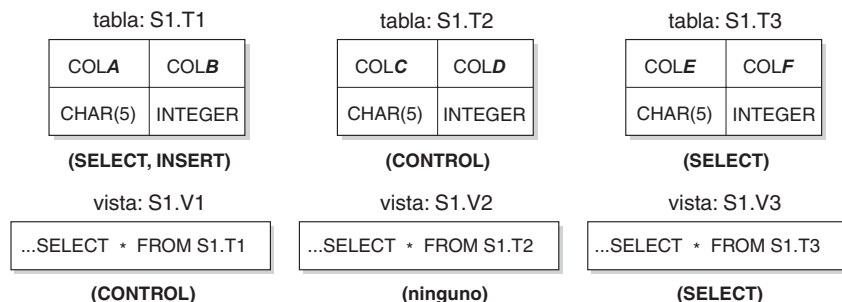


Figura 1. Tablas y vistas para el ejemplo 6

El usuario ZORPIE (que no tiene autorización ACCESSCTRL, DATAACCESS ni DBADM) tiene los privilegios que se muestran entre paréntesis para cada objeto:

- 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. (CONTROL sobre S1.V1 debe otorgarse a ZORPIE por la persona que tiene autorización ACCESSCTRL o SECADM.) No importa qué privilegios tiene, si tiene alguno, para la tabla base subyacente.

- 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).

- 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.

- 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
```

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.

- 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

CREATE VIEW

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 WORK ACTION SET

La sentencia CREATE WORK ACTION SET define un conjunto de acciones de trabajo y acciones de trabajo en el conjunto de acciones de trabajo.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Sintaxis

► CREATE WORK ACTION SET *nombre-conjunto-acciones-trabajo* ►

► FOR DATABASE *nombre-superclase-servicio* |
 SERVICE CLASS *nombre-carga-trabajo* |
 WORKLOAD *nombre-carga-trabajo* ►

► USING WORK CLASS SET *nombre-conjunto-clases-trabajo* ►

► (*definición-acción-trabajo*) [ENABLE | DISABLE] ►

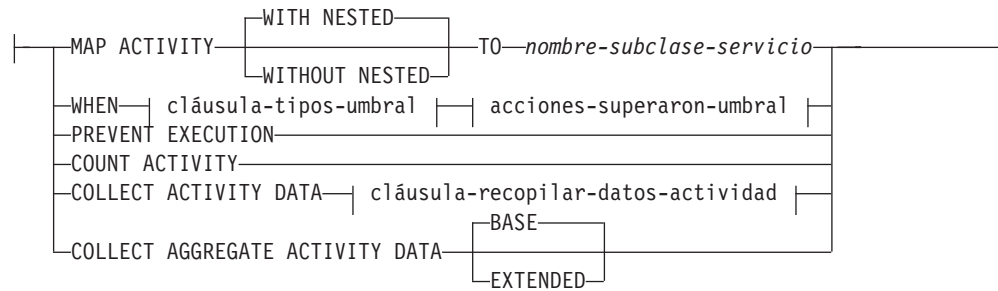
definición-acción-trabajo:

| WORK ACTION *nombre-acción-trabajo* ON WORK CLASS *nombre-clase-trabajo* ►

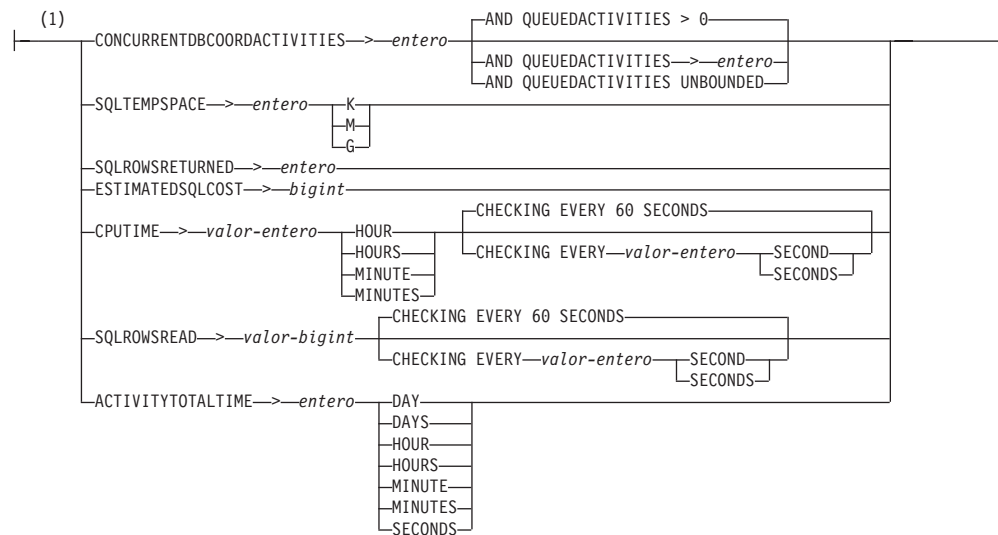
► | *cláusula-tipos-acción* | | *cláusula-plantilla-histograma* | [ENABLE | DISABLE] |

CREATE WORK ACTION SET

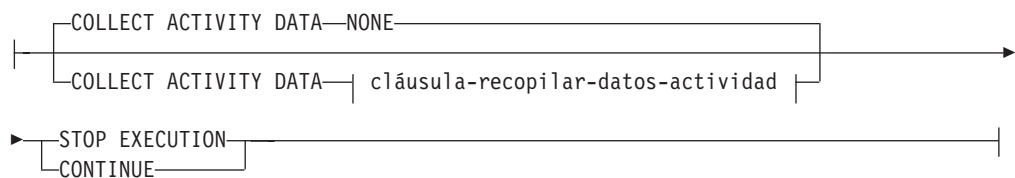
cláusula-tipos-acción:



cláusula-tipos-umbral:

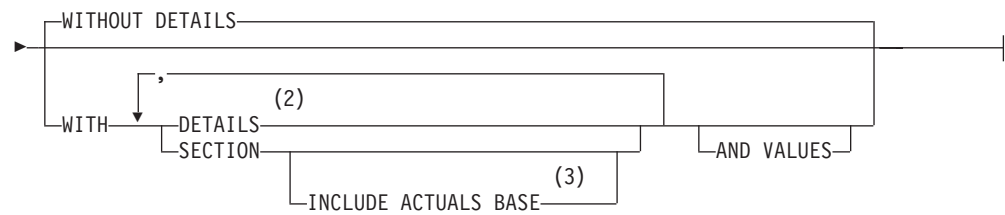


acciones-superaron-umbral:

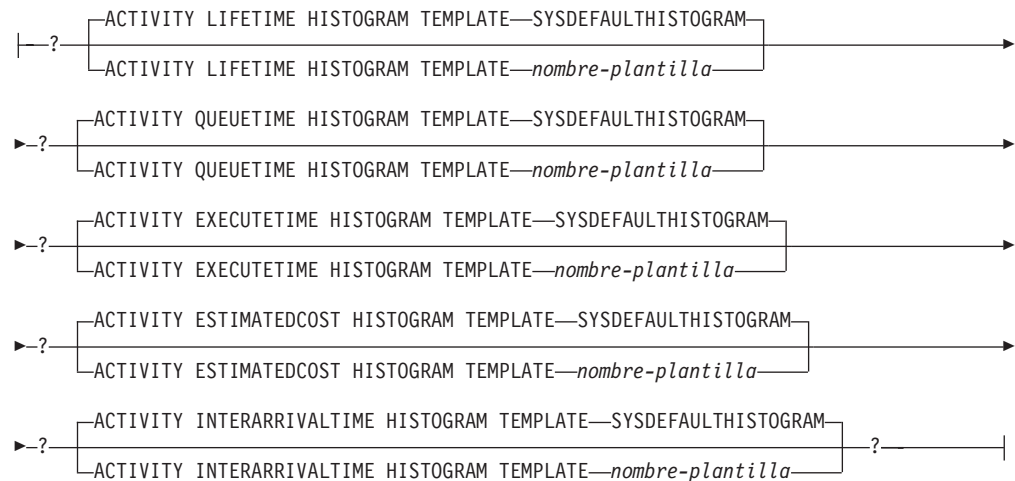


cláusula-recopilar-datos-actividad:





cláusula-plantilla-histograma:



Notas:

- 1 Sólo puede aplicarse una acción de trabajo del mismo tipo de umbral a una única clase de trabajo a la vez.
- 2 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.
- 3 Esta cláusula no se aplica a los umbrales.

Descripción

nombre-conjunto-acciones-trabajo

Da nombre al conjunto de acciones de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-acciones-trabajo* no debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

FOR

Especifica el objeto de gestor de base de datos al que se aplicarán las acciones de este conjunto de acciones de trabajo. Cada uno de los objetos de gestor de base de datos sólo puede tener definido un conjunto de acciones de trabajo definida para el mismo (SQLSTATE 5U017).

DATABASE

Las acciones de este conjunto de acciones de trabajo van a aplicarse a la base de datos. Si se especifica DATABASE, la cláusula MAP ACTIVITY no puede especificarse (SQLSTATE 5U034).

SERVICE CLASS *nombre-superclase-servicio*

Las acciones de este conjunto de acciones de trabajo van a aplicarse a

CREATE WORK ACTION SET

nombre-superclase-servicio. Si se especifica SERVICE CLASS, no pueden especificarse acciones de umbral (SQLSTATE 5U034). *nombre-superclase-servicio* debe existir en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no debe ser una subclase de servicio y no puede ser ninguna de las clases siguientes (SQLSTATE 5U032):

- La clase de servicio de sistema (SYSDEFAULTSYSTEMCLASS)
- La clase de servicio de mantenimiento (SYSDEFAULTMAINTENANCECLASS)
- la clase de servicio de usuario por omisión (SYSDEFAULTUSERCLASS)

WORKLOAD *nombre-carga-trabajo*

Las acciones de este conjunto de acciones de trabajo han de aplicarse a la carga de trabajo *nombre-carga-trabajo*. Si se especifica WORKLOAD, no puede especificarse la acción MAP ACTIVITY (SQLSTATE 5U034). El *nombre-carga-trabajo* debe existir en el servidor actual (SQLSTATE 42704). El *nombre-carga-trabajo* no puede ser SYSDEFAULTADMWORKLOAD (SQLSTATE 5U032).

USING WORK CLASS SET *nombre-conjunto-clases-trabajo*

Especifica el conjunto de clases de trabajo que contiene las clases de trabajo que clasificarán actividades de base de datos en las que realizar acciones. *nombre-conjunto-clases-trabajo* debe existir en el servidor actual (SQLSTATE 42704).

definición-acción-trabajo

Especifica la definición de la acción de trabajo.

WORK ACTION *nombre-acción-trabajo*

Da nombre a la acción de trabajo. El *nombre-acción-trabajo* no debe identificar una acción de trabajo que ya exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42710). El *nombre-acción-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

ON WORK CLASS *nombre-clase-trabajo*

Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY

Especifica una acción de trabajo de correlación de la actividad. Esta acción sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED

Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED

Todas las actividades de base de datos que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de datos anidadas bajo esta actividad, se correlacionarán a la subclase de servicio; es decir, las actividades con un nivel de anidamiento superior a cero se ejecutarán en la misma clase de servicio que las actividades con un nivel de anidamiento de cero.

WITHOUT NESTED

Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO nombre-subclase-servicio

Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN

Especifica el umbral que se aplicará a la actividad de base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo. Un umbral sólo puede especificarse si el objeto de gestor de base de datos para el que se ha definido esta acción de trabajo está definido como base de datos o carga de trabajo (SQLSTATE 5U034). Ninguno de estos umbrales se aplican a las actividades de base de datos internas iniciadas por medio del gestor de base de datos o a las actividades de base de datos generadas por medio de rutinas de SQL administrativas.

cláusula-tipos-umbral

Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia "CREATE THRESHOLD".

acciones-superaron-umbral

Para obtener una descripción de las acciones de excedido umbral válidas, consulte la sentencia "CREATE THRESHOLD".

PREVENT EXECUTION

Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY

Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo para las que se define esta acción de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad asociada a la clase de trabajo para la que esta acción de trabajo está definida, deben enviarse a cualquier supervisor de sucesos de actividades activas cuando se completa la actividad. El valor por omisión es COLLECT ACTIVITY DATA WITHOUT DETAILS.

*cláusula-recopilar-datos-actividad***ON COORDINATOR MEMBER**

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se

CREATE WORK ACTION SET

especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad que se ejecuta en la clase de servicio deben enviarse a cualquier supervisor de sucesos de actividades activas, cuando la ejecución completa la actividad. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH

DETAILS

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. Los datos reales de sección sólo se recopilarán en cualquier miembro donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actuals**.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula INCLUDE ACTUALS, el parámetro de configuración de base de datos **section_actuals** y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del parámetro de configuración de la base de datos **section_actuals** es NONE y <collectsectionactuals> se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

BASE especifica que lo siguiente debe habilitarse y recopilarse durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos `wlm_collect_int`. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE

Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal. Sólo las actividades a las que se ha aplicado un umbral `SQLTEMPSPACE` participan en esta marca de límite superior.
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED

Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

ENABLE o DISABLE

Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

ENABLE

Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE

Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometan actividades de base de datos.

ENABLE o DISABLE

Especifica si el conjunto de acciones de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

CREATE WORK ACTION SET

ENABLE

Especifica que el conjunto de acciones de trabajo está habilitado y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE

Especifica que el conjunto de acciones de trabajo está inhabilitado y no va a tomarse en consideración cuando se sometan actividades de base de datos.

cláusula-plantilla-histograma

Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades asociadas a la clase de trabajo a la que se asigna esta acción de trabajo. Los datos de actividades agregadas sólo se recopilan para la clase de trabajo cuando el tipo de acción de trabajo es COLLECT AGGREGATE ACTIVITY DATA.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos de las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada miembro en el que se ejecuta la actividad. En el miembro coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En los miembros que no son coordinadores, este es el tiempo que estos miembros pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en un miembro no coordinador más de una vez y cada vez el miembro no coordinador recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en un miembro. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar

datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (plantilla de histograma)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (clase de servicio)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (umbral)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (conjunto de acciones de trabajo)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (conjunto de clases de trabajo)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (carga de trabajo)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
- El ámbito de aplicación se basa de forma automática en el tipo de umbral. Para los umbrales de tipo CONCURRENTDBCOORDACTIVITIES, el entorno se

CREATE WORK ACTION SET

utiliza también para determinar el ámbito de aplicación, donde toma como valor por omisión el ámbito de aplicación DATABASE en entornos que no son de DB2 pureScale, y el ámbito de aplicación MEMBER en los entornos de DB2 pureScale.

Ejemplos

- *Ejemplo 1:* Cree un conjunto de acciones de trabajo denominado DATABASE_ACTIONS que se aplique a todas las actividades de base de datos. Utilice el conjunto de clases de trabajo LARGE_QUERIES y defina las siguientes acciones de trabajo. La acción de trabajo ONE_CONCURRENT_QUERY tiene una acción de umbral que permita que una consulta simultánea se ejecute en el sistema a la vez para las consultas que caigan en la clase de trabajo LARGE_ESTIMATED_COST. Si se supera dicho umbral, el gestor de base de datos va a poner en cola la actividad, pero no va a permitir que se ponga en cola más de una actividad la vez. Si se supera el umbral, no se permite la ejecución de la actividad de base de datos. La acción de trabajo TWO_CONCURRENT_QUERIES tiene una acción de umbral que permite que dos consultas simultáneas se ejecuten a la vez para las consultas que caigan dentro de la clase de trabajo LARGE_CARDINALITY y no permite que se pongan en cola más de dos. Si se van a poner en cola más de dos consultas, la actividad de base de datos va a seguir colocando las consultas en la cola y va a recopilar los datos de actividad de base de datos en el supervisor de sucesos de actividades, si hay alguno activo.

```
CREATE WORK ACTION SET DATABASE_ACTIONS
FOR DATABASE USING WORK CLASS SET LARGE_QUERIES
(WORK ACTION ONE_CONCURRENT_QUERY ON WORK CLASS LARGE_ESTIMATED_COST
WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1
STOP EXECUTION,
WORK ACTION TWO_CONCURRENT_QUERIES ON WORK CLASS LARGE_CARDINALITY
WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 2
COLLECT ACTIVITY DATA CONTINUE)
```

- *Ejemplo 2:* Cree un conjunto de acciones de trabajo denominada ADMIN_APPS_ACTIONS con una acción de trabajo denominada MAP_SELECTS que va a aplicarse a las actividades de base de datos que se ejecutan bajo la superclase de servicio ADMIN_APPS. La acción de trabajo va a correlacionar toda la actividad de base de datos que caiga dentro de la clase de trabajo SELECT_CLASS con la subclase de servicio SELECTS_SERVICE_CLASS, que está en el conjunto de clases de trabajo DML_SELECTS.

```
CREATE WORK ACTION SET ADMIN_APPS_ACTIONS
FOR SERVICE CLASS ADMIN_APPS USING
WORK CLASS SET DML_SELECTS
(WORK ACTION MAP_SELECTS ON WORK CLASS SELECT_CLASS
MAP ACTIVITY TO SELECTS_SERVICE_CLASS)
```


CREATE WORK CLASS SET

La sentencia CREATE WORK CLASS SET define un conjunto de clases de trabajo.

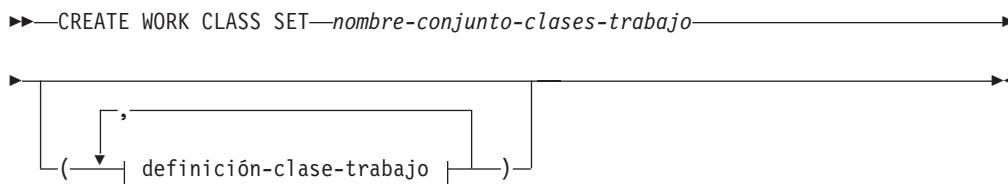
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

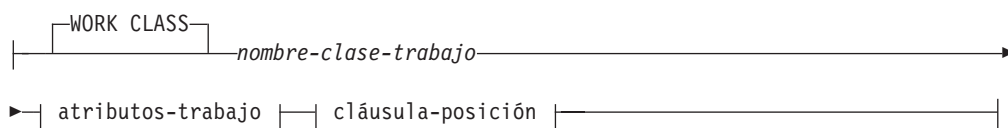
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

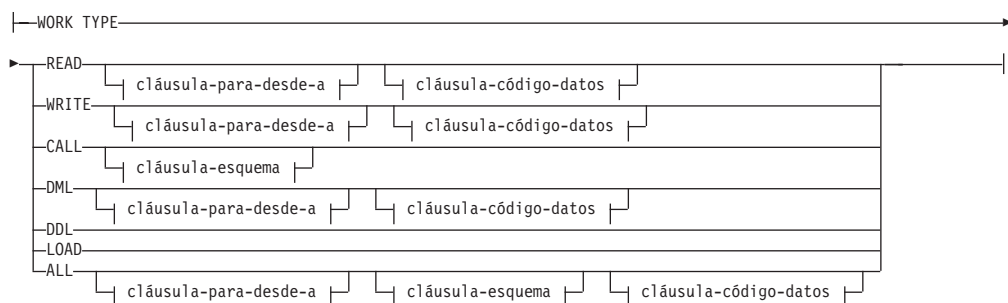
Sintaxis



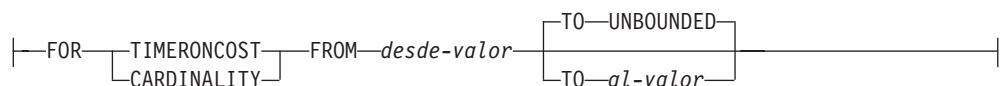
definición-clase-trabajo:



atributos-trabajo:



cláusula-para-desde-a:



CREATE WORK CLASS SET

cláusula-código-datos:

|—DATA TAG LIST CONTAINS*constante-entera*—|

cláusula-esquema:

|—ROUTINES IN SCHEMA—*nombre-esquema*—|

cláusula-posición:

|—POSITION LAST—|
|—POSITION BEFORE—*nombre-clase-trabajo*—|
|—POSITION AFTER—*nombre-clase-trabajo*—|
|—POSITION AT—*posición*—|

Descripción

nombre-conjunto-clases-trabajo

Da nombre al conjunto de clases de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-clases-trabajo* no debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

definición-clase-trabajo

Especifica la definición de la clase de trabajo.

WORK CLASS *nombre-clase-trabajo*

Da nombre a la clase de trabajo. El *nombre-clase-trabajo* no debe identificar una clase de trabajo que ya existe en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42710). El *nombre-clase-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

atributos-trabajo

Los atributos de la actividad de base de datos deben corresponderse con todos los atributos especificados en esta clase de trabajo si va a asociarse dicha actividad con esta clase de trabajo.

WORK TYPE

Especifica el tipo de actividad de base de datos.

READ

Esta actividad incluye las sentencias siguientes:

- Todas las sentencias SELECT o SELECT INTO que no contengan una sentencia DELETE, INSERT, MERGE o UPDATE y todas las sentencias VALUES INTO
- Todas las sentencias XQuery

WRITE

Esta actividad incluye las sentencias siguientes:

- UPDATE
- DELETE
- INSERT
- MERGE

- Todas las sentencias SELECT que contengan una sentencia DELETE, INSERT o UPDATE y todas las sentencias VALUES INTO

CALL

Incluye la sentencia CALL. Se toma en consideración una sentencia CALL para una clase de trabajo con un tipo de trabajo de CALL o ALL.

DML

Incluye las sentencias listadas bajo READ y WRITE.

DDL

Esta actividad incluye las sentencias siguientes:

- ALTER
- CREATE
- COMMENT
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- FLUSH PACKAGE CACHE
- GRANT
- REFRESH TABLE
- RENAME
- REVOKE
- SET INTEGRITY

LOAD

Operaciones de carga DB2.

ALL

Toda actividad de gestión de carga de trabajo (WLM) reconocida que se clasifique en una de las palabras clave enumeradas anteriormente en la descripción de WORK TYPE.

FOR

Indica el tipo de información que está especificándose en la cláusula FROM *desde-valor* TO *al-valor*. La cláusula FOR sólo se utiliza para los siguientes tipos de trabajo:

- ALL
- DML
- READ
- WRITE

TIMERONCOST

El coste estimado del trabajo, en acciones de temporización. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

CARDINALITY

La cardinalidad estimada del trabajo. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

FROM *desde-valor* TO UNBOUNDED o FROM *desde-valor* TO *al-valor*

Especifica el rango de valor de activación de temporizador (para el coste estimado) o cardinalidad en la que debe caer la actividad de la base de datos si va a formar parte de esta clase de trabajo. El rango

CREATE WORK CLASS SET

incluye *desde-valor* y *al-valor*. Si esta cláusula no se especifica para la clase de trabajo, se incluirá todo trabajo que caiga en el tipo de trabajo especificado (es decir, el valor por omisión es FROM 0 TO UNBOUNDED). Este rango sólo se utiliza para los siguientes tipos de trabajo:

- ALL
- DML
- READ
- WRITE

FROM *desde-valor* TO UNBOUNDED

desde-valor debe ser cero o un valor DOUBLE positivo (SQLSTATE 5U019). El rango no tiene vinculación superior.

FROM *desde-valor* TO *al-valor*

desde-valor debe ser cero o un valor DOUBLE positivo y *al-valor* debe ser un valor DOUBLE positivo. *desde-valor* debe ser igual o más pequeño que *al-valor* (SQLSTATE 5U019).

DATA TAG LIST CONTAINS *constante-entera*

Especifica el valor del código que se proporciona a cualquier dato que puede tocar la actividad de la base de datos si va a formar parte de esta clase de trabajo. Si no se especifica la cláusula para la clase de trabajo, todo el trabajo que se inscribe dentro del tipo de trabajo especificado, independientemente de los datos que pueda tocar, se incluirá (es decir, el valor por omisión es ignorar el código de datos). Esta cláusula sólo se utiliza si el tipo de trabajo es READ, WRITE, DML o ALL y la actividad de base de datos es una sentencia DML. Los valores válidos para *constante-entero* son enteros del 1 al 9.

cláusula-esquema

ROUTINES IN SCHEMA *nombre-esquema*

Especifica el nombre de esquema del procedimiento que estará llamando la sentencia CALL. Esta cláusula sólo se utiliza si el tipo de trabajo es CALL o ALL y la actividad de base de datos es una sentencia CALL. Si no se especifica ningún valor, se incluirán todos los esquemas.

cláusula-posición

POSITION

Especifica el lugar en el que va a colocarse esta clase de trabajo dentro del conjunto de clases de trabajo, lo cual determina el orden en el que se evalúan estas clases de trabajo. Al efectuar la asignación de clases de trabajo en tiempo de ejecución, el gestor de base de datos determina en primer lugar el conjunto de clases de trabajo asociado al objeto, la base de datos o una superclase de servicio. Se selecciona la primera clase de trabajo coincidente en dicho conjunto de clases de trabajo. Si no se especifica esta palabra clave, la clase de trabajo se ubica en la última posición.

LAST

Especifica que la clase de trabajo va a colocarse en última posición en la lista ordenada de las clases de trabajo del conjunto de clases de trabajo. Es el valor por omisión.

BEFORE *nombre-clase-trabajo*

Especifica que la clase de trabajo va a colocarse antes de la clase de

trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AFTER *nombre-clase-trabajo*

Especifica que la clase de trabajo va a colocarse detrás de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AT *posición*

Especifica la posición absoluta en la que la carga de trabajo va a colocarse en el conjunto de clases de trabajo en la lista ordenada de clases de trabajo. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42615). Si *posición* es mayor que el número de clases de trabajo existentes más una, la clase de trabajo se sitúa en la última posición del conjunto de clases de trabajo.

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.

Ejemplos

- *Ejemplo 1:* Cree un conjunto de clases de trabajo denominado LARGE_QUERIES que tenga un conjunto de clases de trabajo que representen todo el DML con un coste estimado superior a 9999 y una cardinalidad estimada superior a 1000.

```
CREATE WORK CLASS
SET LARGE_QUERIES
(WORK CLASS LARGE_ESTIMATED_COST WORK TYPE DML
FOR TIMERONCOST FROM 9999 TO UNBOUNDED,
WORK CLASS LARGE_CARDINALITY WORK TYPE DML
FOR CARDINALITY FROM 1000 TO UNBOUNDED)
```

CREATE WORK CLASS SET

- *Ejemplo 2:* Cree una clase de trabajo denominada DML_SELECTS que tenga una clase de trabajo que represente a todas las sentencias DML SELECT que no contengan una sentencia DELETE, INSERT, MERGE o UPDATE.

```
CREATE WORK CLASS SET DML_SELECTS  
(WORK CLASS SELECT_CLASS WORK TYPE READ)
```

CREATE WORKLOAD

La sentencia CREATE WORKLOAD define una carga de trabajo.

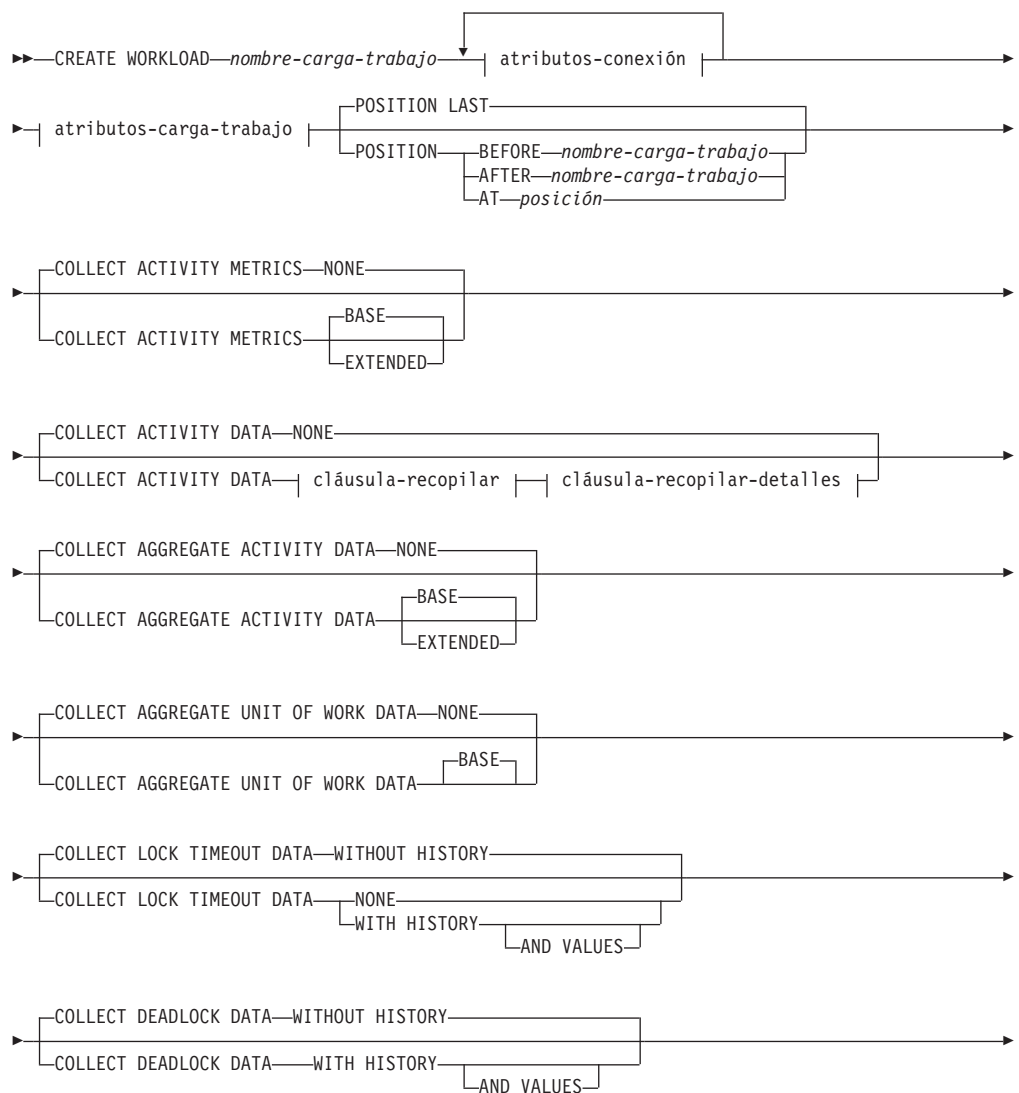
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

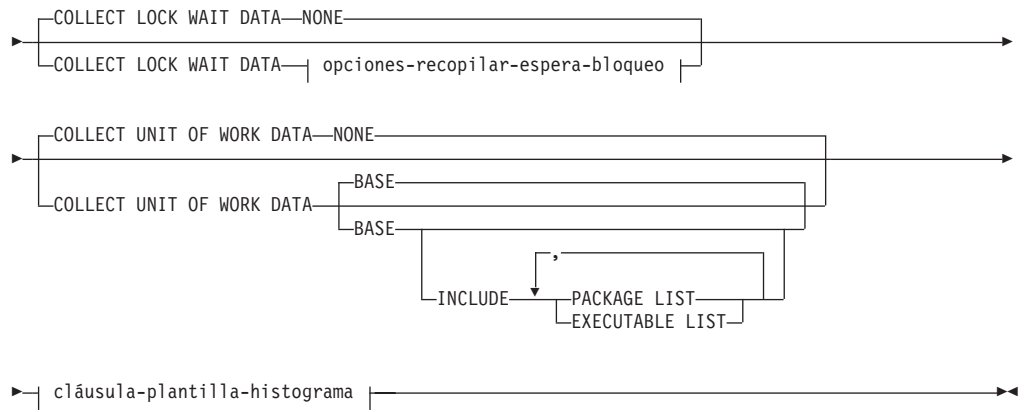
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

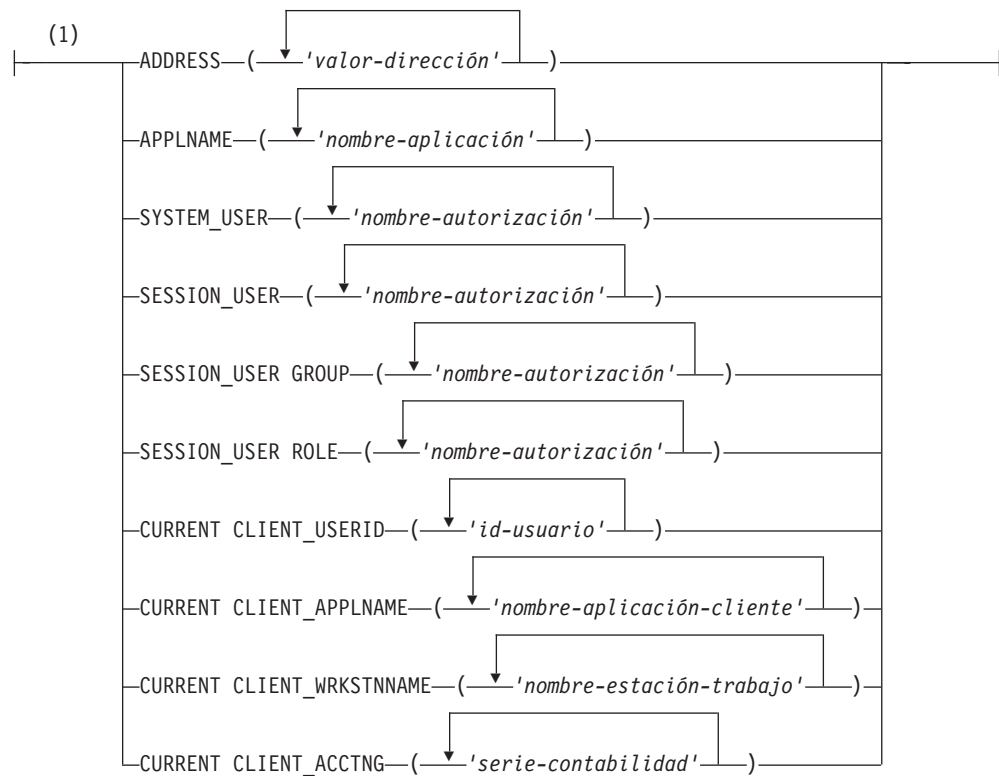
Sintaxis



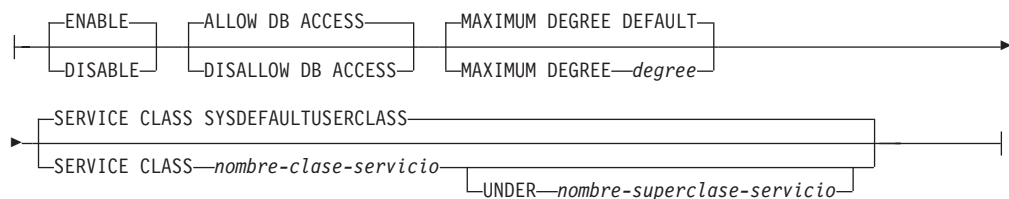
CREATE WORKLOAD



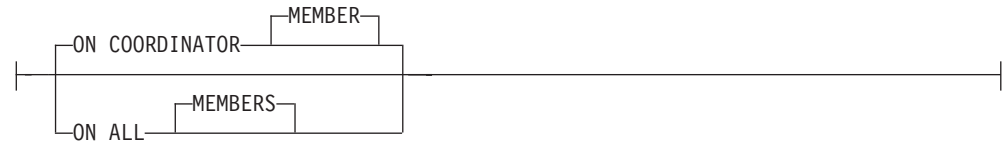
atributos-conexión:



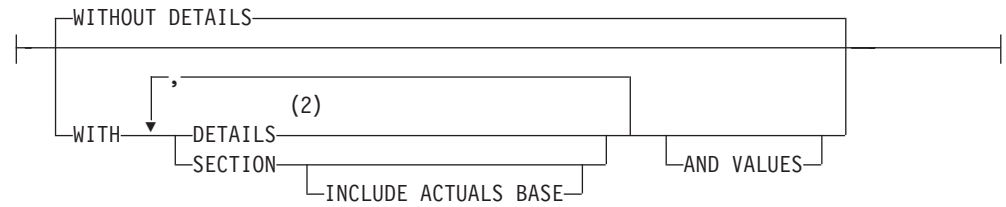
atributos-carga-trabajo:



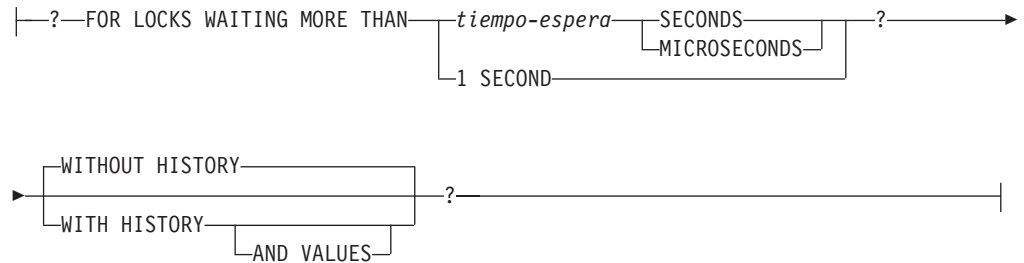
recopilar-en-cláusula:



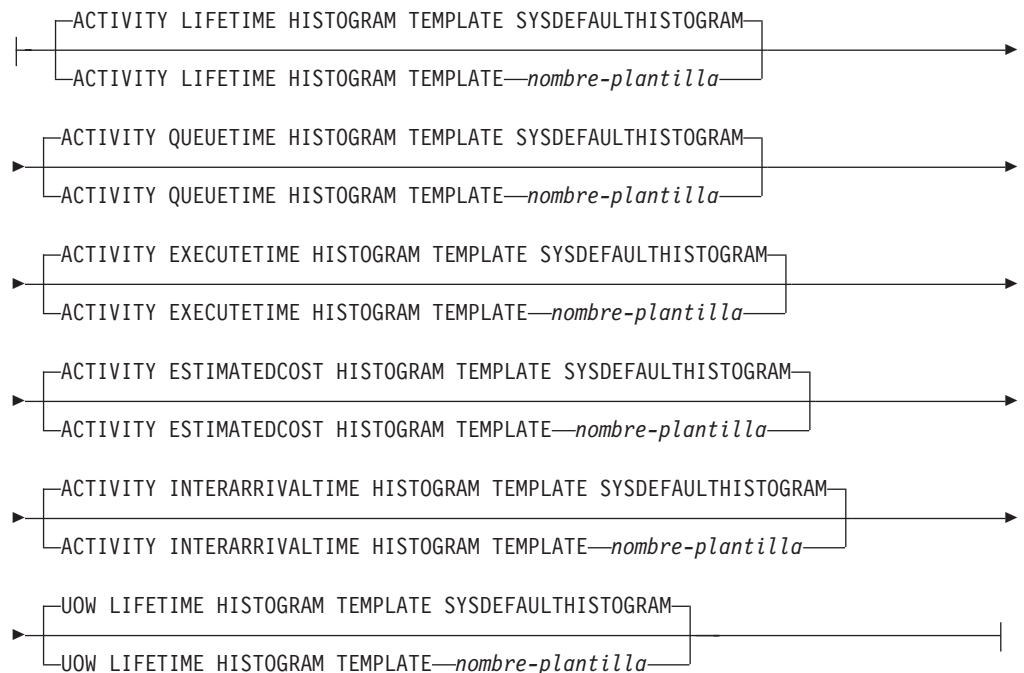
cláusula-recopilar-detalles:



opciones-recopilar-espera-bloqueo:



cláusula-plantilla-histograma:



CREATE WORKLOAD

Notas:

- 1 Cada cláusula de atributo de conexión sólo puede especificarse una vez.
- 2 La palabra clave DETAILS es la información mínima que debe especificarse, seguida de la opción separada por una coma.

Descripción

nombre-carga-trabajo

Da nombre a la carga de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-carga-trabajo* no debe identificar una carga de trabajo que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

atributos-conexión

Los atributos de la conexión deben corresponderse con todos los atributos especificados en esta definición de carga de trabajo si va a asociarse con esta carga de trabajo cuando se establezca la conexión. Si se especifica una lista de valores para un atributo de conexión de la definición de carga de trabajo, el atributo correspondiente de la conexión debe corresponderse con al menos uno de los valores de la lista. Si no se especifica un atributo de conexión en la definición de carga de trabajo, la conexión puede tener cualquier valor para el atributo de conexión correspondiente.

Nota: Todos los atributos de conexión distinguen entre mayúsculas y minúsculas, excepto ADDRESS.

ADDRESS ('valor-dirección', ...)

Especifica una o más direcciones IPv4, direcciones IPv6 o nombres de dominio seguros para el atributo de conexión ADDRESS. Un valor de dirección no puede aparecer más de una vez en la lista (SQLSTATE 42713). El valor de dirección debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 192.0.2.1. El valor localhost o su representación equivalente 127.0.0.1 no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal. Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.1) no darán como resultado una coincidencia. De modo análogo, localhost o su representación abreviada de IPv6 ::1 no dará como resultado una coincidencia. Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.example.com. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de carga de trabajo si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio.

Al crear un objeto de carga de trabajo, debe especificar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP

estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP) donde un dispositivo puede tener una dirección IP cada vez que se conecta con la red.

APPLNAME ('nombre-aplicación', ...)

Especifica una o más aplicaciones para el atributo de conexión de APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si el *nombre-aplicación* no contiene un solo carácter de asterisco (*), equivale al valor que se muestra en el campo “Nombre de la aplicación” en la salida del supervisor del sistema y en la salida del mandato LIST APPLICATIONS. Si *nombre-aplicación* no contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de nombres de aplicación, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la aplicación, utilice una secuencia de dos caracteres de asterisco (**).

SYSTEM_USER ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SYSTEM_USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER GROUP ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER GROUP. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ROLE ('nombre-autorización', ...)

Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER ROLE. Los roles de un ID de autorización de sesión en este contexto hacen referencia a todos los roles a disposición del ID de autorización de sesión, sin tener en cuenta el modo en que se obtuvieron los roles. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT_CLIENT_USERID ('id-usuario', ...)

Especifica uno o más ID de usuario cliente para el atributo de conexión CURRENT_CLIENT_USERID. Un ID de usuario cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *ID-usuario* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de ID de usuario, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el ID de usuario, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT_CLIENT_APPLNAME ('nombre-aplicación-cliente', ...)

Especifica una o más aplicaciones para el atributo de conexión de CURRENT_CLIENT_APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si el *nombre-aplicación-cliente* no contiene un solo carácter de asterisco (*), equivale al valor que se muestra en el campo “Nombre de aplicación cliente TP Monitor” en la salida del supervisor del sistema. Si *nombre-aplicación-cliente* no contiene un solo carácter de asterisco (*), el

CREATE WORKLOAD

valor se utiliza como expresión para representar un conjunto de nombres de aplicación, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la aplicación, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT CLIENT_WRKSTNNAME ('nombre-estación-trabajo', ...)

Especifica uno o más nombres de estación de trabajo cliente para el atributo de conexión CURRENT CLIENT_WRKSTNNAME. Un nombre de estación de trabajo cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *nombre-estación-trabajo* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de nombres de estación de trabajo, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en el nombre de la estación de trabajo, utilice una secuencia de dos caracteres de asterisco (**).

CURRENT CLIENT_ACCTNG ('serie-contabilidad', ...)

Especifica uno o más series de contabilidad cliente para el atributo de conexión CURRENT CLIENT_ACCTNG. Una serie de contabilidad cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713). Si *serie-contabilidad* contiene un solo carácter de asterisco (*), el valor se utiliza como expresión para representar un conjunto de series de contabilidad, donde el asterisco (*) representa una serie de cero o más caracteres. Si la expresión tiene que incluir un carácter de asterisco en la serie de contabilidad, utilice una secuencia de dos caracteres de asterisco (**).

atributos-carga-trabajo

Especifica los atributos de la carga de trabajo.

ENABLE o DISABLE

Especifica si esta carga de trabajo se tomará o no en consideración cuando se seleccione una carga de trabajo. El valor por omisión es ENABLE.

ENABLE

Especifica que la carga de trabajo se habilitará y se tomará en consideración cuando se seleccione una carga de trabajo.

DISABLE

Especifica que la carga de trabajo se inhabilitará y no se tomará en consideración cuando se seleccione una carga de trabajo.

ALLOW DB ACCESS o DISALLOW DB ACCESS

Especifica si se permitirá o no a la aparición de carga de trabajo asociada a esta carga de trabajo acceder a la base de datos. El valor por omisión es ALLOW DB ACCESS.

ALLOW DB ACCESS

Especifica que se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos.

DISALLOW DB ACCESS

Especifica que no se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos. La siguiente unidad de trabajo asociada a esta carga de trabajo se rechazará (SQLSTATE 5U020). Se permite que se completen las apariciones de carga de trabajo que ya se estén ejecutando.

MAXIMUM DEGREE

Especifica el grado de ejecución máximo de paralelismo para esta carga de trabajo. El valor por omisión es DEFAULT.

DEFAULT

Especifica que esta carga de trabajo hereda el valor de paralelismo intrapartición del parámetro de configuración del gestor de bases de datos **intra_parallel**. Cuando **intra_parallel** se establece en NO, la carga de trabajo se ejecuta con el paralelismo intrapartición inhabilitado. Cuando **intra_parallel** se establece en YES, la carga de trabajo se ejecuta con el paralelismo intrapartición habilitado. La carga de trabajo no especifica un grado de tiempo de ejecución máximo para las aplicaciones asignadas. Por tanto, el grado de tiempo de ejecución real se determina como el más bajo de los valores del parámetro de configuración **max_querydegree**, el valor establecido por el comando SET RUNTIME DEGREE y el grado de compilación de la sentencia de SQL.

grado

Especifica el grado máximo de paralelismo para esta carga de trabajo. Los valores válidos son 1 a 32.767. Con valor 1, las peticiones asociadas se ejecutan con el paralelismo intrapartición inhabilitado. Con un valor de 2 a 32.767, las peticiones asociadas se ejecutan con el paralelismo intrapartición habilitado. El grado de tiempo de ejecución real se determina como el más bajo de este *grado*, el valor del parámetro de configuración **max_querydegree**, el valor establecido por el comando SET RUNTIME DEGREE y el grado de compilación de la sentencia de SQL.

Nota: Un valor MAXIMUM DEGREE mayor que 1 no habilitará el paralelismo intrapartición a menos que la pila de clasificación compartida esté disponible.

SERVICE CLASS *nombre-clase-servicio*

Especifica que las peticiones asociadas a esta carga de trabajo se ejecuten en la clase de servicio *nombre-clase-servicio*. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-clase-servicio* no puede ser 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032). El valor por omisión es SYSDEFAULTUSERCLASS.

UNDER *nombre-superclase-servicio*

La cláusula se utiliza al especificar una subclase de servicio. El *nombre-superclase-servicio* identifica la superclase de servicio de *nombre-superclase-servicio*. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no puede ser 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032).

POSITION

Especifica el lugar en el que va a situarse esta carga de trabajo en la lista ordenada de cargas de trabajo. En tiempo de ejecución, se examina esta lista por orden para buscar la primera carga de trabajo que coincida con los atributos de conexión necesarios. El valor por omisión es LAST.

LAST

Especifica que la carga de trabajo va a ser la última de la lista, antes de las cargas de trabajo por omisión SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

BEFORE *nombre-carga-trabajo-relativa*

Especifica que la carga de trabajo va a colocarse antes de la carga de

CREATE WORKLOAD

trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción BEFORE no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AFTER *nombre-carga-trabajo-relativa*

Especifica que la carga de trabajo va a colocarse detrás de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción AFTER no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AT *posición*

Especifica la posición absoluta en la que la carga de trabajo va a colocarse en la lista. Este valor puede ser cualquier entero positivo (diferente de cero) (SQLSTATE 42615). Si *posición* es mayor que el número de cargas de trabajo existentes más una, la carga de trabajo se sitúa en la última posición justo antes de SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

COLLECT ACTIVITY METRICS

Especifica que la métrica del supervisor debe recopilarse para una actividad enviada por una ocurrencia de la carga de trabajo. El valor por omisión es COLLECT ACTIVITY METRICS NONE.

Nota: La configuración efectiva de la recopilación de métrica de actividad es la combinación del atributo especificado por la cláusula COLLECT ACTIVITY METRICS de la carga de trabajo que envía la petición y el parámetro de configuración de base de datos **mon_act_metrics**. Si ni el atributo de carga de trabajo ni el parámetro de configuración tienen un valor distinto de NONE, se recopilará la métrica para la actividad.

NONE

Especifica que no se recopilará ninguna métrica para cualquier actividad enviada por una ocurrencia de la carga de trabajo.

BASE

Especifica que la métrica básica se recopilará para cualquier actividad enviada por una ocurrencia de la carga de trabajo.

EXTENDED

Especifica que la métrica básica se recopilará para cualquier actividad enviada por una ocurrencia de la carga de trabajo. Asimismo, especifica que los valores de los siguientes elementos de supervisor deben determinarse con granularidad adicional:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

COLLECT ACTIVITY DATA

Especifica que los datos sobre cada actividad asociada a esta carga de trabajo se enviarán al supervisor de sucesos de actividades activas cuando la actividad finalice. El valor por omisión es COLLECT ACTIVITY DATA NONE.

recopilar-en-cláusula

Especifica el lugar en el que se recopilarán los datos de actividad. El valor por omisión es ON COORDINATOR MEMBER.

ON COORDINATOR MEMBER

Especifica que los datos de actividad se recopilan sólo en el miembro coordinador de la actividad.

ON ALL MEMBERS

Especifica que los datos de actividad van a recopilarse en todos los miembros en los que se procesa la actividad. En miembros remoto, es posible capturar un registro de la actividad varias veces, puesto que esta va y viene en dichos miembros. Si se especifica la cláusula AND VALUES, los valores de entrada de la actividad se recopilarán sólo para miembros del coordinador.

NONE

Especifica que los datos de actividad no se recopilan para cada actividad que se asocie con esta carga de trabajo.

cláusula-recopilar-detalles

Especifica el tipo de datos de actividad que se van a recopilar. El valor por omisión es WITHOUT DETAILS.

WITHOUT DETAILS

Especifica que los datos sobre cada actividad asociada con esta carga de trabajo se deben enviar a cualquier supervisor de actividades activas, cuando la actividad finalice su ejecución. Los detalles sobre la sentencia, el entorno de compilación y los datos del entorno de sección no se envían.

WITH**DETAILS**

Especifica que la sentencia y los datos del entorno de compilación deben enviarse a cualquier supervisor de sucesos de actividades activas para aquellas actividades que las tienen. Los datos del entorno de sección no se envían.

SECTION

Especifica que los datos de sentencia, de entorno de compilación, de entorno de sección y los datos reales de sección han de enviarse a cualquier supervisor de sucesos de actividades activo para aquellas actividades que incluyan estos. Se debe especificar DETAILS si se especifica SECTION. Los datos reales de sección sólo se recopilarán en cualquier miembro donde se recopilen datos de actividad.

INCLUDE ACTUALS BASE

Especifica que los datos reales de sección sólo se recopilarán en las particiones donde se recopilen datos de actividad. Para que se recopilen datos reales de sección, se debe especificar la cláusula INCLUDE ACTUALS o se debe establecer el parámetro de configuración de base de datos **section_actu**als.

La configuración efectiva para la recopilación de datos reales de sección es la combinación del valor de la cláusula INCLUDE ACTUALS, el parámetro de configuración de base de datos **section_actu**als y el valor <collectsectionactuals> especificado en la rutina WLM_SET_CONN_ENV. Por ejemplo, si se especifica INCLUDE ACTUALS BASE, pero el valor del

CREATE WORKLOAD

parámetro de configuración de la base de datos **section_actuals** es NONE y <collectsectionactuals> se establece en NONE, el valor efectivo de la colección de datos reales de la sección es BASE.

BASE especifica que lo siguiente debe habilitarse y recopilarse durante la ejecución de la actividad:

- Recuentos de cardinalidad de operador básico
- Estadísticas para cada objeto referenciado (sólo sentencias DML)

AND VALUES

Especifica que los valores de datos de entrada van a enviarse al supervisor de sucesos de actividades activas para las actividades que dispongan de los mismos.

COLLECT AGGREGATE ACTIVITY DATA

Especifica que los datos de actividades agregadas sobre las actividades asociadas con esta carga de trabajo deben enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. Cuando no se especifica COLLECT AGGREGATE ACTIVITY DATA, el valor por omisión es COLLECT AGGREGATE ACTIVITY DATA NONE. Cuando se especifica COLLECT AGGREGATE ACTIVITY DATA, el valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE.

BASE

Especifica que los datos de actividades agregadas básicas sobre las actividades asociadas con esta carga de trabajo se deben enviar al supervisor de sucesos estadísticos, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior de tiempo de CPU de actividad
- Histograma de tiempo de ejecución de la actividad
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo en cola de la actividad
- Marca de límite superior de lectura de filas de actividad
- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal. Sólo las actividades a las que se ha aplicado un umbral SQLTEMPSPACE participan en esta marca de límite superior.

EXTENDED

Especifica que todos los datos de actividades agregadas sobre las actividades asociadas con esta carga de trabajo deben enviarse al supervisor de sucesos estadísticos, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado del lenguaje de manipulación (DML) de datos de actividad
- Histograma de tiempo de llegada de DML de actividad

NONE

Especifica que no debe recopilarse ningún dato de actividad agregada para esta carga de trabajo.

COLLECT AGGREGATE UNIT OF WORK DATA

Especifica que se enviarán al supervisor de sucesos estadísticos, si hay alguno activo, los datos agregados de unidad de trabajo correspondientes a las unidades de trabajo asociadas a esta carga de trabajo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos `wlm_collect_int`. El valor por omisión cuando no se especifica **COLLECT AGGREGATE UNIT OF WORK DATA** es **COLLECT AGGREGATE UNIT OF WORK DATA NONE**.

BASE

Especifica que se enviarán al supervisor de sucesos estadísticos, si hay alguno activo, los datos agregados básicos de unidad de trabajo correspondientes a las unidades de trabajo asociadas a esta carga de trabajo. Los datos agregados básicos de unidad de trabajo incluyen:

- Unidad de histograma de ciclo de vida de trabajo

NONE

Especifica que no debe recopilarse ningún dato agregado de unidad de trabajo para esta carga de trabajo.

COLLECT LOCK TIMEOUT DATA

Especifica que los datos sobre los sucesos de tiempo de espera de bloqueo que se produzcan en esta carga de trabajo se envían al supervisor de sucesos aplicable cuando se produzca el suceso de bloqueo. Los datos de tiempo de espera de bloqueo se recopilan en todos los miembros. El valor por omisión es **COLLECT LOCK TIMEOUT DATA WITHOUT HISTORY**. Este valor funciona conjuntamente con el valor del parámetro de configuración de base de datos `mon_locktimeout`. Se respeta la configuración que genera la salida más detallada.

WITHOUT HISTORY

Especifica que los datos sobre los sucesos de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. El historial de actividades anteriores y los valores de entrada no se envían al supervisor de sucesos.

NONE

Especifica que los datos de tiempo de espera de bloqueo para la carga de trabajo no se recopilan en ningún miembro.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos los sucesos de bloqueo de este tipo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro `DB2_MAX_INACT_STMTS` para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB,

CREATE WORKLOAD

datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

COLLECT DEADLOCK DATA

Especifica que los datos sobre los sucesos de punto muerto que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. Los datos de tiempo de espera de bloqueo se recopilan en todos los miembros. El valor por omisión es COLLECT DEADLOCK DATA WITHOUT HISTORY. Este valor sólo se respetará si el parámetro de configuración de base de datos **mon_deadlock** está establecido en NONE.

WITHOUT HISTORY

Especifica que los datos sobre los sucesos de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando se produzca el suceso de bloqueo. El historial de actividades anteriores y los valores de entrada no se envían al supervisor de sucesos.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos los sucesos de bloqueo de este tipo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro DB2_MAX_INACT_STMTS para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB, datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

COLLECT LOCK WAIT DATA

Especifica que los datos sobre los sucesos de espera de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando el bloqueo no se haya adquirido en el *tiempo-espera*. El valor por omisión es COLLECT LOCK WAIT DATA NONE con un valor de *tiempo-espera* por omisión de 0 microsegundos. Este valor funciona conjuntamente con el valor del parámetro de configuración de base de datos **mon_lockwait** y **mon_lw_thresh**. Se respeta la configuración que genera la salida más detallada.

NONE

Especifica que el suceso de espera de bloqueo para la carga de trabajo no se recopila en ningún miembro.

FOR LOCKS WAITING MORE THAN *tiempo-espera* (SECONDS | MICROSECONDS) | 1 SECOND

Especifica que los datos sobre los sucesos de espera de bloqueo que se produzcan en esta carga de trabajo se envían a cualquier supervisor de sucesos de bloqueo activo cuando el bloqueo no se haya adquirido en el *tiempo-espera*.

Este valor puede ser cualquier entero no negativo. Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *tiempo-espera*. El valor válido mínimo para el parámetro *tiempo-espera* es 1000 microsegundos.

WITH HISTORY

Especifica que se recopilará el historial de actividades anteriores en la unidad de trabajo actual para todos los sucesos de bloqueo de este tipo. El almacenamiento intermedio del historial de actividades se derivará una vez utilizado el límite de tamaño máximo.

El límite por omisión en el número de actividades anteriores que debe mantener una aplicación es 250. Si el número de actividades anteriores es mayor que el límite, sólo se notifican las actividades más recientes. Este valor por omisión puede alterarse temporalmente utilizando la variable de registro DB2_MAX_INACT_STMTS para especificar otro valor. Puede elegir otro valor para el límite a fin de aumentar o reducir la cantidad de almacenamiento dinámico de supervisión del sistema que se utiliza para la información de actividades anteriores.

AND VALUES

Especifica que se envíen los valores de datos de entrada a cualquier supervisor de sucesos de bloqueo activo para las actividades que dispongan de ellos. Estos valores de datos no incluirán datos LOB, datos LONG VARCHAR, datos LONG VARGRAPHIC, datos de tipo estructurado o datos XML. Para sentencias de SQL compilado mediante la opción de vinculación REOPT ALWAYS, no se proporcionará ningún valor de datos de compilación de REOPT o ejecución de sentencia en la información de sucesos.

COLLECT UNIT OF WORK DATA

Especifica que los datos sobre cada transacción asociada con esta carga de trabajo deben enviarse a la unidad del supervisor de sucesos de trabajo, si hay alguno activo, cuando finalice la unidad de trabajo. El valor por omisión cuando no se especifica COLLECT UNIT OF WORK DATA es COLLECT UNIT OF WORK DATA NONE. El valor por omisión cuando se especifica COLLECT UNIT OF WORK DATA es COLLECT UNIT OF WORK DATA BASE. Si el parámetro de configuración de base de datos **mon_uow_data** se establece en BASE, tendrá prioridad sobre el parámetro COLLECT UNIT OF WORK DATA. Un valor NONE para **mon_uow_data** indica que se utilizan los parámetros COLLECT UNIT OF WORK DATA de cargas de trabajo individuales.

NONE

Especifica que ninguna unidad de datos de trabajo para las transacciones asociadas con esta carga de trabajo se envía al supervisor de sucesos de unidad de trabajo. El valor por omisión es COLLECT UNIT OF WORK DATA NONE.

CREATE WORKLOAD

BASE

Especifica que se envíe el nivel básico de datos para las transacciones asociadas con esta carga de trabajo al supervisor de sucesos de unidad de trabajo.

Parte de la información notificada en un suceso de unidad de trabajo corresponde a métricas de petición de nivel del sistema. La recopilación de estas métricas se controla independientemente de la recopilación de datos de la unidad de trabajo. Las métricas de petición se controlan con la cláusula COLLECT REQUEST METRICS de la superclase o bien mediante la utilización del parámetro de configuración de base de datos **mon_req_metrics**. La superclase de servicio con la que está asociada la carga de trabajo, o la superclase de servicio de la subclase de servicio con la que está asociada la carga de trabajo, debe tener habilitada la recopilación de métricas de petición para que las métricas de petición estén presentes en el suceso de unidad de trabajo. Si la recopilación de métricas de petición no está habilitada, el valor de las métricas de petición será cero.

INCLUDE PACKAGE LIST

Especifica que han de enviarse el nivel básico de datos y la lista de paquetes de las transacciones que se asocian a esta carga de trabajo al supervisor de sucesos de unidad de trabajo.

El tamaño de la lista de paquetes recopilados lo determina el valor del parámetro de configuración de base de datos **mon_pkglist_sz**. Si este valor es 0, la lista de paquetes no se recopilará, aunque se haya especificado la opción PACKAGE LIST.

En un entorno de base de datos particionada, la lista de paquetes sólo está disponible en el miembro coordinador. El nivel BASE se recopilará en los miembros remotos.

Parte de la información notificada en un suceso de unidad de trabajo corresponde a métricas de petición de nivel del sistema. La recopilación de estas métricas se controla independientemente de la recopilación de datos de la unidad de trabajo. Las métricas de petición se controlan con la cláusula COLLECT REQUEST METRICS de la superclase o bien mediante la utilización del parámetro de configuración de base de datos **mon_req_metrics**. La superclase de servicio con la que está asociada la carga de trabajo, o la superclase de servicio de la subclase de servicio con la que está asociada la carga de trabajo, debe tener habilitada la recopilación de métricas de petición para que las métricas de petición estén presentes en el suceso de unidad de trabajo. Si la recopilación de métricas de petición no está habilitada, el valor de las métricas de petición será cero.

INCLUDE EXECUTABLE LIST

Especifica que se recopilará la lista de ID ejecutables para una unidad de trabajo junto con el nivel básico de datos y se enviarán al supervisor de sucesos de la unidad de trabajo.

cláusula-plantilla-histograma

Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades que se ejecutan en la carga de trabajo.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en milisegundos, de las actividades de DB2 en ejecución en la carga de trabajo durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de

ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la carga de trabajo están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, en que las actividades de DB2 en ejecución en la carga de trabajo están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en el miembro coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECTAGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML en ejecución en la carga de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en milisegundos, entre la llegada de una actividad DML a esta carga de trabajo y la llegada de la siguiente actividad DML a esta carga de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

UOW LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*

Especifica la plantilla que describe el histograma que se utiliza para recopilar datos estadísticos sobre la duración, en milisegundos, de unidades de trabajo que se ejecutan en la carga de trabajo durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE UNIT OF WORK DATA con la opción BASE.

CREATE WORKLOAD

Normas

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de WLM son:
 - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
 - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
 - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
 - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
 - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Cuando se establece una conexión de base de datos, el gestor de base de datos busca una carga de trabajo coincidente basada en los atributos de conexión especificados en la cláusula POSITION (por orden de especificación). Si se encuentra una carga de trabajo coincidente, el gestor de base de datos compruebe si el usuario de sesión actual tiene privilegio de USAGE en dicha carga de trabajo. Si el usuario de sesión no tiene privilegio de USAGE en la carga de trabajo, el gestor de base de datos busca la siguiente carga de trabajo coincidente. Si el usuario de sesión tiene privilegio de USAGE en esta carga de trabajo, la conexión se asociará a la carga de trabajo. Si no se encuentra una carga de trabajo coincidente, la conexión se asocia a la carga de trabajo de usuario por omisión, SYSDEFAULTUSERWORKLOAD. Si el usuario de sesión no tiene privilegio de USAGE en SYSDEFAULTUSERWORKLOAD, se devuelve un error (SQLSTATE 42501).
- La asociación de carga de trabajo vuelve a evaluarse al principio de cada unidad de trabajo nueva si el gestor de base de datos detecta una de las siguientes condiciones:
 - Han cambiado los atributos de conexión. Esto puede suceder si se han producido alguno de los sucesos siguientes:
 - Se ha invocado la API de información de cliente establecido (sqleseti) y se ha cambiado los atributos de conexión incluidos en la definición de carga de trabajo. Tenga en cuenta que aunque la información de cliente puede establecerse por medio del usuario final de modo que pueda iniciarse una reevaluación de carga de trabajo, la propia acción de correlación de carga de trabajo podría producirse si el usuario de sesión no tiene el privilegio de USAGE en la carga de trabajo.

- La sentencia SET SESSION AUTHORIZATION se ha invocado y ha cambiado el usuario de sesión actual.
- Han cambiado los roles que están disponibles para un usuario de sesión.
- Se crea una carga de trabajo.
- Se descarta una carga de trabajo.
- Se modifica una carga de trabajo.
- El privilegio de USAGE de una carga de trabajo se otorga a un usuario, grupo o rol.
- El privilegio de USAGE de una carga de trabajo se revoca de un usuario, grupo o rol.

Si la reevaluación de la carga de trabajo da como resultado que no se reasigne ninguna carga de trabajo, la aparición de carga de trabajo actual sigue ejecutándose; es decir, no se iniciará una aparición de carga de trabajo nueva.

- No puede reasignarse una conexión a una carga de trabajo diferente cuando una actividad siga activa. Una operación de carga, un procedimiento de ejecución o las sentencias que mantengan recursos entre varias unidades de trabajo, por ejemplo un cursor de WITH HOLD de apertura son ejemplos de tales actividades. La aparición de carga de trabajo actual sigue ejecutándose hasta que finalizan todas las actividades en ejecución. La reasignación de la carga de trabajo se produce al principio de la siguiente unidad de trabajo.
- Una vez que se haya hecho referencia a una clase de servicio por medio de una carga de trabajo, no podrá descartarse hasta que ya no se haga referencia al mismo por medio de ninguna carga de trabajo. Para eliminar una referencia de clase de servicio desde una carga de trabajo puede adoptarse cualquiera de las acciones siguientes:
 - Modificar la carga de trabajo para cambiar el nombre de clase de servicio
 - Descartar la carga de trabajo
- Una vez que se haya hecho referencia a un rol por medio de una carga de trabajo, no podrá descartarse hasta que ya no se haga referencia al mismo por medio de ninguna carga de trabajo. Para eliminar una referencia de rol desde una carga de trabajo puede adoptarse cualquiera de las acciones siguientes:
 - Modificar la carga de trabajo para eliminar el rol
 - Descartar la carga de trabajo
- **Privilegios:** el privilegio USAGE no se otorga a ningún usuario, grupo o rol cuando se crea una carga de trabajo. Para habilitar el uso de una carga de trabajo, otorgue el privilegio USAGE de esa carga de trabajo a un usuario, grupo o rol mediante la sentencia GRANT USAGE ON WORKLOAD.
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - Puede especificarse DATABASE PARTITION en lugar de MEMBER, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse DATABASE PARTITIONS en lugar de MEMBERS, salvo cuando la variable de registro DB2_ENFORCE_MEMBER_SYNTAX se establece en ON.
 - Puede especificarse COLLECT UNIT OF WORK DATA PACKAGE LIST en lugar de COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST.

CREATE WORKLOAD

Ejemplos

- *Ejemplo 1:* Cree una carga de trabajo denominada CAMPAIGN para las peticiones que envía un usuario de sesión que pertenece al grupo FINANCE. Estas peticiones van a ejecutarse en la clase de servicio de usuario por omisión SYSDEFAULTUSERCLASS.

```
CREATE WORKLOAD CAMPAIGN
SESSION_USER GROUP ('FINANCE')
```

- *Ejemplo 2:* Cree una carga de trabajo denominada PAYROLL para un usuario de sesión con el rol HR que tenga el registro especial CURRENT CLIENT_APPLNAME establecido en SALARYSYS. Las unidades de trabajo asociadas a esta carga de trabajo van a ejecutarse en la clase de servicio MEDIUMSC que está bajo la superclase de servicio HRSC. Cuando se selecciona una carga de trabajo en tiempo de ejecución, esta carga de trabajo debería evaluarse únicamente después de que se haya evaluado la carga de trabajo CAMPAIGN y se haya determinado que no coinciden.

```
CREATE WORKLOAD PAYROLL
SESSION_USER ROLE ('HR')
CURRENT_CLIENT_APPLNAME ('SALARYSYS') SERVICE CLASS MEDIUMSC
UNDER HRSC POSITION AFTER CAMPAIGN
```

- *Ejemplo 3:* Una aparición de la carga de trabajo CAMPAIGN (del ejemplo 1) se está ejecutando en la actualidad en el sistema. Cree una carga de trabajo denominada NEWCAMPAIGN, también para las peticiones que envía un usuario de sesión que pertenezca al grupo FINANCE, pero sólo para las peticiones que se envían por medio de la aplicación DB2BP.EXE. Las peticiones asociadas a esta carga de trabajo van a ejecutarse en la clase de servicio MARKETINGSC. NEWCAMPAIGN debería evaluarse antes de CAMPAIGN.

```
CREATE WORKLOAD NEWCAMPAIGN
SESSION_USER GROUP ('FINANCE')
APPLNAME ('DB2BP.EXE') SERVICE CLASS MARKETINGSC
POSITION BEFORE CAMPAIGN
```

La aparición de carga de trabajo en ejecución de CAMPAIGN continúa ejecutándose hasta que se complete la unidad de trabajo actual, en cuyo momento tiene lugar una reevaluación de carga de trabajo y a continuación, la conexión puede volver a correlacionarse con la carga de trabajo NEWCAMPAIGN.

- *Ejemplo 4:* Cree una carga de trabajo denominada REPORTS para las peticiones sometidas por medio de la aplicación appl1, appl2 o appl3 por medio del usuario del sistema BOB o MARY.

```
CREATE WORKLOAD REPORTS
APPLNAME ('appl1', 'appl2', 'appl3')
SYSTEM_USER ('BOB', 'MARY')
```

- *Ejemplo 5:* suponiendo que existe un supervisor de sucesos de bloqueo denominado PAYROLL y que está activo, crear registros de sucesos de bloqueo con un historial de sentencias para sucesos de tiempo de espera de bloqueo que tengan lugar dentro de la carga de trabajo EMPLOYEEES.

```
CREATE WORKLOAD EMPLOYEEES
APPLNAME ("app1", "app2")
COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

- *Ejemplo 6:* suponiendo que existe un supervisor de sucesos de bloqueo denominado PAYROLL y que está activo, crear registros de sucesos de bloqueo sólo para sucesos de tiempo de espera de punto muerto y bloqueo que tengan lugar en la carga de trabajo FINANCE en todas las particiones.


```
CREATE WORKLOAD FINANCE
  APPLNAME ("app1", "app2")
  COLLECT DEADLOCK DATA
  COLLECT LOCK TIMEOUT DATA
```

- *Ejemplo 7:* suponiendo que existe un supervisor de sucesos de bloqueo denominado PAYROLL y que está activo, crear registros de sucesos de bloqueo con un historial de sentencias para sucesos de punto muerto que tengan lugar dentro de la carga de trabajo MANAGERS.

```
CREATE WORKLOAD MANAGERS
  APPLNAME ("app1", "app2")
  COLLECT DEADLOCK DATA WITH HISTORY AND VALUES
```

- *Ejemplo 8:* suponiendo que existe un supervisor de sucesos de bloqueo denominado PAYROLL y que está activo, crear registros de sucesos de bloqueo con un historial de sentencias para bloqueos que se adquieran tras esperar 5000 milisegundos.

```
CREATE WORKLOAD MANAGERS
  APPLNAME ("app1", "app2")
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 5 SECONDS WITH HISTORY
```

- *Ejemplo 9:* crear una carga de trabajo denominada ACCRECS para todas las aplicaciones de cuentas por cobrar que compartan un nombre similar (*accrec01, accrec02 ... accrec15*) y asignarlas a la clase de servicio ACCOUNTNGSC. Los nombres de aplicación se identifican a través del atributo de conexión APPLNAME con la ayuda de un comodín (*) y no tienen que especificarse de forma individual.

```
CREATE WORKLOAD ACCRECS
  SESSION_USER_GROUP ('ACCOUNTING')
  APPLNAME ('accrec*')
  SERVICE CLASS ACCOUNTNGSC
```

- *Ejemplo 10:* crear una carga de trabajo denominada CAMPAIGN para las peticiones enviadas a través de la aplicación appl1 y hacer que se recopilen y envíen los datos de unidad de trabajo a cualquier supervisor de sucesos de unidad de trabajo.

```
CREATE WORKLOAD CAMPAIGN
  APPLNAME ('appl1')
  COLLECT UNIT OF WORK DATA BASE
```

- *Ejemplo 11:* las sentencias siguientes muestran cómo se pueden especificar los distintos formatos de valor de dirección soportados por el atributo de conexión ADDRESS al crear una carga de trabajo.

- Para especificar un nombre de dominio seguro:

```
CREATE WORKLOAD DOMAINWORKLOAD
  ADDRESS ('aviator.example.com')
```

- Para especificar un valor de dirección IPv4:

```
CREATE WORKLOAD IPWORKLOAD1
  ADDRESS ('192.0.2.11')
```

- Para especificar un valor de dirección IPv6 (formato largo):

```
CREATE WORKLOAD IPWORKLOAD2
  ADDRESS ('2001:db8:519:13:204:acff:fe57:6135')
```

- Para especificar un valor de dirección IPv6 (formato corto):

```
CREATE WORKLOAD IPWORKLOAD3
  ADDRESS ('2001:db8::202:55ff:fe9a:6eee')
```

CREATE WRAPPER

La sentencia CREATE WRAPPER registra un derivador en un servidor federado. Un derivador es un mecanismo mediante el cual un servidor federado puede interactuar con determinados tipos de fuentes de datos.

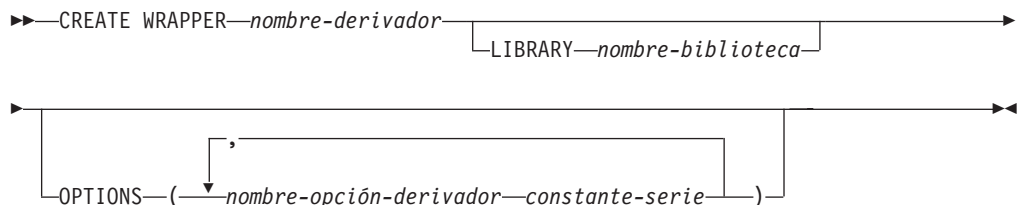
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM.

Sintaxis



Descripción

nombre-derivador

Nombra el derivador. Puede ser:

- Un nombre predefinido. Si se especifica un nombre predefinido, el servidor federado asigna automáticamente un valor por omisión a *nombre-biblioteca*.
- Un nombre suministrado por el usuario. Si se proporciona un nombre suministrado por el usuario, también se debe especificar el *nombre-biblioteca* adecuado que se debe utilizar con ese derivador y sistema operativo.

LIBRARY *nombre-biblioteca*

Nombra el archivo que contiene el módulo de biblioteca del derivador.

El nombre de biblioteca puede especificarse como un nombre de vía de acceso absoluta o, simplemente, puede especificarse el nombre base (sin la vía de acceso). Si sólo se especifica el nombre base, la biblioteca debe residir en el subdirectorio `lib` (UNIX) o en el subdirectorio `bin` (Windows) de la vía de acceso de DB2. El *nombre-biblioteca* se debe especificar entre comillas simples.

La opción LIBRARY sólo es necesaria si se utiliza un *nombre-derivador* proporcionado por el usuario. Esta opción no debe utilizarse cuando se proporciona un *nombre-derivador* predefinido.

OPTIONS

Indica las opciones de derivador que se deben habilitar o restablecer.

nombre-opción-derivador

Nombra una opción de derivador que se debe habilitar o restaurar.

constante-serie

Especifica el valor para *nombre-opción-derivador* como una constante de serie de caracteres. La *constante-serie* se debe especificar entre comillas simples. Algunas opciones de derivador pueden utilizarlas todos los derivadores y otras son específicas de un derivador en particular.

Notas

- **Alternativas de sintaxis:** se soporta la sintaxis siguiente para mantener la compatibilidad con versiones anteriores de DB2:
 - ADD puede especificarse antes de *nombre-opción-derivador constante-serie*.

Ejemplos

- *Ejemplo 1:* Registre el derivador NET8 en un servidor federado para acceder a fuentes de datos Oracle. *NET8* es el nombre predefinido para el derivador que puede utilizar para acceder a las fuentes de datos Oracle.

```
CREATE WRAPPER NET8
```

- *Ejemplo 2:* Registre un derivador en un servidor federado DB2 que utiliza el sistema operativo Linux para acceder a fuentes de datos ODBC. Asigne el nombre *odbc* al derivador que se registra en la base de datos federada. La vía de acceso completa de la biblioteca que contiene el gestor de controladores ODBC se define en la opción de derivador *MODULE '/usr/lib/odbc.so'*.

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

- *Ejemplo 3:* Registre un derivador en un servidor federado DB2 que utiliza el sistema operativo Windows para acceder a fuentes de datos ODBC. El nombre de biblioteca para el derivador de ODBC es *'db2rcodbc.dll'*.

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

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 incorporarse dentro de un programa de aplicación. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica. Cuando se invoca utilizando el procesador de línea de mandatos, se pueden especificar opciones adicionales. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

Autorización

El término "sentencia SELECT del cursor" se utiliza para especificar las normas de autorización. La sentencia SELECT del cursor es una de las sentencias siguientes:

- La sentencia-select preparada que se identifica mediante el *nombre-sentencia*
- La *sentencia-select* especificada

Los privilegios que tiene el ID de autorización de la sentencia deben incluir los privilegios necesarios para ejecutar la *sentencia-select*. Consulte la sección Autorización en "Consultas de SQL".

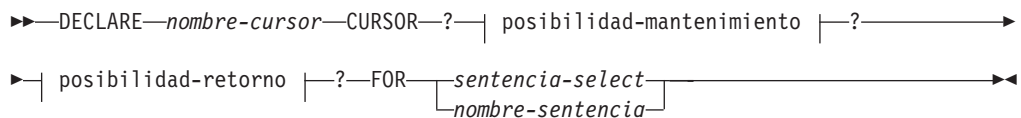
Si se especifica el *nombre-sentencia*:

- El ID de autorización de la sentencia es el ID de autorización de ejecución.
- La comprobación de autorización se lleva a cabo cuando se prepara la sentencia SELECT.
- El cursor no puede abrirse si la sentencia SELECT no está en estado preparado.

Si se especifica la *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



posibilidad de mantenimiento:



posibilidad de retorno:



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. El cursor deberá abrirse para poder utilizarlo.

WITHOUT HOLD o WITH HOLD

Especifica si se debe impedir que el cursor se cierre como consecuencia de una operación de confirmación.

WITHOUT HOLD

No impide que el cursor se cierre como consecuencia de una operación de confirmación. Es el valor por omisión.

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 de resultados.

Si se emite la sentencia DISCONNECT después de la sentencia COMMIT para una conexión con cursores WITH HOLD, los cursores retenidos 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 retenidos 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 de SQL dinámico (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.
 - El conjunto de filas modificado por:
 - Una sentencia de cambio de datos
 - Rutinas que modifican datos SQL incorporados dentro de cursores WITH HOLD abiertos

se confirma.
- Para las unidades de trabajo que finalizan con ROLLBACK:

DECLARE CURSOR

- Se cierran todos los cursores abiertos.
- Se liberan todos los bloqueos adquiridos durante la unidad de trabajo.
- Se liberan los localizadores de LOB.
- Para el caso especial de COMMIT:
 - Los paquetes se pueden volver a crear explícitamente, vinculándolos, o implícitamente, porque se han invalidado y, a continuación, se han vuelto a crear dinámicamente la primera vez que se ha hecho referencia a ellos. Todos los cursores retenidos se cierran cuando se vuelve a enlazar el paquete. Esto puede producir errores en una posterior ejecución.

WITHOUT RETURN o WITH RETURN

Especifica si la tabla de resultados del cursor está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento.

WITHOUT RETURN

Especifica que la tabla de resultados del cursor no está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento.

WITH RETURN

Especifica que la tabla de resultados del cursor está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento. WITH RETURN sólo es relevante si la sentencia DECLARE CURSOR está incluida junto con el código fuente de un procedimiento. En los demás casos, es posible que el precompilador acepte la cláusula, pero esto no tiene ningún efecto.

Dentro de un procedimiento de SQL, los cursores declarados mediante la cláusula WITH RETURN que todavía están abiertos cuando finaliza el procedimiento de SQL, definen los conjuntos de resultados del procedimiento de SQL. Todos los demás cursores abiertos de un procedimiento de SQL se cierran cuando finaliza el procedimiento de SQL. Dentro de un procedimiento externo (un procedimiento no definido utilizando LANGUAGE SQL), el valor por omisión para todos los cursores es WITH RETURN TO CALLER. Por lo tanto, todos los cursores que estén abiertos cuando el procedimiento finalice se considerarán conjuntos de resultados. Los cursores que se devuelven de un procedimiento no se pueden declarar como cursores desplazables.

TO CALLER

Especifica que el cursor puede devolver un conjunto de resultados al llamador. Por ejemplo, si el llamador es otro procedimiento, el conjunto de resultados se devuelve a ese procedimiento. Si el llamador es una aplicación cliente, el conjunto de resultados se devuelve a la aplicación cliente.

TO CLIENT

Especifica que el cursor puede devolver un conjunto de resultados a la aplicación cliente. Este cursor es invisible para cualquier procedimiento anidado intermedio. Si una función, un método o un activador ha llamado al procedimiento directa o indirectamente, no se podrán devolver conjuntos de resultados al cliente y el cursor se cerrará tras completarse el procedimiento.

sentencia-select

Identifica la sentencia SELECT del cursor. La *sentencia-select* no debe incluir marcadores de parámetro, pero puede incluir referencias a las variables del

lenguaje principal. Las declaraciones de variables del lenguaje principal deben preceder a la sentencia DECLARE CURSOR en el programa fuente.

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.

Para obtener información acerca de las sentencias SELECT preparadas, consulte "PREPARE".

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 no anidados que no estén definidos con LANGUAGE SQL tendrán WITH RETURN TO CALLER como comportamiento por omisión si se especifica DECLARE CURSOR sin una cláusula WITH RETURN y el cursor se deja abierto en el procedimiento. Esto proporciona compatibilidad con procedimientos de versiones anteriores, en las cuales los procedimientos pueden devolver conjuntos de resultados 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 "datetime" (de fecha y hora) respectivo 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 de resultados 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 sólo identifica una 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.
 - La selección completa externa no incluye operaciones SET (UNION, EXCEPT o INTERSECT) a excepción de UNION ALL.
 - La selección completa externa no contiene una especificación de período FOR SYSTEM_TIME.
 - La lista de selección de la selección completa externa no incluye DISTINCT.

DECLARE CURSOR

- La selección completa externa no incluye una cláusula ORDER BY (aunque la cláusula ORDER BY esté anidada en una vista) y no se ha especificado la cláusula FOR UPDATE.
- La sentencia select no incluye una cláusula FOR READ ONLY.
- La cláusula FROM de la selección completa externa no incluye una *referencia-tabla-cambio-datos*.
- Se dan una o más de las condiciones siguientes:
 - Se ha especificado la cláusula FOR UPDATE.
 - El cursor se ha definido de forma estática, a menos que la opción de vinculación STATICREADONLY sea YES.
 - La opción de vinculación LANGLEVEL es MIA o SQL92E.

Una columna de la lista de selección de la selección completa externa asociada a un cursor es *actualizable* si se cumplen todas las condiciones siguientes:

- El cursor es suprimible.
- La columna se devuelve en una columna de la tabla base.
- La opción de vinculación LANGLEVEL es MIA o SQL92E, o la sentencia select incluye la cláusula FOR UPDATE (la columna debe haberse especificado 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 ha preparado de forma dinámica.
- La sentencia select no incluye la cláusula FOR READ ONLY ni la cláusula FOR UPDATE.
- La opción de vinculación LANGLEVEL es SAA1.
- De otro modo, el cursor cumple 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 actualizable.

- Los cursores de procedimientos a los que llaman programas de aplicación que se han escrito mediante la CLI pueden utilizarse para definir conjuntos de resultados que se devuelvan directamente a la aplicación cliente. Los cursores de procedimientos SQL también pueden devolver resultados a un procedimiento de SQL llamador sólo si están definidos utilizando la cláusula WITH RETURN.
- Los cursores que se han declarado en rutinas que se invocan directa o indirectamente desde un cursor declarado como WITH HOLD, no heredan la opción WITH HOLD. De esta forma, a menos que el cursor de la rutina se haya definido explícitamente como WITH HOLD, una ejecución de COMMIT en la aplicación hará que se cierre.

Consideremos la aplicación y los dos UDF siguientes:

Aplicación:

```
DECLARE APPCUR CURSOR WITH HOLD FOR SELECT UDF1() ...
OPEN APPCUR
FETCH APPCUR ...
COMMIT
```

UDF1:

```
DECLARE UDF1CUR CURSOR FOR SELECT UDF2() ...
OPEN UDF1CUR
FETCH UDF1CUR ...
```

UDF2:


```
DECLARE UDF2CUR CURSOR WITH HOLD FOR SELECT UDF2() ...  
OPEN UDF2CUR  
FETCH UDF2CUR ...
```

Después de que la aplicación haya buscado el cursor APPCUR, se abrirán los tres cursores. Cuando la aplicación emite la sentencia COMMIT, APPCUR sigue abierto, ya que se ha declarado como WITH HOLD. Sin embargo, en UDF1, el cursor UDF1CUR está cerrado, ya que no se ha definido con la opción WITH HOLD. Cuando se cierra el cursor UDF1CUR, se completan todas las invocaciones de rutina de la sentencia-select correspondiente (recibiendo una llamada final, si así se ha definido). UDF2 se completa, lo que da lugar a que UDF2CUR se cierre.

Ejemplos

Ejemplo 1: La sentencia DECLARE CURSOR asocia el nombre del cursor C1 con los resultados de la sentencia SELECT.

```
EXEC SQL DECLARE C1 CURSOR FOR  
SELECT DEPTNO, DEPTNAME, MGRNO  
FROM DEPARTMENT  
WHERE ADMRDEPT = 'A00';
```

Ejemplo 2: Supongamos que la tabla EMPLOYEE se ha modificado para añadir una columna generada, WEEKLYPAY, que calcula la paga semanal basada en el salario anual. Declare un cursor para recuperar el valor de la columna generada por el sistema de una fila que se va a insertar.

```
EXEC SQL DECLARE C2 CURSOR FOR  
SELECT E.WEEKLYPAY  
FROM NEW TABLE  
(INSERT INTO EMPLOYEE  
(EMPNO, FIRSTNAME, MIDINIT, LASTNAME, EDLEVEL, SALARY)  
VALUES('000420', 'Peter', 'U', 'Bender', 16, 31842) AS E;
```

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 descarta la descripción de la tabla temporal.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

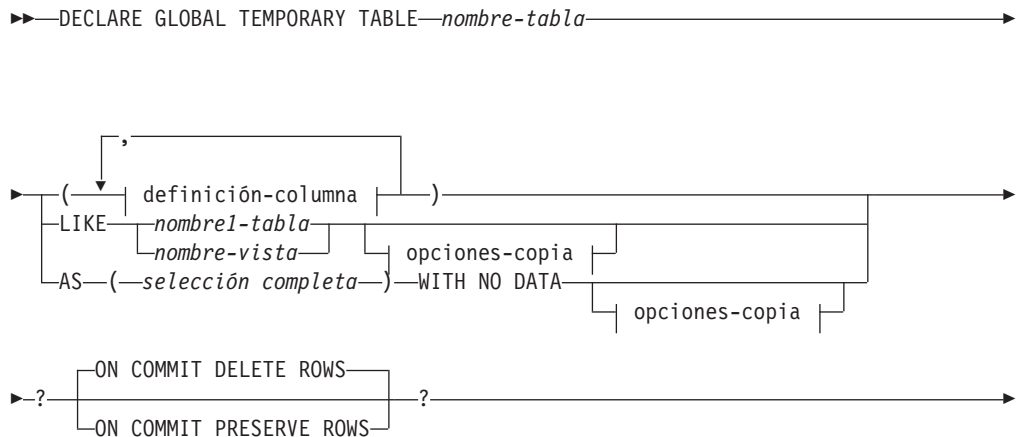
Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio USE para el espacio de tablas USER TEMPORARY
- Autorización DBADM
- Autorización SYSADM
- Autorización SYSCTRL

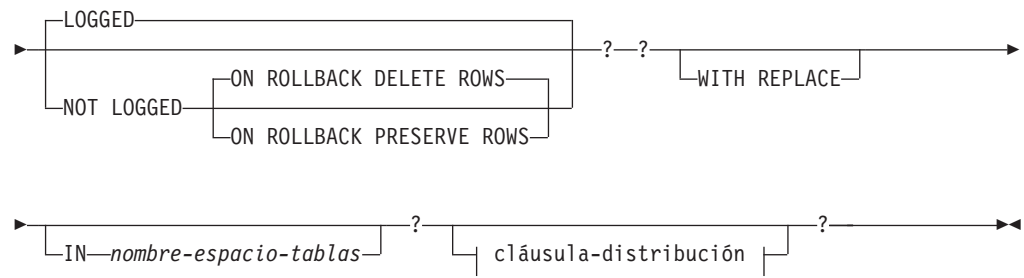
Cuando se define una tabla utilizando LIKE o una selección completa, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla o vista identificada:

- Privilegio SELECT para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Autorización DATAACCESS

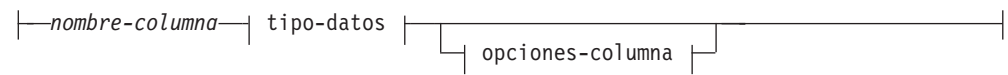
Sintaxis



DECLARE GLOBAL TEMPORARY TABLE



definición-columna:

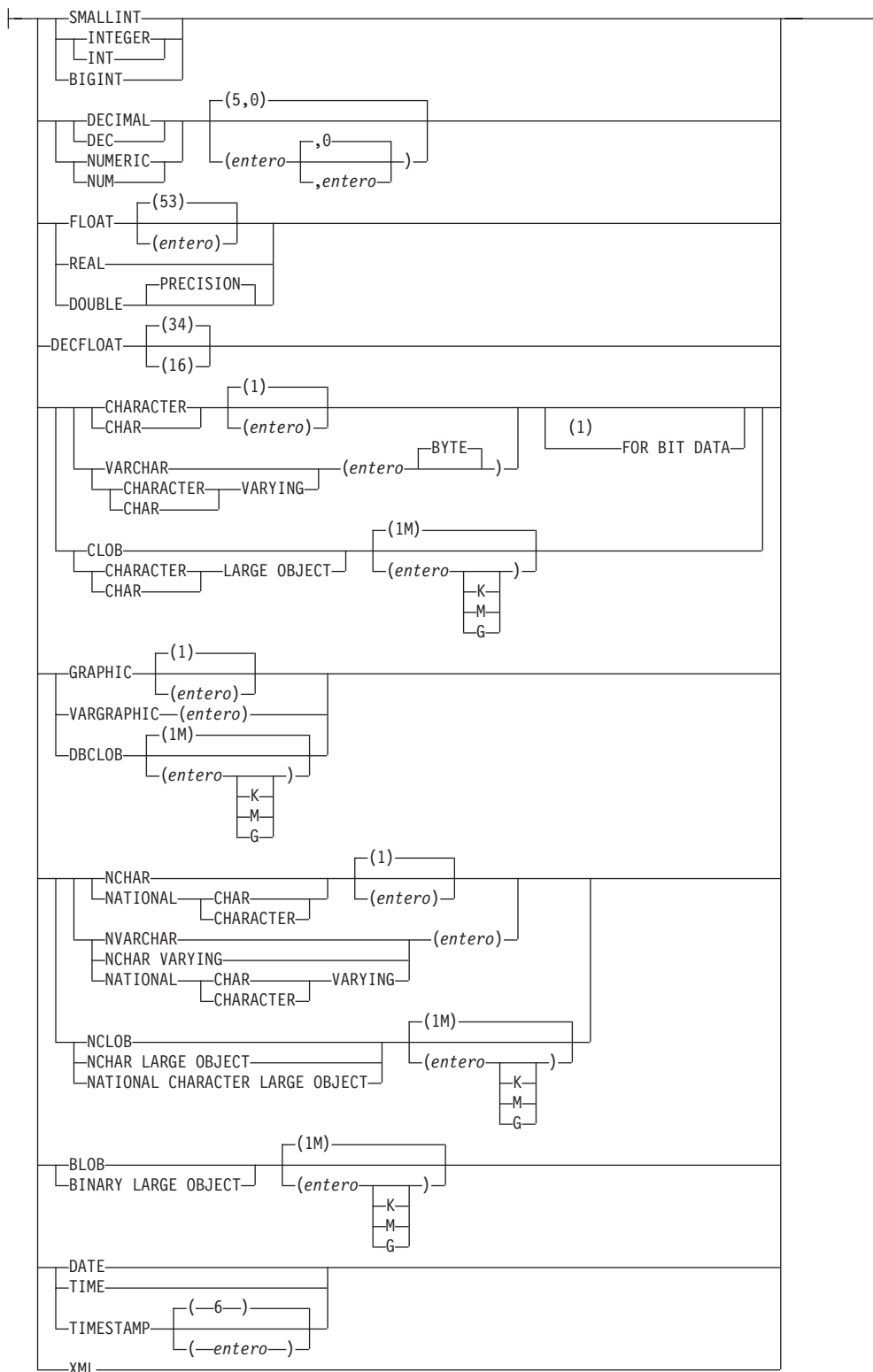


tipo-datos:

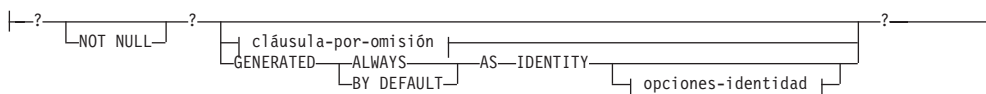


tipo-incorporado:

DECLARE GLOBAL TEMPORARY TABLE

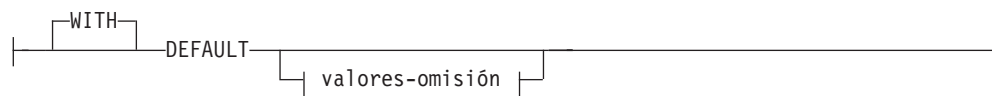


opciones-columna:

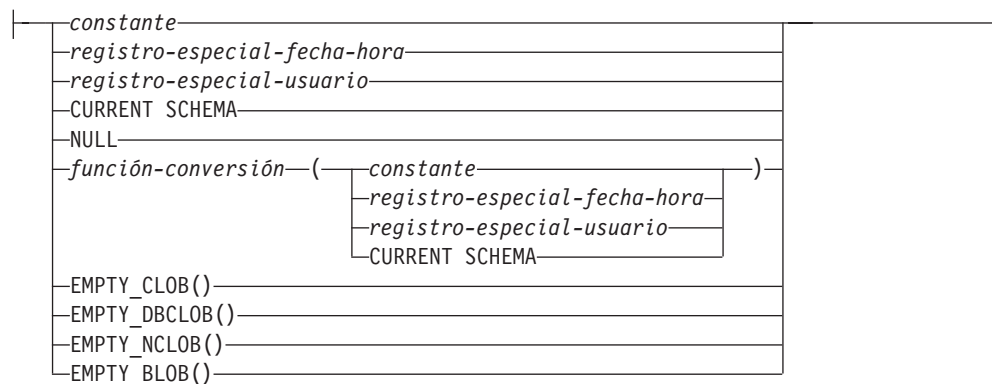


DECLARE GLOBAL TEMPORARY TABLE

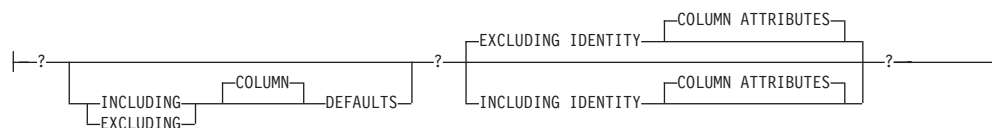
cláusula-por-omisión:



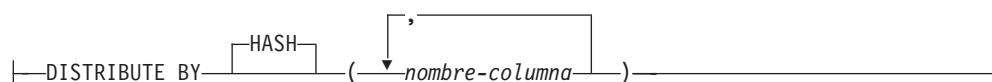
valores-omisión:



opciones-copia:



cláusula-distribución:



Notas:

- 1 La cláusula `FOR BIT DATA` se puede especificar en cualquier orden con las restricciones de columna siguientes.

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 declarada con el mismo *nombre-tabla* tiene su propia descripción exclusiva de esa tabla temporal declarada. 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, alias 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 declarada *nombre-tabla* sin provocar un error ni un aviso
- Las referencias a `SESSION.nombre-tabla` se resolverán en una tabla temporal 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 puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener los atributos siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no debe ser superior a 4.005.
- Un tamaño de página de 8K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 8.101.
- Un tamaño de página de 16K con un máximo de 1.012 columnas, donde el número total 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, donde el número total de bytes de las columnas no debe ser superior a 32.677.

Una tabla temporal creada no puede tener una columna de inicio de fila, una columna de fin de fila o una columna de ID de inicio de transacción.

Para ver más detalles, consulte el apartado “Tamaño de fila” en “CREATE TABLE”.

tipo-datos

Especifica el tipo de datos de la columna.

tipo-incorporado

Especifica un tipo de datos incorporado. Consulte “CREATE TABLE” para obtener una descripción de *tipo-incorporado*.

No se puede especificar un tipo de datos `SYSPROC.DB2SECURITYLABEL` para una tabla temporal declarada.

opciones-columna

Define otras opciones relacionadas con las columnas de la tabla.

NOT NULL

Evita que la columna contenga valores nulos. Para obtener información acerca de la especificación de valores nulos, consulte `NOT NULL` en “CREATE TABLE”.

cláusula-por-omisión

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 “ALTER TABLE”.

DECLARE GLOBAL TEMPORARY TABLE

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-por-omisión* (SQLSTATE 42842).

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 normas de asignación
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- ser una constante numérica o un valor especial de coma flotante decimal si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y, a continuación, se convierten a coma flotante decimal, si la columna de destino es DECFLOAT. Para las columnas DECFLOAT(16), las constantes decimales que tengan una precisión de más de 16 dígitos se redondearán utilizando las modalidades de redondeo especificadas por el registro especial CURRENT DECFLOAT ROUNDING MODE.
- no tener dígitos que no sean cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal (por ejemplo, 1.234 no puede ser el valor por omisión de una columna DECIMAL(5,2)).
- estar expresada con un máximo de 254 bytes, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificados y paréntesis, cuando la constante es el argumento de una *función-conversión*

registro-especial-fecha-hora

Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de 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).

DECLARE GLOBAL TEMPORARY TABLE

registro-especial-usuario

Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER.

CURRENT SCHEMA

Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud superior o igual al atributo de longitud del registro especial CURRENT SCHEMA.

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 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 esta debe coincidir con el nombre del tipo de 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 indicación 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 normas de una constante para el tipo de 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 serie de caracteres.

DECLARE GLOBAL TEMPORARY TABLE

registro-especial-fecha-hora

Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario

Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de 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.

CURRENT SCHEMA

Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()

Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

IDENTITY y opciones-identidad

Para obtener información acerca de la especificación de columnas de identidad, consulte IDENTITY y *opciones-identidad* en "CREATE TABLE".

LIKE nombre1-tabla 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 *nombre-tabla-1*, la vista (*nombre-vista*) o el apodo (*apodo*) identificados. 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). No puede especificarse una tabla protegida (SQLSTATE 42962). No puede especificarse una tabla con una columna definida como IMPLICITLY HIDDEN (SQLSTATE 560AE).

El uso de LIKE es una definición implícita de n columnas, donde n es el número de columnas de la tabla identificada (incluidas las columnas implícitamente ocultas), vista o apodo. La definición implícita depende de lo que se identifica después de LIKE.

- 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-tabla-1*. 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

DECLARE GLOBAL TEMPORARY TABLE

resultante de la selección completa definida en *nombre-vista*. Los tipos de datos de las columnas de la vista deben ser tipos de datos válidos para las columnas de una tabla.

- Si se designa un 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, activadores, índices, claves de particionamiento de tablas ni claves de distribución. 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.

Cuando una tabla se identifica en la cláusula LIKE y dicha tabla contiene una columna ROW CHANGE TIMESTAMP, la columna correspondiente de la nueva tabla hereda sólo el tipo de datos de la columna ROW CHANGE TIMESTAMP. La nueva columna no se considera una columna generada.

Si el control de acceso de nivel de fila o columna (RCAC) se implanta para *nombre1-tabla*, la nueva tabla no hereda RCAC.

AS (*selección-completa*) WITH NO DATA

Especifica que las columnas de la tabla tienen el mismo nombre y descripción que las columnas que aparecerían en la tabla de resultados derivada de la selección completa si esta tuviera que ejecutarse. El uso de AS (*selección completa*) es una definición implícita de n columnas de la tabla temporal declarada, donde n es el número de columnas resultantes de la selección completa.

La definición implícita incluye los atributos siguientes de las n columnas (si son aplicables al tipo de datos):

- Nombre de columna
- Tipo de datos, longitud, precisión y escala
- Capacidad de nulos

Los atributos siguientes no se incluyen (el valor por omisión y los atributos de identidad pueden incluirse utilizando las *opciones-copia*):

- Valor por omisión
- Atributos de identidad
- Atributo oculto
- ROW CHANGE TIMESTAMP

La definición implícita no incluye ningún otro atributo opcional de las tablas o vistas a las que se hace referencia en la selección completa.

Cada elemento de la lista de selección debe tener un nombre exclusivo (SQLSTATE 42711). La cláusula AS puede utilizarse en la cláusula select para proporcionar nombres exclusivos. La selección completa no debe hacer referencia a variables del lenguaje principal ni incluir marcadores de parámetro. Los tipos de datos de las columnas de resultados de la selección completa deben ser tipos de datos válidos para las columnas de una tabla.

DECLARE GLOBAL TEMPORARY TABLE

Si se implanta el control de acceso de nivel de fila o columna para cualquier tabla que se especifique en *selección completa*, RCAC no se aplica en cascada a la nueva tabla.

opciones-copia

Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados fuente (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 de resultados fuente. 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-1*, y *nombre-tabla-1* identifica una tabla base, una tabla temporal creada o una tabla temporal declarada, INCLUDING COLUMN DEFAULTS será el valor por omisión.

EXCLUDING COLUMN DEFAULTS

Especifica que no deben copiarse los valores por omisión de columnas contenidos en la definición de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica una tabla base, una tabla temporal creada 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 de resultados fuente. Es posible copiar estos atributos si el elemento de la columna correspondiente en la tabla, vista o selección completa es el nombre de una columna de una tabla o el nombre de una columna de una vista que, directa o indirectamente, se correlaciona con el nombre de columna de una tabla base o de una tabla temporal creada que tiene la propiedad 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)
- 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 de resultados fuente.

ON COMMIT

Especifica la acción que se realiza sobre la tabla temporal global cuando se ejecuta una operación COMMIT. El valor por omisión es DELETE ROWS.

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.

DECLARE GLOBAL TEMPORARY TABLE

PRESERVE ROWS

Las filas de la tabla se conservan.

LOGGED o NOT LOGGED

Especifica si se anotan cronológicamente las operaciones para la tabla. El valor por omisión es LOGGED.

LOGGED

Especifica que las operaciones de inserción, actualización o supresión efectuadas en la tabla, así como la creación o descarte de la tabla, deben anotarse cronológicamente.

NOT LOGGED

Especifica que las operaciones de inserción, actualización o supresión efectuadas en la tabla no deben anotarse cronológicamente, pero que la creación o descarte de la tabla sí debe anotarse cronológicamente. Durante una operación ROLLBACK (o ROLLBACK TO SAVEPOINT):

- Si la tabla se había creado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla se descarta
- Si la tabla se había descartado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla vuelve a crearse, pero sin datos

ON ROLLBACK

Especifica la acción que va a tener lugar en la tabla temporal global sin anotaciones cronológicas cuando se realice una operación ROLLBACK (o ROLLBACK TO SAVEPOINT). El valor por omisión es DELETE ROWS.

DELETE ROWS

Si se han cambiado los datos de la tabla, se suprimirán todas las filas.

PRESERVE ROWS

Las filas de la tabla se conservan.

WITH REPLACE

Indica que, en el caso de que ya exista una tabla temporal declarada con el nombre especificado, la tabla existente se sustituye por la tabla temporal definida por esta sentencia (y todas las filas de la tabla existente se eliminan).

Si no se especifica WITH REPLACE, el nombre especificado no debe identificar una tabla temporal declarada que ya exista en la sesión actual (SQLSTATE 42710).

IN nombre-espacio-tablas

Identifica el espacio de tablas donde se creará una instancia de la tabla temporal declarada. El espacio de tablas debe existir y estar definido como USER TEMPORARY (SQLSTATE 42838), para el cual el ID de autorización 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 que pertenezca 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 ningún espacio de tablas USER TEMPORARY elegible, se produce un error (SQLSTATE 42727).

La determinación del espacio de tablas puede cambiar cuando:

DECLARE GLOBAL TEMPORARY TABLE

- Se descartan o crean espacios de tablas
- Se otorgan o revocan los 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. Para ver más detalles, consulte el apartado “Tamaño de fila” en “CREATE TABLE”.

cláusula-distribución

Especifica el particionamiento de base de datos o el modo en que se distribuyen los datos en diversas particiones de base de datos.

DISTRIBUTE BY HASH (*nombre-columna,...*)

Especifica la utilización de la función de hash por omisión en las columnas especificadas (denominada *clave de distribución*) como método de distribución en las particiones de base de datos. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No se puede utilizar como parte de una clave de distribución ninguna columna cuyo tipo de datos sea BLOB, CLOB, DBCLOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42962).

Si no se especifica esta cláusula y la tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de distribución se define como la primera columna cuyo tipo de datos es válido para una clave de distribución.

Si ninguna columna cumple los requisitos de una clave de distribución por omisión, la tabla se crea sin ninguna. Estas tablas solo están permitidas en espacios de tablas definidos en grupos de particiones de base de datos de partición única.

Para las tablas de los espacios de tablas que están definidos en grupos de particiones de base de datos de partición única, se puede utilizar cualquier conjunto de columnas con tipos de datos válidos para una clave de distribución a fin de definir la clave de distribución. Si no se especifica esta cláusula, no se crea ninguna clave de distribución.

Notas

- Debe existir un espacio de tablas temporal de usuario antes de declarar una tabla temporal declarada (SQLSTATE 42727).
- **Referencia a una tabla temporal declarada:** La descripción de una tabla temporal 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 declarada (*nombre-tabla*) tiene su propia descripción exclusiva de esa tabla.

Para hacer referencia a la tabla temporal 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 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 declarada 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 estático que hacen referencia a tablas calificadas, implícita o explícitamente, por SESSION, esas

DECLARE GLOBAL TEMPORARY TABLE

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 el tiempo de ejecución, cada referencia de tabla se resolverá en una tabla temporal declarada, en caso de que exista, o en una tabla temporal creada o en una tabla permanente. Si no existe ninguna, aparecerá un error (SQLSTATE 42704).

- **Privilegios:** Cuando se define una tabla temporal declarada, el definidor de la tabla recibe todos los privilegios para la tabla, incluida la capacidad para descartar la tabla. Además, estos privilegios se otorgan a PUBLIC. (Con la opción GRANT no se otorga ninguno de los privilegios, y en la tabla de catálogo no aparece ninguno de los privilegios.) Esto permite que cualquier sentencia de SQL de la sesión haga referencia a una tabla temporal declarada que ya se ha definido en esa sesión.
- **Creación de instancias y terminación:** En las explicaciones siguientes, P representa una sesión y T es una tabla temporal 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 en P puede hacer referencia a T, y cualquier referencia a T en P es una referencia a la misma instancia de T.
 - Si se especifica una sentencia DECLARE GLOBAL TEMPORARY TABLE dentro de la sentencia compuesta del procedimiento de SQL (definida por BEGIN y END), el ámbito de la tabla temporal declarada es la conexión, no sólo la sentencia compuesta, y la tabla es conocida fuera de la sentencia compuesta. La tabla no se descarta implícitamente al final de la sentencia compuesta. Una tabla temporal declarada no se puede definir varias veces con el mismo nombre en otras sentencias compuestas de la sesión, a menos que se haya descartado 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 salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye una modificación de SESSION.T:
 - Si se especificó NOT LOGGED, se suprimen todas las filas desde SESSION.T, salvo que se hubiera especificado también ON ROLLBACK PRESERVE ROWS
 - Si no se ha especificado NOT LOGGED, los cambios realizados en T se deshacen
 - Si se especificó NOT LOGGED y una sentencia INSERT, UPDATE o DELETE falla durante su ejecución (en contraposición con un error de compilación), se suprimen todas las filas desde SESSION.T.
 - Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye la declaración de SESSION.T, la retrotracción incluye la operación DROP SESSION.T.
 - Si una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye descartar la tabla temporal declarada SESSION.T, la retrotracción deshace la operación de descartar la tabla. Si se ha especificado NOT LOGGED, la tabla también se habrá vaciado.

DECLARE GLOBAL TEMPORARY TABLE

- Cuando el proceso de aplicación donde se declaró T finaliza o se desconecta de la base de datos, se descarta T y se destruyen las instancias de sus filas.
 - Cuando finaliza la conexión con el servidor donde se declaró T, se descarta T y se destruyen las instancias de sus filas.
 - **Restricciones en el uso de tablas temporales declaradas:** las tablas temporales declaradas:
 - No se pueden especificar en las sentencias ALTER, COMMENT, GRANT, LOCK, RENAME ni REVOKE (SQLSTATE 42995).
 - No se puede hacer referencia a ellas en una sentencia AUDIT, CREATE ALIAS o CREATE VIEW (SQLSTATE 42995).
 - No se pueden especificar en restricciones de referencia (SQLSTATE 42995).
 - La compresión de filas de datos está habilitada para una tabla temporal declarada. Cuando el gestor de bases de datos determina que existe un beneficio de rendimiento, se comprimirán los datos de fila de tabla, incluidos los documentos XML almacenados en línea en el objeto de tabla base. Sin embargo, la compresión de datos del objeto almacenado XML de una tabla temporal declarada no está soportada.
 - La compresión de índices está habilitada para los índices que se creen en tablas temporales declaradas.
 - **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - DEFINITION ONLY puede especificarse en lugar de WITH NO DATA
 - La cláusula PARTITIONING KEY puede especificarse en lugar de la cláusula DISTRIBUTE BY
- La sintaxis siguiente se acepta como comportamiento por omisión:
- CCSID ASCII
 - CCSID UNICODE

Ejemplos

- *Ejemplo 1:* definir una tabla temporal declarada con definiciones de columna para un número de empleado, salario, incentivo y comisión.

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMP_EMP
  (EMPNO CHAR(6) NOT NULL,
   SALARY DECIMAL(9, 2),
   BONUS DECIMAL(9, 2),
   COMM DECIMAL(9, 2)) ON COMMIT PRESERVE ROWS
```

- *Ejemplo 2:* supongamos que existe la tabla base USER1.EMPTAB y que contiene tres columnas, una de las cuales es una columna de identidad. Declarar una tabla temporal que tenga los mismos nombres y atributos de columna (incluidos los atributos de identidad) que la tabla base.

```
DECLARE GLOBAL TEMPORARY TABLE TEMPTAB1
  LIKE USER1.EMPTAB
  INCLUDING IDENTITY
  ON COMMIT PRESERVE ROWS
```

En este ejemplo, se utiliza SESSION como calificador implícito de TEMPTAB1.

DELETE

La sentencia DELETE suprime filas de una tabla, apodo o vista, o las tablas, apodos o vistas subyacentes de la selección completa especificada.

La supresión de una fila de un apodo suprime la fila del objeto de fuente de datos al que hace referencia el apodo. La supresión de una fila de una vista suprime la fila de la tabla en la que se basa la vista si no se ha definido ningún activador INSTEAD OF para la operación de supresión en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador.

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 incorporarse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

Para ejecutar cualquiera de las dos formas de esta sentencia, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio DELETE para la tabla, vista o apodo de donde van a suprimirse filas
- Privilegio CONTROL sobre la tabla, vista o apodo de donde van a suprimirse filas
- Autorización DATAACCESS

Para ejecutar una sentencia DELETE buscada, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla, vista o apodo al que haga referencia una subconsulta:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

Si el paquete utilizado para procesar la sentencia se ha compilado previamente con normas SQL92 (opción LANGLEVEL con un valor de SQL92E o MIA) y la forma buscada de una sentencia DELETE incluye una referencia a una columna de la tabla o vista en la *condición-búsqueda*, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

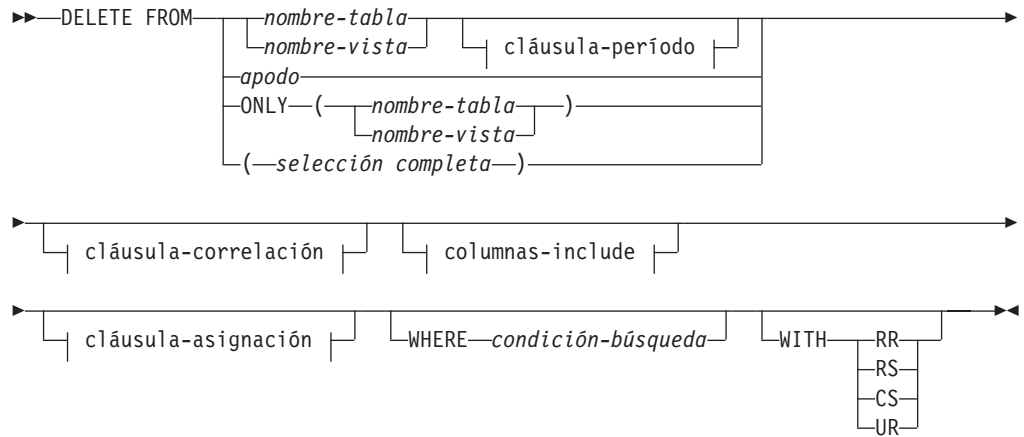
- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

Los privilegios de grupo no se comprueban para las sentencias DELETE estáticas.

Si el destino de la operación de supresión es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis (delete-con-búsqueda)



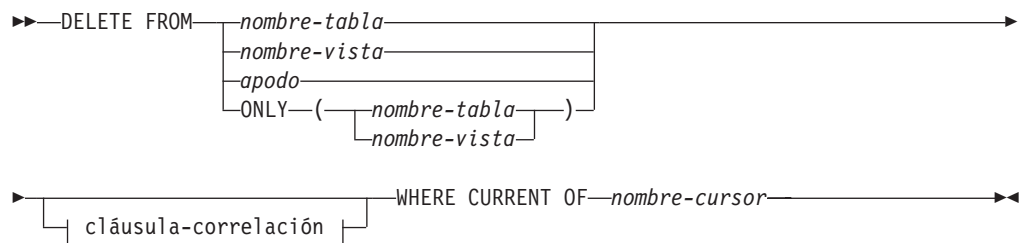
cláusula-período:



columnas-include:

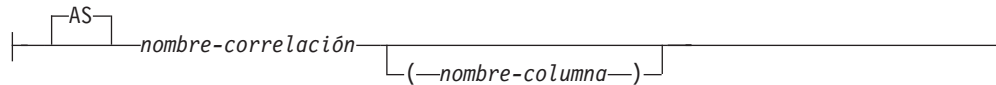


Sintaxis (delete-con-posición)



DELETE

cláusula-correlación:



Descripción

FROM *nombre-tabla*, *nombre-vista*, *apodo*, **o** (*selección completa*)

Identifica el objeto de la operación de supresión. El nombre debe identificar uno de los objetos siguientes:

- Una tabla o vista que existe en el catálogo en el servidor actual
- Una tabla o vista en el servidor remoto que se especifica mediante un nombre-objeto-remoto

El objeto no debe ser una tabla de catálogo, una vista de catálogo, una tabla de consulta materializada mantenida por el sistema o una vista de sólo lectura.

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 normal con una tabla subyacente que es una tabla con tipo, puede que la sentencia suprima filas de la tabla con tipo o cualquiera de sus propias subtablas.

Si el objeto de la operación de actualización es una selección completa, esta debe ser actualizable, según lo definido en el elemento "Vistas suprimibles" de Notas de la descripción de la sentencia CREATE VIEW.

Para obtener restricciones adicionales relacionadas con las tablas temporales y el uso de una vista o selección completa como destino de la operación de supresión, consulte "Consideraciones para una tabla temporal de período de sistema" y "Consideraciones para una tabla temporal de período de aplicación" en la sección Notas.

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 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 efecto en la sentencia.

cláusula-período

Especifica que una cláusula de período se aplica al destino de la operación de supresión.

Si el destino de la operación de supresión es una vista, se aplican las condiciones siguientes a la vista:

- La cláusula FROM de la selección completa externa de la definición de vista debe incluir una referencia, directa o indirectamente, a una tabla temporal de período de aplicación (SQLSTATE 42724M).
- No debe definirse un activador INSTEAD OF DELETE para la vista (SQLSTATE 428HY).

FOR PORTION OF BUSINESS_TIME

Especifica que la supresión sólo se aplica a los valores de fila de la parte correspondiente al período en la fila que especifique la cláusula de período. El período BUSINESS_TIME debe existir en la tabla (SQLSTATE 4274M). No debe especificarse FOR PORTION OF BUSINESS_TIME si el valor del registro especial CURRENT TEMPORAL BUSINESS_TIME no es NULL cuando la opción de vinculación BUSTIMESENSITIVE está establecida en YES (SQLSTATE 428HY).

FROM *valor1* TO *valor2*

Especifica que la supresión se aplica a filas durante el período especificado desde *valor1* hasta *valor2*. No se suprime ninguna fila si *valor1* es superior o igual a *valor2* o si *valor1* o *valor2* es el valor nulo (SQLSTATE 02000).

Para el período especificado con FROM *valor1* TO *valor2*, el período BUSINESS_TIME en una fila del destino de la supresión tiene uno de los estados siguientes:

- **Se solapa con el principio** del período especificado si el valor de la columna de inicio es inferior a *valor1* y el valor de la columna de fin es superior a *valor1*.
- **Se solapa con el final** del período especificado si el valor de la columna de fin es superior o igual a *valor2* y el valor de la columna de inicio es inferior a *valor2*.
- **Está contenido por completo** en el período especificado si el valor de la columna de inicio para BUSINESS_TIME es superior o igual a *valor1* y el valor para la columna de fin correspondiente es inferior o igual a *valor2*.
- Está **contenido parcialmente** en el período especificado si la fila se solapa con el principio del período especificado o el fin del período especificado, pero no con ambos.
- **Se solapa por completo** con el período especificado si el período de la fila se solapa con el principio y el fin del período especificado.
- **No está contenido** en el período si las dos columnas de BUSINESS_TIME son inferiores o iguales a *valor1* o superiores o iguales a *valor2*.

Si el período BUSINESS_TIME de una fila no está contenido en el período especificado, la fila no se suprime. De lo contrario, la supresión se aplica en función de cómo los valores de las columnas del período BUSINESS_TIME se solapan con el período especificado, de la siguiente forma:

- Si el período BUSINESS_TIME de una fila está contenido por completo en el período especificado, se suprime la fila.

DELETE

- Si el período BUSINESS_TIME de una fila está parcialmente contenido en el período especificado y se solapa con el principio del período especificado:
 - La fila se suprime.
 - Se inserta una fila con los valores originales de la fila, salvo que la columna final se establece en *valor1*.
- Si el período BUSINESS_TIME de una fila está parcialmente contenido en el período especificado y se solapa con el final del período especificado:
 - La fila se suprime.
 - Se inserta una fila con los valores originales de la fila, salvo que la columna de inicio se establece en *valor2*.
- Si el período BUSINESS_TIME de una fila se solapa por completo con el período especificado:
 - La fila se suprime.
 - Se inserta una fila con los valores originales de la fila, salvo que la columna final se establece en *valor1*.
 - Se inserta una fila adicional con los valores originales de la fila, salvo que la columna de inicio se establece en *valor2*.

valor1 y valor2

Cada expresión debe devolver un valor que tenga un tipo de datos de fecha, un tipo de datos de indicación de fecha y hora, o un tipo de datos válido para la representación en forma de serie de caracteres de una fecha o indicación de fecha y hora (SQLSTATE 428HY). El resultado de cada expresión debe ser comparable al tipo de datos de las columnas del período especificado (SQLSTATE 42884). Consulte las normas de comparación descritas en “Asignaciones y comparaciones”.

Cada expresión puede contener alguno de los operandos soportados siguientes (SQLSTATE 428HY):

- Constante
- Registro especial
- Variable
- Función escalar cuyos argumentos sean operandos soportados (aunque las funciones definidas por el usuario y las funciones no deterministas no se pueden utilizar)
- Especificación CAST donde el operando de conversión es un operando soportado
- Expresión que utiliza operadores y operandos aritméticos

cláusula-correlación

Se puede utilizar dentro de la *condición-búsqueda* para designar una tabla, vista, apodo o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte “referencia-tabla” en la descripción de “Subselección”.

columns-include

Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia DELETE cuando está anidada en la cláusula FROM de una selección completa. Las *columns-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

INCLUDE

Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia DELETE.

nombre-columna

Especifica una columna de la tabla de resultados intermedia de la sentencia DELETE. El nombre no puede coincidir con el nombre de otra columna incluye ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

tipo-datos

Especifica el tipo de datos de la columna incluye. El tipo de datos debe ser uno que reciba soporte de la sentencia CREATE TABLE.

cláusula-asignación

Consulte la descripción de *cláusula-asignación* bajo la sentencia UPDATE. Se aplican las mismas normas. Las *columnas-include* son las únicas columnas que se pueden definir utilizando la *cláusula-asignación* (SQLSTATE 42703).

WHERE

Especifica una condición que selecciona las filas que se deben 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

Cada *nombre-columna* de la condición de búsqueda, que no sea una subconsulta, debe identificar una columna de la tabla o vista.

La *condición-búsqueda* se aplica a cada fila de la tabla, vista o apodo y las filas que se suprimen son aquellas para las que el resultado de la *condición-búsqueda* es verdadero.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que esta 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 norma 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 ha definido en una sentencia DECLARE CURSOR del programa. La sentencia DECLARE CURSOR debe preceder a la sentencia DELETE.

La tabla, vista o apodo que se ha indicado también debe indicarse en la cláusula FROM de la sentencia SELECT del cursor y la tabla de resultados del cursor no debe ser de sólo lectura. Para ver una explicación sobre las tablas de resultados de sólo lectura, consulte "DECLARE CURSOR".

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 de resultados. 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.

DELETE

- 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. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

Normas

- **Activadores:** Las sentencias DELETE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en las filas suprimidas. Si una sentencia DELETE de una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobará la integridad referencial de las actualizaciones que se han realizado en el activador y no la de las tablas subyacentes de la vista que ha dado lugar a que se ejecutara el activador.
- **Integridad referencial:** Si la tabla identificada o la tabla base de la vista identificada es padre, las filas seleccionadas para la supresión no deberán tener ningún dependiente en una relación con una norma de supresión de RESTRICT, y la sentencia DELETE no deberá aplicarse en cascada a las filas descendentes que tengan dependientes en una relación con una norma de supresión de RESTRICT.

Si no se impide la operación de supresión por una norma de supresión RESTRICT, se suprimen las filas seleccionadas. Las filas que son dependientes de las 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 norma de supresión de SET NULL se establecen en el valor nulo.
- Las filas que sean sus dependientes en una relación con una norma de supresión de CASCADE también se suprimen y se aplican, a su vez, las normas anteriores a estas filas.

Se comprueba la norma 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.

- **Política de seguridad:** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
 - Acceso de grabación a todas las columnas protegidas (SQLSTATE 42512)
 - Acceso de grabación y de lectura a todas las filas que se han seleccionado para la supresión (SQLSTATE 42519)

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.
- Salvo 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 de resultados. 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 de resultados.
- SQLERRD(3) en la SQLCA muestra el número de filas que van a incluirse en la operación de supresión. En el contexto de una sentencia de procedimiento de SQL, el valor puede recuperarse utilizando la variable ROW_COUNT de la sentencia GET DIAGNOSTICS. 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 norma de supresión CASCADE y filas en las que se han establecido claves foráneas en el valor null como resultado de una norma de supresión SET NULL. En relación a las sentencias activadas, incluye el número de filas que se han insertado, actualizado o suprimido.
- 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.
- Para los apodos, la opción del servidor externo iud_app_svpt_enforce plantea una limitación adicional. Consulte la documentación acerca de los objetos federados para obtener más información.
- Para algunas fuentes de datos, SQLCODE -20190 podría devolverse en una supresión realizada para un apodo debido a una posible falta de coherencia en los datos. Consulte la documentación acerca de los objetos federados para obtener más información.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
 - La palabra clave FROM puede omitirse.
- **Consideraciones para una tabla temporal de período de sistema:** El destino de la sentencia DELETE no debe ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para SYSTEM_TIME si la vista está definida con WITH CHECK OPTION y la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):
 - Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente)
 - Una invocación de una rutina de SQL que tiene un paquete asociado a ella
 - Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

DELETE

Si el registro especial CURRENT TEMPORAL SYSTEM_TIME se establece en un valor no nulo, el destino subyacente de la sentencia UPDATE no debe ser una tabla temporal de período de sistema (SQLSTATE 51046), y el destino de la sentencia DELETE no debe ser una vista definida con WITH CHECK OPTION si la definición de vista incluye cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente)
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Si la sentencia DELETE tiene una condición de búsqueda que contiene una subconsulta correlacionada que hace referencia a filas históricas (con referencia explícita al nombre de la tabla histórica o implícita a través del uso de una especificación de período en la cláusula FROM), las filas suprimidas que se almacenan como filas históricas son potencialmente visibles para operaciones de supresión para las filas procesadas posteriormente para la sentencia.

El algoritmo de supresión masiva no se utiliza para una sentencia DELETE para una tabla definida como tabla temporal de período de sistema que no contiene una condición de búsqueda.

- **Consideraciones sobre una tabla histórica:** cuando se suprime una fila de una tabla temporal de período de sistema, se inserta una copia histórica de la fila en la tabla histórica correspondiente y la indicación de fecha y hora final de la fila histórica se captura en forma de valor determinado por el sistema que se corresponde con la hora de la operación de cambio de datos. El gestor de bases de datos asigna el valor que se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exige que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla, o que se suprima una fila de una tabla temporal de período de sistema. El gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de fin de una tabla histórica en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas insertadas en la tabla histórica tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio, lo que puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema (SQLSTATE 01695). El parámetro de configuración de base de datos **system_time_period_adj** debe establecerse en Yes para que tenga lugar este ajuste en el valor de indicación de fecha y hora; de lo contrario, se devuelve un error (SQLSTATE 57062).

En el caso de una operación de supresión, el ajuste solamente afecta al valor de la columna final de la tabla histórica correspondiente a la columna de fin de fila de la tabla temporal de período de sistema asociada. Tenga en cuenta estos ajustes en las referencias posteriores a la tabla cuando se realice una búsqueda de la hora de inicio de la transacción en la columna de inicio de fila y la columna de fin de fila para el período SYSTEM_TIME de la tabla temporal de período de sistema asociada.

- **Consideraciones para una tabla temporal de período de aplicación:** El destino de la sentencia DELETE no debe ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para BUSINESS_TIME si la vista está definida con WITH CHECK OPTION y la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):
 - Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente)

- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Si el registro especial CURRENT TEMPORAL BUSINESS_TIME se establece en un valor no nulo, el destino de la sentencia DELETE no debe ser una vista definida con WITH CHECK OPTION si la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente)
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Una sentencia DELETE para una tabla temporal de período de aplicación que contiene una cláusula FOR PORTION OF BUSINESS_TIME indica entre qué dos puntos en el tiempo las supresiones están en vigor. Cuando se especifica FOR PORTION OF BUSINESS_TIME y el valor de período de una fila, especificado por los valores de columna de inicio de fila y columna de fin de fila, sólo está contenido parcialmente en el período especificado desde *valor1* hasta *valor2*, la fila se suprime y una o dos filas se insertan de forma automática para representar la parte de la fila que no se ha suprimido. Se generan nuevos valores para cada columna generada en una tabla temporal de período de aplicación para cada fila que se inserta de forma automática como resultado de una operación de supresión en la tabla. Si se define una columna generada como parte de una clave exclusiva o primaria, una clave principal de una restricción referencial o un índice exclusivo, es posible que una inserción automática viole una restricción o índice, en cuyo caso se devolverá un error.

Cuando una tabla temporal de período de aplicación es el destino de una sentencia DELETE, el valor en vigor para el registro especial CURRENT TEMPORAL BUSINESS_TIME no es el valor nulo y la opción de vinculación BUSTIMESENSITIVE está establecida en YES, están implícitos los predicados adicionales siguientes:

```
bt_begin <= CURRENT TEMPORAL BUSINESS_TIME
AND bt_end > CURRENT TEMPORAL BUSINESS_TIME
```

donde bt_begin y bt_end son las columnas inicial y final del período BUSINESS_TIME de la tabla de destino de la sentencia DELETE.

- **Consideraciones sobre tablas temporales de período de aplicación y activadores:** Cuando se suprime una fila y se especifica la cláusula FOR PORTION OF BUSINESS_TIME, se pueden insertar implícitamente filas adicionales para reflejar cualquier porción de la fila que no haya sido suprimida. Los activadores de supresión existentes se activan para las filas suprimidas y los activadores de inserción existentes se activan para las filas que se han insertado implícitamente.

Ejemplos

- *Ejemplo 1:* Suprimir 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
```

DELETE

- *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')
```

-
- *Ejemplo 4:* Suprima todas las filas de empleado duplicadas de la tabla EMPLOYEE. Una fila de empleado se considera duplicada si coinciden los apellidos. Conserve la fila de empleado con el nombre menor en orden léxico.

```
DELETE FROM
(SELECT ROWNUMBER() OVER (PARTITION BY LASTNAME ORDER BY FIRSTNAME)
 FROM EMPLOYEE) AS E(RN)
WHERE RN > 1
```

DESCRIBE

La sentencia DESCRIBE obtiene información acerca de un objeto.

Con esta sentencia se pueden obtener dos tipos de información. Cada una de ellos se describe por separado.

- Marcadores de parámetro de entrada de una sentencia preparada. Obtiene información sobre los marcadores de parámetro de entrada en una sentencia preparada. Esta información se pone en un descriptor.
- La salida de una sentencia preparada. Obtiene información sobre una sentencia preparada o información sobre las columnas de lista de selección de una sentencia SELECT preparada. Esta información se pone en un descriptor.

DESCRIBE INPUT

La sentencia DESCRIBE INPUT obtiene información sobre los marcadores de parámetro de entrada de una sentencia preparada.

Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis

►—DESCRIBE INPUT—*nombre-sentencia*—INTO—*nombre-descriptor*—►

Descripción

nombre-sentencia

Identifica la sentencia preparada. Cuando se ejecuta la sentencia DESCRIBE INPUT, el nombre debe identificar una sentencia que el proceso de aplicación haya preparado en el servidor actual.

Para una sentencia CALL, la información devuelta describe los parámetros de entrada, definidos como IN o INOUT, del procedimiento. Los marcadores de parámetro de entrada se consideran siempre como posibilidad de nulos, independientemente de su uso.

INTO *nombre-descriptor*

Identifica un área de descriptor de SQL (SQLDA). Antes de ejecutar la sentencia DESCRIBE INPUT, se debe establecer la variable siguiente en la SQLDA:

SQLN Especifica el número de apariciones de SQLVAR proporcionadas en la SQLDA. SQLN se debe establecer en un valor mayor que o igual a cero antes de que se ejecute la sentencia DESCRIBE INPUT.

Cuando se ejecuta la sentencia DESCRIBE INPUT, el gestor de bases de datos asigna valores a las variables de la SQLDA como se indica a continuación:

SQLDAID

Los 6 primeros bytes se establecen en 'SQLDA ' (es decir, 5 letras seguidas del carácter de espacio).

El séptimo byte, definido como SQLDOUBLED, se establece basándose en los marcadores de parámetro descritos:

- Si la SQLDA contiene dos entradas SQLVAR para cada parámetro de entrada, el séptimo byte se establece en '2'. Se utiliza esta técnica para acomodar los parámetros de entrada de tipo estructurado o LOB.
- De lo contrario, el séptimo byte se establece en el carácter de espacio.

El séptimo byte se establece en el carácter de espacio si no hay suficiente espacio en la SQLDA para incluir la descripción de todos los marcadores de parámetro de entrada.

El octavo byte se establece en el carácter de espacio.

SQLDABC

Longitud de la SQLDA en bytes.

SQLD Número de parámetros IN e INOUT del procedimiento.

SQLVAR

Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las apariciones de SQLVAR.

Si el valor de SQLD es n , donde n es mayor que 0 pero menor que o igual al valor de SQLN, se asignan valores a las primeras n apariciones de SQLVAR. Los valores describen los marcadores de los parámetros de entrada del procedimiento. La primera aparición de SQLVAR describe el primer marcador de parámetro de entrada, la segunda aparición de SQLVAR describe el segundo marcador de parámetro de entrada y así sucesivamente.

SQLVAR base**SQLTYPE**

Código que muestra el tipo de datos del parámetro e indica si puede o no puede contener valores nulos.

SQLLEN

Valor de longitud que depende del tipo de datos del parámetro. SQLLEN es 0 para tipos de datos LOB.

SQLNAME

El valor de la variable sqlname se obtiene como se indica a continuación:

- Si SQLVAR corresponde a un marcador de parámetro que se encuentra en la lista de parámetros de un procedimiento y no forma parte de una expresión, sqlname contendrá el nombre del parámetro si se especificó uno en la sentencia CREATE PROCEDURE.
- Si SQLVAR corresponde a un marcador de parámetro con nombre, sqlname contendrá el nombre del marcador de parámetro.
- De lo contrario, sqlname contendrá un valor literal numérico ASCII que representa la posición de SQLVAR dentro de SQLDA.

SQLVAR secundaria

Estas variables sólo se utilizan si el número de entradas SQLVAR se dobla para acomodar parámetros LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN

Atributo de longitud de un parámetro BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME

Para cualquier parámetro de tipo definido por el usuario (diferenciado o estructurado), el gestor de base de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado. Para un parámetro de tipo de referencia, el gestor de base de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado del tipo de 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

- **Preparación de la SQLDA:** antes de ejecutar la sentencia DESCRIBE INPUT, se debe asignar la SQLDA y el valor de SQLN debe establecerse en un valor superior o igual a cero para indicar cuántas apariciones de SQLVAR se proporcionan en la SQLDA. Es preciso asignar almacenamiento suficiente para contener las apariciones de SQLN. Para obtener la descripción de los marcadores de parámetro de entrada en la sentencia preparada, es preciso que el número de apariciones de SQLVAR no sea inferior al número de marcadores de parámetro de entrada. Además, si los marcadores de parámetro de entrada incluyen LOB o tipos estructurados, el número de apariciones de SQLVAR debe duplicar el número de marcadores de parámetro de entrada.
- Las conversiones de páginas de códigos entre las páginas de código UNIX ampliado (EUC) y las páginas de códigos DBCS o entre las páginas de código Unicode y no Unicode pueden producir la expansión o la contracción de las longitudes de caracteres.
- Si se selecciona un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se ha especificado 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), se devolverá un error.
- **Asignación de la SQLDA:** Entre las maneras posibles de asignar la SQLDA están las tres descritas a continuación:

Primera técnica: Asigne una SQLDA con suficientes apariciones 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 INPUT con una SQLDA que no tenga apariciones de SQLVAR; es decir, una SQLDA para la que SQLN sea cero. El valor devuelto para SQLD es el número de columnas de la tabla de resultados. Este es el número necesario de apariciones 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 que se asocia a ese aviso es +237, +238 o +239, el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD. (En la devolución de estos SQLCODE positivos se da por supuesto que el valor de la opción de enlace SQLWARN era YES (devolución de SQLCODE positivos). Si SQLWARN se había establecido en NO, sigue devolviéndose +238 para indicar que el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD.)
2. Asigne una SQLDA con suficientes apariciones de SQLVAR. Después ejecute la sentencia DESCRIBE de nuevo, utilizando esta SQLDA nueva.

Esta técnica permite una mejor gestión de almacenamiento que la primera técnica, pero duplica el número de sentencias DESCRIBE INPUT.

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 INPUT y compruebe el valor SQLD. Utilice el valor SQLD para el número de apariciones 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

Ejecute una sentencia DESCRIBE INPUT con una SQLDA que tenga suficientes apariciones SQLVAR para describir cualquier número de parámetros de entrada que una sentencia preparada pueda tener. Supongamos que será preciso describir un máximo de cinco parámetros y que los datos de entrada no contienen LOB.

```

    /* STMT1_STR contiene la sentencia INSERT con la cláusula VALUES */
EXEC SQL PREPARE STMT1_NAME FROM :STMT1_STR;
... /* código para establecer SQLN en 5 y asignar la SQLDA */
EXEC SQL DESCRIBE INPUT STMT1_NAME INTO :SQLDA;
.
.
.

```

Este ejemplo utiliza la primera técnica descrita en “Asignación de la SQLDA” en “DESCRIBE OUTPUT”.

DESCRIBE OUTPUT

La sentencia DESCRIBE OUTPUT obtiene información acerca de una sentencia preparada.

Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis

```
►►—DESCRIBE—OUTPUT—nombre-sentencia—INTO—nombre-descriptor—►►
```

Descripción

nombre-sentencia

Identifica la sentencia preparada. Cuando se ejecuta la sentencia DESCRIBE OUTPUT, el nombre debe identificar una sentencia que el proceso de aplicación haya preparado en el servidor actual.

Si la sentencia preparada es una sentencia SELECT o VALUES INTO, la información devuelta describe las columnas en la tabla de resultados. Si la sentencia preparada es una sentencia CALL, la información devuelta describe los parámetros de salida, definidos como OUT o INOUT, del procedimiento.

INTO *nombre-descriptor*

Identifica un área de descriptor de SQL (SQLDA). Antes de ejecutar la sentencia DESCRIBE OUTPUT, se debe establecer la variable siguiente en la SQLDA:

SQLN Especifica el número de apariciones de SQLVAR proporcionadas en la SQLDA. SQLN debe establecerse en un valor mayor que o igual a cero antes de que se ejecute la sentencia DESCRIBE OUTPUT.

Cuando se ejecuta la sentencia DESCRIBE OUTPUT, 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, definido como SQLDOUBLED, se establece basándose en las columnas de resultados o los marcadores de parámetro descritos:

- Si la SQLDA contiene dos entradas SQLVAR para cada columna o parámetro de salida, el séptimo byte se establece en '2'. Esta técnica se utiliza para acomodar columnas o parámetros de salida de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.
- De lo contrario, el séptimo byte se establece en el carácter de espacio.

El séptimo byte se establece en el carácter de espacio si no hay suficiente espacio en la SQLDA para incluir la descripción de todos los marcadores de parámetro de salida o las columnas de resultado.

El octavo byte se establece en el carácter de espacio.

SQLDABC

Longitud de la SQLDA en bytes.

SQLD Si la sentencia preparada es SELECT, SQLD se establece en el número de columnas de la tabla de resultados. Si la sentencia preparada es una sentencia CALL, SQLD se establece en el número de parámetros OUT e INOUT del procedimiento. De lo contrario, SQLD se establece en 0.

SQLVAR

Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las apariciones de SQLVAR.

Si el valor de SQLD es *n*, donde *n* es mayor que 0 pero menor que o igual al valor de SQLN, se asigna valores a SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN y SQLDATATYPE_NAME para las primeras *n* apariciones de SQLVAR. Estos valores describen columnas de la tabla de resultados o marcadores de parámetro para los parámetros de salida del procedimiento. La primera aparición de SQLVAR describe la primera columna o el primer marcador de parámetro de salida, la segunda aparición de SQLVAR describe la segunda columna o el segundo marcador de parámetro de salida y así sucesivamente.

SQLVAR base**SQLTYPE**

Código que muestra el tipo de datos de la columna o del parámetro e indica si puede o no puede contener valores nulos.

SQLLEN

Valor de longitud que depende del tipo de datos de la columna o del parámetro. SQLLEN es 0 para tipos de datos LOB.

SQLNAME

El valor de la variable sqlname se obtiene como se indica a continuación:

- Si SQLVAR corresponde a una columna que se ha obtenido para una referencia a una columna simple en la lista de selección de la sentencia-select, sqlname es el nombre de la columna.
- Si SQLVAR corresponde a un marcador de parámetro que se encuentra en la lista de parámetros de un procedimiento y no forma parte de una expresión, sqlname contendrá el nombre del parámetro si se especificó uno en CREATE PROCEDURE.
- De otro modo, sqlname contiene un valor literal numérico ASCII que representa la posición de SQLVAR dentro de la SQLDA.

SQLVAR secundaria

Estas variables sólo se utilizan si el número de entradas SQLVAR se doblan para acomodar columnas o parámetros de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN

El atributo de longitud de una columna o un parámetro BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME

Para cualquier columna o parámetro de tipo definido por el

DESCRIBE OUTPUT

usuario (diferenciado o estructurado), el gestor de bases de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado. Para una columna o un parámetro de tipo de referencia, el gestor de bases de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado del tipo de 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 OUTPUT, el valor de SQLN debe establecerse de manera que indique cuántas apariciones de SQLVAR se proporcionan en la SQLDA y debe asignarse suficiente almacenamiento para incluir las apariciones de SQLN. Por ejemplo, para obtener la descripción de las columnas de la tabla de resultados de una sentencia SELECT preparada, el número de apariciones 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 que se haya ejecutado la sentencia DESCRIBE OUTPUT pero antes de asignar almacenamiento para que un SQLTYPE de SQL_TYP_xLOB se cambie por SQL_TYP_xLOB_LOCATOR o SQL_TYP_xLOB_FILE con los cambios correspondientes en otros campos, como SQLLEN. Después asigne el almacenamiento basado en SQLTYPE y continúe.
- Las conversiones de páginas de códigos entre páginas de códigos UNIX ampliado (EUC) y páginas de códigos DBCS, o entre páginas de códigos Unicode y no Unicode, puede producir la expansión y la contracción de las longitudes de caracteres.
- Si se selecciona un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se ha especificado 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), se devolverá un error.
- **Asignación de la SQLDA:** Entre las maneras posibles de asignar la SQLDA están las tres descritas a continuación:

Primera técnica: Asigne una SQLDA con suficientes apariciones 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 OUTPUT con una SQLDA que no tenga apariciones de SQLVAR; es decir, una SQLDA para la que SQLN es cero. El valor devuelto para SQLD es el número de columnas de la tabla de resultados. Este es el número necesario de apariciones 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 que se asocia a ese aviso es +237, +238 o +239, el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD. (En la devolución de estos SQLCODE positivos se da por supuesto que el valor de la opción de enlace SQLWARN era YES (devolución de SQLCODE positivos). Si SQLWARN se había establecido en NO, sigue devolviéndose +238 para indicar que el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD.)

2. Asigne una SQLDA con suficientes apariciones de SQLVAR. A continuación, ejecute la sentencia DESCRIBE OUTPUT de nuevo, utilizando esta SQLDA nueva.

Esta técnica permite una mejora gestión de almacenamiento que la primera técnica, pero doble el número de sentencias DESCRIBE OUTPUT.

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 apariciones 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.

- **Consideraciones para columnas ocultas implícitamente:** una sentencia DESCRIBE OUTPUT devuelve información de una columna implícitamente oculta sólo si la columna está especificada explícitamente como parte de la lista SELECT de la tabla resultante final de la consulta que se describe. Si las columnas implícitamente ocultas no forman parte de la tabla resultante de una consulta, la sentencia DESCRIBE OUTPUT que devuelve información sobre dicha consulta no contendrá información sobre columnas implícitamente ocultas.

Ejemplo

En un programa C, ejecute una sentencia DESCRIBE OUTPUT con una SQLDA que no tenga apariciones de SQLVAR. Si SQLD es mayor que cero, utilice el valor para asignar una SQLDA con el número necesario de apariciones 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 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 de resultados
*/
```

DESCRIBE OUTPUT

```
EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;  
.  
.  
.
```

DISCONNECT

La sentencia DISCONNECT finaliza una o más conexiones cuando no existe ninguna unidad de trabajo activa (es decir, tras una operación de confirmación o de retrotracción).

Si el destino de la sentencia DISCONNECT es una única conexión, la conexión finaliza sólo si la base de datos ha participado en una unidad de trabajo existente, con independencia de si existe o no una unidad de trabajo activa. Por ejemplo, si otras bases de datos han realizado tareas, pero el destino en cuestión no, la desconexión puede tener lugar sin finalizarse la conexión.

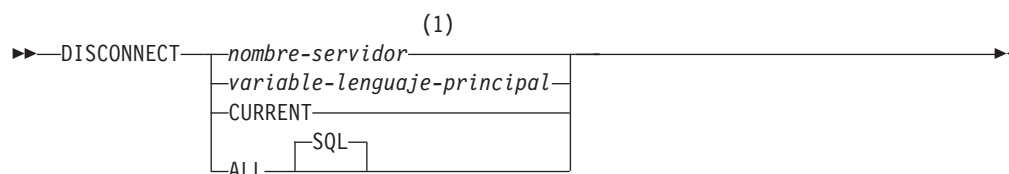
Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente.

Autorización

No se necesita.

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-lenguaje-principal*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, 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 de indicador. El *nombre-servidor* contenido dentro de la *variable-lenguaje-principal* debe alinearse a la izquierda y no debe delimitarse mediante comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El alias-basedatos especificado o el alias-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el alias-basedatos no identifica ninguna conexión existente, se genera un error (SQLSTATE 08003).

DISCONNECT

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 deben 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 norma 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). Se utiliza cuando la opción de precompilador 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-lenguaje-principal* en la sentencia DISCONNECT, debe identificar la conexión actual porque CONNECT Tipo 1 sólo da soporte a una conexión cada vez. Generalmente, DISCONNECT no se ejecutará correctamente si se produce dentro de una unidad de trabajo, con la excepción que se indica en "Normas".
- 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. Para obtener información acerca de la especificación de la opción DISCONNECT, consulte el apartado "Bases de datos relacionales distribuidas".

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. 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 se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Cuando se descartan objetos que admiten nombres de dos partes, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio DROPIN para el esquema del objeto
- Propietario del objeto, tal como está registrado en la columna OWNER de la vista de catálogo del objeto
- Privilegio CONTROL para el objeto (aplicable sólo a índices, especificaciones de índice, apodos, paquetes, tablas y vistas)
- Propietario del tipo definido por el usuario, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.DATATYPES (aplicable sólo al descartar un método asociado a un tipo definido por el usuario)
- Autorización DBADM

Al descartar una jerarquía de tablas o vistas, los privilegios del ID de autorización de la sentencia deben incluir uno de los privilegios indicados anteriormente para cada tabla o vista de la jerarquía.

Cuando se descarta una política de auditoría, los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Cuando se descarta una agrupación de almacenamientos intermedios, un grupo de particiones de base de datos, un grupo de almacenamiento o un espacio de tablas, los privilegios del ID de autorización de la sentencia deben incluir la autorización SYSADM o SYSCTRL.

Cuando se descarta una correlación de tipo de datos, una correlación de funciones, una definición de servidor o un derivador, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM.

Cuando se descarta un supervisor de sucesos, el privilegio del ID de autorización de la sentencia debe incluir la autorización SQLADM o DBADM.

Cuando se descarta un rol, los privilegios del ID de autorización de la sentencia deben incluir la autorización SECADM.

Cuando se descarta un permiso de fila o una máscara de columna, los privilegios del ID de autorización de la sentencia deben incluir la autorización SECADM.

Cuando se descarta un esquema, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM o ser el propietario del esquema, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.SCHEMATA.

Cuando se descarta una etiqueta de seguridad, un componente de una etiqueta de seguridad o una política de seguridad, los privilegios del ID de autorización de la sentencia deben incluir la autorización SECADM.

Cuando se descarta una clase de servicio, un conjunto de acciones de trabajo, un conjunto de clases de trabajo, una carga de trabajo, un umbral o una plantilla de histograma, los privilegios del ID de autorización de la sentencia deben incluir la autorización WLMADM o DBADM.

Cuando se descarta una tabla temporal de período de sistema, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegios para descartar la tabla histórica asociada
- Autorización administrativa

Cuando se descarta una transformación, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM, o debe ser el propietario del *nombre-tipo*.

Cuando se descarta un contexto fiable, los privilegios del ID de autorización de la sentencia deben incluir la autorización SECADM.

Cuando se descarta un supervisor de sucesos o la lista de uso, el privilegio del ID de autorización de la sentencia debe incluir la autorización SQLADM o DBADM.

Cuando se descarta una correlación de usuario, los privilegios del ID de autorización de la sentencia deben incluir la autorización DBADM, si dicho ID de autorización no es el 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.

Sintaxis

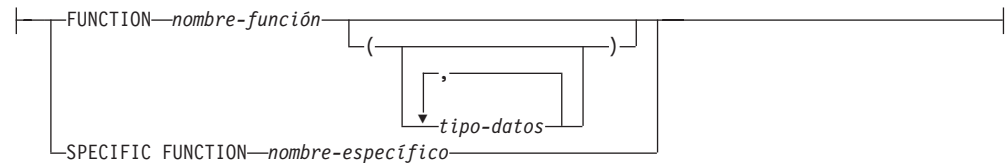
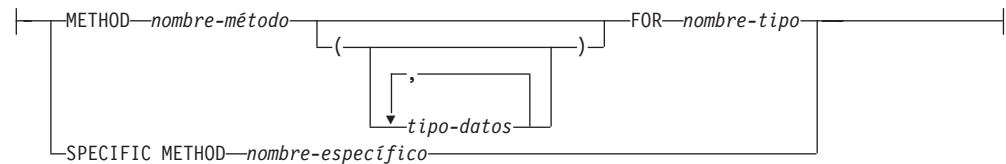
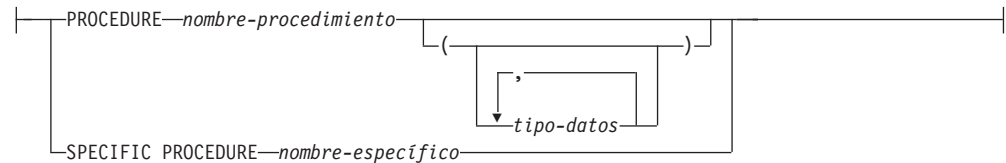
►►—DROP—►

DROP

designador-alias			
AUDIT POLICY	nombre-política		
BUFFERPOOL	nombre-agrup-almac-interm		
DATABASE PARTITION GROUP	nombre-grupo-particiones-bd		
EVENT MONITOR	nombre-supervisor-sucesos		
designador-función		RESTRICT	
FUNCTION MAPPING	nombre-correlación-función		
HISTOGRAM TEMPLATE	nombre-plantilla		
	(1)		
INDEX	nombre-índice		
INDEX EXTENSION	nombre-extensión-índice	RESTRICT	
MASK	nombre-máscara		
designador-método		RESTRICT	
MODULE	nombre-módulo		
NICKNAME	apodo		
PACKAGE	nombre-esquema.	id-paquete	VERSION id-versión
PERMISSION	nombre-permiso		
designador-procedimiento		RESTRICT	
ROLE	nombre-rol		
SCHEMA	nombre-esquema	RESTRICT	
SECURITY LABEL	nombre-etiqueta-seguridad	RESTRICT	
SECURITY LABEL COMPONENT	nombre-componente-etiqueta-seguridad	RESTRICT	
SECURITY POLICY	nombre-política-seguridad	RESTRICT	
SEQUENCE	nombre-secuencia	RESTRICT	
SERVER	nombre-servidor		
designador-clase-servicio		RESTRICT	
STOGROUP	nombre-grupo-almacenamiento	RESTRICT	
TABLE	nombre-tabla		
TABLE HIERARCHY	nombre-tabla-raíz		
TABLESPACE	nombre-espacio-tablas		
TABLESPACES			
TRANSFORM	ALL	FOR	nombre-tipo
TRANSFORMS	nombre-grupo		
THRESHOLD	nombre-umbral		
TRIGGER	nombre-activador		
TRUSTED CONTEXT	nombre-contexto		
TYPE	nombre-tipo	RESTRICT	
TYPE MAPPING	nombre-correlación-tipos		
USAGE LIST	nombre-lista-uso		
USER MAPPING FOR	nombre-autorización	SERVER	nombre-servidor
	USER		
VARIABLE	nombre-variable	RESTRICT	
VIEW	nombre-vista		
VIEW HIERARCHY	nombre-vista-raíz		
WORK ACTION SET	nombre-conjunto-acciones-trabajo		
WORK CLASS SET	nombre-conjunto-clases-trabajo		
WORKLOAD	nombre-carga-trabajo		
WRAPPER	nombre-derivador		
XROBJECT	nombre-objetoxsr		

designador-alias:

ALIAS	nombre-alias	FOR TABLE
PUBLIC		FOR MODULE
		FOR SEQUENCE

designador-función:**designador-método:****designador-procedimiento:****designador-clase-servicio:****Notas:**

- 1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.

Descripción*designador-alias***ALIAS** *nombre-alias*

Identifica el alias que se debe descartar. El *nombre-alias* debe designar un alias descrito en el catálogo (SQLSTATE 42704). Se suprime el alias especificado.

Para TABLE, FOR MODULE o FOR SEQUENCE

Especifica el tipo de objeto del alias.

FOR TABLE

El alias es para una tabla, vista o apodo.

FOR MODULE

El alias es para un módulo.

FOR SEQUENCE

El alias es para una secuencia.

DROP

Todas las vistas y activadores que hacen referencia al alias se establecen como no operativos. Esto incluye las referencias al alias de la cláusula ON de la sentencia CREATE TRIGGER y las que se encuentran dentro de las sentencias de SQL activadas. Se descartará cualquier tabla de consulta materializada o tabla de etapas que haga referencia al alias.

Si se especifica PUBLIC, el *nombre-alias* debe identificar un alias público que existe en el servidor actual (SQLSTATE 42704).

Si se hace referencia al alias en la definición de un permiso de fila o una máscara de columna, el alias no se puede descartar (SQLSTATE 42893).

AUDIT POLICY *nombre-política*

Identifica la política de auditoría que se va a descartar. El *nombre-política* debe identificar una política de auditoría que exista en el servidor actual (SQLSTATE 42704). La política de auditoría no debe asociarse con ningún objeto de base de datos (SQLSTATE 42893). La política de auditoría especificada se suprime del catálogo.

BUFFERPOOL *nombre-agrup-almac-interm*

Identifica la agrupación de almacenamientos intermedios que se debe descartar. El *nombre-agrup-almac-interm* debe identificar una agrupación de almacenamientos intermedios que se haya descrito 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 descartar (SQLSTATE 42832). La memoria de la agrupación de almacenamientos intermedio se libera inmediatamente, para que DB2 pueda utilizarla. Puede que el almacenamiento en disco no se libere hasta que se produzca la siguiente conexión con la base de datos.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*

Identifica el grupo de particiones de base de datos que va a descartarse. El parámetro *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos que se haya descrito en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

Al descartar un grupo de particiones de base de datos se descartan todos los espacios de tablas que se han definido en el grupo de particiones de base de datos. Todos los objetos de base de datos que existan con dependencias en las tablas del espacio de tablas (por ejemplo, paquetes, restricciones de referencia, etcétera) se descartan o invalidan (según sea adecuado), y las vistas y los activadores dependientes no estarán operativos.

Los grupos de particiones de base de datos IBMCATGROUP, IBMDEFAULTGROUP e IBMTEMPGROUP no se pueden descartar (SQLSTATE 42832).

Si se emite una sentencia DROP DATABASE PARTITION GROUP para un grupo de particiones de base de datos que actualmente está sometiéndose a una redistribución de datos, la operación de descartar el grupo de particiones de base de datos no se realiza satisfactoriamente y se devuelve un error (SQLSTATE 55038). Sin embargo, puede descartarse un grupo de particiones de base de datos redistribuido parcialmente. Un grupo de particiones de base de datos puede redistribuirse parcialmente si no se completa la ejecución de un mandato REDISTRIBUTE DATABASE PARTITION GROUP. Esto puede suceder si se interrumpe a consecuencia de un error o por haberse emitido un mandato FORCE APPLICATION ALL. (Para un grupo de particiones de base de datos redistribuido parcialmente, el REDISTRIBUTE_PMAP_ID del catálogo SYSCAT.DBPARTITIONGROUPS no es -1.)

EVENT MONITOR *nombre-supervisor-sucesos*

Identifica el supervisor de sucesos que se debe descartar. El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que se haya descrito en el catálogo (SQLSTATE 42704).

Si el supervisor de sucesos identificado está activo, se devuelve un error (SQLSTATE 55034); de lo contrario, el supervisor de sucesos se suprime. Tenga en cuenta que si un supervisor de sucesos se ha activado previamente mediante la sentencia SET EVENT MONITOR STATE y la base de datos se ha desactivado y se ha vuelto a activar, debe utilizar la sentencia SET EVENT MONITOR STATE para desactivar el supervisor de sucesos antes de emitir la sentencia DROP.

Si hay archivos de sucesos en la vía de acceso de destino de un supervisor de sucesos WRITE TO FILE que se está descartando, los archivos de sucesos no se suprimirán. Sin embargo, si se crea un nuevo supervisor de sucesos que especifique la misma vía de acceso de destino, se suprimirán los archivos de sucesos.

Cuando se descartan supervisores de sucesos WRITE TO TABLE, se elimina la información de tabla de la vista de catálogo SYSCAT.EVENTTABLES, pero no se descartan las tablas en sí.

designador-función

Identifica una instancia de una función definida por el usuario (una función completa o una plantilla de función) que se debe descartar. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

La instancia de función especificada debe ser una función definida por el usuario descrita en el catálogo. Las funciones siguientes no se pueden descartar:

- Una función generada implícitamente por una sentencia CREATE TYPE (SQLSTATE 42917)
- Una función que se encuentra en el esquema SYSIBM, SYSFUN, SYSIBMADM o SYSPROC (SQLSTATE 42832)
- Una función a la que se hace referencia en la definición de un permiso de fila o una máscara de columna (SQLSTATE 42893)

RESTRICT

La palabra clave RESTRICT aplica la norma que establece que la función no se descartará si existe alguna de las dependencias siguientes:

- Otra función tiene su origen en la propia función.
- Otra rutina utiliza la función.
- Una vista utiliza la función.
- Un activador utiliza la función.
- Una tabla de consulta materializada utiliza la función de su definición.

La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.

En este caso, se aplican las consideraciones siguientes:

- Otros objetos pueden depender de una función. Deben eliminarse todas estas dependencias antes de que pueda descartarse la función, a excepción de los paquetes que están marcados como no operativos. El intento de descartar una función con dichas dependencias dará como resultado un

error (SQLSTATE 42893). Consulte la sección “Normas” para obtener una lista de estas dependencias. Si la función puede descartarse, se descarta.

- Cualquier paquete dependiente de la función específica que se está descartando se marca como no operativo. Dicho paquete no se vuelve a enlazar implícitamente. Deberá volver a enlazarse mediante la utilización del mandato BIND o REBIND o deberá volver a prepararse mediante la utilización del mandato PREP.

No es posible descartar una función que se encuentre en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

Si se hace referencia a la función en la definición de un permiso de fila o una máscara de columna, la función no se puede descartar (SQLSTATE 42893).

FUNCTION MAPPING *nombre-correlación-funciones*

Identifica la correlación de funciones que se debe descartar. El *nombre-correlación-funciones* debe identificar una correlación de funciones definida por el usuario que se haya descrito en el catálogo (SQLSTATE 42704). La correlación de funciones se suprime de la base de datos.

No se pueden descartar las correlaciones de funciones por omisión, pero se pueden inhabilitar mediante la sentencia CREATE FUNCTION MAPPING. Si se descarta una correlación de funciones definida por el usuario que se había creado para alterar temporalmente una correlación de funciones por omisión, se restituye la correlación de funciones por omisión.

Los paquetes que tengan una dependencia de la correlación de funciones descartada se invalidarán.

HISTOGRAM TEMPLATE *nombre-plantilla*

Identifica la plantilla de histograma que se debe descartar. El *nombre-plantilla* debe identificar una plantilla de histograma que exista en el servidor actual (SQLSTATE 42704). El *nombre-plantilla* no puede ser SYSDEFAULTHISTOGRAM (SQLSTATE 42832). La plantilla de histograma no puede descartarse si de ella depende una clase de servicio o una acción de trabajo (SQLSTATE 42893). La plantilla de histograma especificada se suprime del catálogo.

INDEX *nombre-índice*

Identifica el índice o especificación de índice que se debe descartar. 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 el sistema necesite para una clave primaria o restricción de unicidad, para una tabla de consulta materializada duplicada o una columna XML (SQLSTATE 42917). Se suprime el índice especificado o la especificación de índice.

Los paquetes que tengan una dependencia de un índice o de una especificación descartada se invalidarán.

INDEX EXTENSION *nombre-extensión-índice* **RESTRICT**

Identifica la extensión de índice que se debe descartar. 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).

MASK *nombre-máscara*

Identifica la máscara de columna que se debe descartar. El nombre debe identificar una máscara de columna existente en el servidor actual (SQLSTATE 42704).

designador-método

Identifica un cuerpo de método que se debe descartar. Para obtener más

información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22. 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 descartar.

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 descartar el cuerpo de un método, se puede eliminar la especificación del método en la definición del tipo sujeto, mediante ALTER TYPE DROP METHOD.

RESTRICT

La palabra clave RESTRICT aplica la norma que establece que el método no se descartará si existe alguna de las dependencias siguientes:

- Una función tiene su origen en el propio método.
- Otra rutina utiliza el método.
- Una vista utiliza el método.
- Un activador utiliza el método.
- Una tabla de consulta materializada utiliza el método de su definición.

La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.

En este caso, se aplican las consideraciones siguientes:

- Otros objetos pueden depender de un método. Todas estas dependencias deben eliminarse para poder descartar el método, con la excepción de los paquetes, que se marcarán como no operativos si la operación de descartar es efectiva. El intento de descartar un método con dichas dependencias dará como resultado un error (SQLSTATE 42893). Si el método se puede descartar, se descartará.
- Los paquetes dependientes del método específico que se está descartando 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.

Si el método específico que se descarta altera temporalmente otro método, se invalidarán todos los paquetes que dependen del método alterado temporalmente, y que dependen de los métodos que alteran temporalmente este método en los supertipos del método específico que se descarta.

MODULE *nombre-módulo*

Identifica el módulo que se debe descartar. El *nombre-módulo* debe identificar un módulo que exista en el servidor actual (SQLSTATE 42704). El módulo especificado se descarta del esquema, incluidos todos los objetos del módulo. Se descartan también todos los privilegios del módulo.

Si se hace referencia al módulo en la definición de un permiso de fila o una máscara de columna, el módulo no se puede descartar (SQLSTATE 42893).

NICKNAME *apodo*

Identifica el apodo que va a descartarse. El apodo debe aparecer en la lista del 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 descarta cualquier tabla de consulta materializada que dependa del apodo. Se descarta cualquier especificación de índice que es

DROP

dependiente del apodo. Las vistas que dependen del apodo se marcan como no operativas. Se invalidan los paquetes que dependen de las especificaciones de índice o de las vistas no operativas que se han descartado. No afecta a la tabla de fuente de datos al que el apodo hace referencia.

Si una función o método de SQL depende de un apodo, ese apodo no se puede descartar (SQLSTATE 42893).

PACKAGE *nombre-esquema.id-paquete*

Identifica el paquete que se debe descartar. Si no se especifica un nombre de esquema, el esquema por omisión califica implícitamente el identificador del paquete. El nombre de esquema y el identificador de paquete, junto con el identificador de versión especificado implícita o explícitamente, deben identificar un paquete que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el paquete especificado. Si el paquete que está descartándose es el único paquete que *nombre-esquema.id-paquete* identifican (es decir, si no existen otras versiones), también se suprimen todos los privilegios para el paquete.

VERSION *id-versión*

Identifica qué versión del paquete va a descartarse. Si no se especifica un valor, la versión tomará por omisión una serie de caracteres vacía. Si existen varios paquetes con el mismo nombre de paquete pero con distinta versión, sólo puede descartarse una versión del paquete en cada invocación de la sentencia DROP. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

Si la sentencia se invoca desde el indicador de mandatos del sistema operativo, preceda cada delimitador de dobles comillas con una barra inclinada invertida para asegurar que el sistema operativo no divide los delimitadores.

PERMISSION *nombre-permiso*

Identifica el permiso de fila que se debe descartar. El nombre debe identificar un permiso de fila existente en el servidor actual (SQLSTATE 42704). El nombre no debe identificar el permiso de fila por omisión creado de forma implícita por la base de datos DB2 (SQLSTATE 42917).

designador-procedimiento

Identifica una instancia de un procedimiento que debe descartarse. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22. La instancia de procedimiento especificada debe ser un procedimiento descrito en el catálogo. No es posible descartar un procedimiento que se encuentre en el esquema SYSIBM, SYSFUN, SYSIBMADM o SYSPROC (SQLSTATE 42832).

RESTRICT

La palabra clave RESTRICT impide que se descarte el procedimiento si una definición de activador o una definición de rutina de SQL contiene una sentencia CALL que identifica al procedimiento. La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se cumplen las condiciones siguientes:

- Se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.

- Una definición de activador en línea, definición de función de SQL en línea o definición de método de SQL en línea contiene una sentencia CALL que identifica el procedimiento.

No es posible descartar un procedimiento que se encuentre en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

ROLE *nombre-rol*

Identifica el rol que debe descartarse. El *nombre-rol* debe identificar un rol que ya exista en el servidor actual (SQLSTATE 42704). El *nombre-rol* no debe identificar un rol o un rol que contenga *nombre-rol*, si el rol tiene privilegio EXECUTE sobre una rutina o privilegio USAGE sobre una secuencia y un objeto SQL que no sea un paquete depende de la rutina o secuencia (SQLSTATE 42893). El propietario del objeto SQL es *nombre-autorización* o cualquier usuario que sea miembro de *nombre-autorización*, donde *nombre-autorización* es un rol.

Una sentencia A DROP ROLE no se ejecuta correctamente (SQLSTATE 42893) si se da cualquiera de las condiciones siguientes para el rol que se desea descartar:

- Existe una carga de trabajo de modo que uno de los valores del atributo de conexión SESSION_USER ROLE es *nombre-rol*
- Existe un contexto fiable que utiliza *nombre-rol*

El rol especificado se suprime del catálogo.

SCHEMA *nombre-esquema* **RESTRICT**

Identifica el esquema en particular que se debe descartar. El *nombre-esquema* debe identificar un esquema que se describe en el catálogo (SQLSTATE 42704). La palabra clave RESTRICT impone la norma de que no puede haber ningún objeto definido en el esquema especificado para que se suprima de la base de datos (SQLSTATE 42893).

SECURITY LABEL *nombre-etiqueta-seguridad*

Identifica la etiqueta de seguridad que se descartará. Se debe calificar el nombre con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704).

RESTRICT

Esta opción, que es el valor por omisión, impide que se descarte la etiqueta de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

- Uno o varios ID de autorización tienen la etiqueta de seguridad para acceso de lectura
- Uno o varios Id de autorización tienen la etiqueta de seguridad para acceso de grabación
- La etiqueta de seguridad se utiliza para proteger una o varias columnas

SECURITY LABEL COMPONENT *nombre-componente-etiqueta-seguridad*

Identifica el componente de la etiqueta de seguridad que se descartará. El *nombre-comp-etiqueta-seg* debe identificar un componente de la etiqueta de seguridad que se describe en el catálogo (SQLSTATE 42704).

RESTRICT

Esta opción, que es el valor por omisión, impide que se descarte el componente de la etiqueta de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

DROP

- Una o varias políticas de seguridad que incluyen el componente de la etiqueta de seguridad están definidos

SECURITY POLICY *nombre-política-seguridad*

Identifica la política de seguridad que se descartará. El *nombre-política-seguridad* debe identificar una política de seguridad que exista en el servidor actual (SQLSTATE 42704).

RESTRICT

Esta opción, que es el valor por omisión, impide que se descarte la política de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

- Una o varias tablas están asociadas con esta política de seguridad
- Uno o varios ID de autorización tienen una exención con una de las normas de esta política de seguridad
- Una o varias etiquetas de seguridad están definidas para esta política de seguridad

SEQUENCE *nombre-secuencia*

Identifica la secuencia en particular que se debe descartar. 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 implícita o explícitamente, se devuelve un error (SQLSTATE 42704).

RESTRICT

La palabra clave RESTRICT impide el descarte de la secuencia si existe cualquiera de las dependencias siguientes:

- Existe un activador que da lugar a que una expresión NEXT VALUE o PREVIOUS VALUE en el cuerpo del activador especifique la secuencia (SQLSTATE 42893).
- Existe una rutina de SQL que da lugar a que una expresión NEXT VALUE en el cuerpo de la rutina especifique la secuencia (SQLSTATE 42893).

La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se cumplen las condiciones siguientes:

- Se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.
- Una definición de activador en línea, definición de función de SQL en línea o definición de método de SQL en línea hace referencia a la secuencia.

SERVER *nombre-servidor*

Identifica la fuente de datos cuya definición se debe descartar 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 descartan todos los apodos para tablas y vistas que residen en la fuente de datos. Se descarta cualquier especificación de índice dependiente de estos apodos. También se descarta 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 descartada. Se invalidan todos los paquetes dependientes de la definición de servidor, correlaciones de función, apodos y especificaciones de índices descartados. Todos los procedimientos federados que dependen de la definición del servidor también se descartan.

*designador-clase-servicio***SERVICE CLASS** *nombre-clase-servicio*

Identifica la clase de servicio que debe descartarse. El *nombre-clase-servicio* debe identificar una clase de servicio que esté descrita en el catálogo (SQLSTATE 42704). Para descartar una subclase de servicio, el *nombre-superclase-servicio* debe especificarse utilizando la cláusula UNDER.

UNDER *nombre-superclase-servicio*

Especifica la superclase de servicio de la subclase de servicio al descartar una subclase de servicio. El *nombre-superclase-servicio* debe identificar una superclase de servicio que esté descrita en el catálogo (SQLSTATE 42704).

RESTRICT

Esta palabra clave aplica la norma que establece que la clase de servicio no se descartará si existe alguna de las dependencias siguientes:

- La clase de servicio es una superclase de servicio y existe una subclase de servicio definida por el usuario bajo la clase de servicio (SQLSTATE 5U031). Primero debe descartarse la subclase de servicio.
- La clase de servicio es una superclase de servicio y existe una correlación de conjuntos de acciones de trabajo con la clase de servicio (SQLSTATE 5U031). Primero debe descartarse el conjunto de acciones de trabajo.
- La clase de servicio es una subclase de servicio y existe una correlación de acciones de trabajo con la clase de servicio (SQLSTATE 5U031). Primero debe descartarse la acción de trabajo.
- La clase de servicio tiene una correlación de cargas de trabajo (SQLSTATE 5U031). Primero debe eliminarse la correlación. Elimine la correlación de cargas de trabajo descartando la carga de trabajo o modificando la carga de trabajo para que no se correlacione con la clase de servicio.
- La clase de servicio tiene un umbral asociado (SQLSTATE 5U031). Primero debe descartarse el umbral.
- La clase de servicio es el destino de una acción REMAP ACTIVITY en un umbral (SQLSTATE 5U031). Modifique el umbral para establecer otra subclase de servicio como destino de la acción REMAP ACTIVITY o descarte el umbral.
- No se inhabilita la clase de servicio (SQLSTATE 5U031). Primero debe inhabilitarse la clase de servicio.

RESTRICT es el comportamiento por omisión.

STOGROUP *nombre-grupo-almacenamiento*

Identifica el grupo de almacenamiento que se debe descartar; *nombre-grupo-almacenamiento* debe identificar un grupo de almacenamiento que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte.

RESTRICT

La palabra clave RESTRICT impide que se descarte el grupo de almacenamiento si existe un espacio de tablas que utiliza el grupo de almacenamiento (SQLSTATE 42893). RESTRICT es el comportamiento por omisión.

DROP

No se puede descartar el grupo de almacenamiento por omisión actual (SQLSTATE 42893). Se puede designar un valor por omisión nuevo con la sentencia ALTER STOGROUP.

No se puede ejecutar la sentencia DROP STOGROUP mientras se está añadiendo un servidor de particiones de base de datos (SQLSTATE 55071).

TABLE *nombre-tabla*

Identifica la tabla base, la tabla temporal creada, o la tabla temporal declarada que se debe descartar. El *nombre-tabla* debe identificar una tabla que esté descrita en el catálogo, si se trata de 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 descartarse todas las subtablas antes de poder descartar una supertabla (SQLSTATE 42893). El *nombre-tabla* no debe identificar una tabla de catálogo (SQLSTATE 42832), o una tabla histórica asociada con una tabla temporal de período de sistema (SQLSTATE 42893). La tabla especificada se suprime de la base de datos.

Se descartan todos los índices, las claves primarias, las claves foráneas, los permisos de fila (incluido el permiso de fila por omisión), las máscaras de columna, las restricciones de comprobación, las tablas de consulta materializada y las tablas de etapas que se definen en la tabla. Todas las vistas y activadores que hacen referencia a la tabla se establecen como no operativos, incluidas tanto la tabla a la que se hace referencia en la cláusula ON de la sentencia CREATE TRIGGER como todas las tablas a las que se hace referencia en las sentencias de SQL activadas. Todos los paquetes que dependen de cualquier objeto descartado o marcado como no operativo se invalidarán. Esto incluye los paquetes que dependan de cualquier supertabla por encima de la subtabla en la jerarquía. Todas las columnas referenciadas para las que la tabla descartada se ha definido 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 descarta una tabla esa clase. No obstante, los paquetes dependen de las tablas temporales creadas y se invalidan cuando se descarta una tabla.

En un sistema federado, puede descartarse una tabla remota que se ha creado utilizando un DDL transparente. Si se descarta una tabla remota también se descarta el apodo asociado a la misma y se invalidan los paquetes que dependen de ese apodo.

Cuando se descarta una subtabla de una jerarquía de tablas, las columnas asociadas a 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 descarta una subtabla, todas sus filas se suprimen en las supertablas. Ello puede provocar la activación de activadores o de restricciones de integridad referencial definidos para las supertablas.

Cuando se descarta una tabla temporal creada o una tabla temporal declarada, y su creación precedió a la unidad de trabajo activa o punto de salvaguarda, la tabla se descartará funcionalmente y la aplicación no podrá acceder a la tabla. Sin embargo, la tabla seguirá reservando parte del espacio de su espacio de tablas y evitará que el espacio de tablas USER TEMPORARY se descarte o que el grupo de particiones de base de datos del espacio de tablas USER TEMPORARY se redistribuya hasta que se confirme la unidad de trabajo o hasta que finalice el punto de salvaguarda. Si se descarta una tabla temporal creada o una tabla temporal declarada, se destruirán los datos de la tabla, con independencia de si DROP está confirmado o retrotraído.

Si el *nombre-tabla* es una tabla temporal de período de sistema, las tablas históricas asociadas y los índices definidos en la tabla histórica también se descartan. Para descartar una tabla temporal de período de sistema, el conjunto de privilegios también debe contener la autorización obligatoria para descartar la tabla histórica (SQLSTATE 42501).

Una tabla histórica asociada con una tabla temporal de período de sistema no se puede descartar de forma explícita con la sentencia DROP (SQLSTATE 42893). Una tabla histórica se descarta de forma implícita cuando se descarta la tabla temporal de período de sistema asociada.

Una tabla no podrá descartarse si tiene el atributo RESTRICT ON DROP.

Una tabla desenlazada recientemente inicialmente no es accesible. Esto impide que se lea, se modifique o se descarte la tabla hasta que la sentencia SET INTEGRITY se pueda ejecutar para actualizar de forma incremental las MQT o para completar el proceso de restricciones de clave foránea. Una vez que se ejecute la sentencia SET INTEGRITY en todas las tablas dependientes, la tabla es completamente accesible, su atributo desenlazado se restaura y se puede descartar.

Cuando se descarta una tabla, también se descartan todos los permisos de fila, incluido el permiso de fila por omisión, y las máscaras de columna que se crean para la tabla.

Si se hace referencia a la tabla en la definición de un permiso de fila o una máscara de columna, la tabla no se puede descartar (SQLSTATE 42893).

TABLE HIERARCHY *nombre-tabla-raíz*

Identifica la jerarquía de tablas con tipo que se debe descartar. El *nombre-tabla-raíz* debe identificar una tabla con tipo que es la 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 descartan todos los índices, tablas de consulta materializada, tablas de etapas, claves primarias, claves foráneas y restricciones de comprobación que hacen referencia a las tablas descartadas. Todas las vistas y activadores que hacen referencia a las tablas descartadas se hacen no operativas. Todos los paquetes que dependen de cualquier objeto descartado o marcado como no operativo se invalidarán. Todas las columnas de referencia para las que una de las tablas descartadas se define como ámbito de la referencia se quedan sin ámbito.

A diferencia de descartar una subtabla individual, descartar la jerarquía de tablas no produce la activación de los activadores 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 descartarán; el *nombre-espacio-tablas* debe identificar un espacio de tablas que se describe en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte. El *nombre-espacio-tablas* no debe identificar un espacio de tablas que contiene una tabla histórica, a menos que la tabla temporal de período de sistema con la que está asociado también se vaya a descartar (SQLSTATE 42893).

Los espacios de tablas no se descartarán (SQLSTATE 55024) si existe alguna tabla que almacene al menos una de sus partes en un espacio de tablas que se está descartando y tiene una o varias partes en otro espacio de tablas que no se

DROP

está descartando (estas tablas deben descartarse primero) o si alguna tabla que reside en el espacio de tablas tiene el atributo RESTRICT ON DROP.

Los objetos cuyos nombres contienen el prefijo 'SYS' son objetos incorporados y, salvo en el caso de los espacios de tablas SYSTOOLSPACE y SYSTOOLSTMPSPACE, no se pueden descartar (SQLSTATE 42832).

Un espacio de tablas temporal del sistema (SYSTEM TEMPORARY) no se puede descartar (SQLSTATE 55026) si es el único espacio de tablas temporal que existe en la base de datos. No se puede descartar un espacio de tablas USER TEMPORARY si hay una instancia de una tabla temporal creada o una tabla temporal declarada creada en el mismo (SQLSTATE 55039). Aunque se haya descartado una tabla temporal creada, se seguirá considerando que el espacio de tablas USER TEMPORARY se sigue utilizando hasta que se descarten todas las instancias de la tabla temporal creada. Las instancias de una tabla temporal creada se descartan cuando la sesión termina o cuando se hace referencia a la tabla temporal creada en la sesión. Aunque se haya descartado una tabla temporal declarada, se considerará al espacio de tablas USER TEMPORARY en uso hasta que se haya confirmado la unidad de trabajo que contiene la sentencia DROP TABLE.

Al descartar un espacio de tablas se descartan 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 descartan o invalidan (lo que sea adecuado) y las vistas y activadores dependientes pasan a estar no operativos.

Los contenedores creados por un usuario no se suprimen. Los directorios de la vía de acceso del nombre del contenedor creados por el gestor de bases de datos durante la ejecución de CREATE TABLESPACE se suprimen. Se suprimen todos los contenedores que están por debajo del directorio de bases de datos. Cuando se confirma la sentencia DROP TABLESPACE, se suprimen los contenedores de archivos DMS o los contenedores SMS del espacio de tablas especificado, si es posible. Si no es posible suprimirlos (por ejemplo, porque otro agente los tiene abiertos), los archivos se truncan a una longitud de cero. Una vez terminados todos los contenedores, o cuando se emite el mandato DEACTIVATE DATABASE, estos archivos de longitud cero se suprimen.

THRESHOLD *nombre-umbral*

Identifica el umbral que debe descartarse. El *nombre-umbral* debe identificar un umbral que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte. Los umbrales con una cola, por ejemplo TOTALSCPARTITIONCONNECTIONS y CONCURRENTDBCOORDACTIVITIES, deben inhabilitarse antes de que se puedan descartar (SQLSTATE 5U025). El umbral especificado se suprime del catálogo.

TRIGGER *nombre-activador*

Identifica el activador que se debe descartar. El *nombre-activadores* debe identificar un activador que se haya descrito en el catálogo (SQLSTATE 42704). Se suprime el activador especificado.

El descarte de activadores hace que algunos paquetes se marquen como no válidos.

Si *nombre-activador* especifica un activador INSTEAD OF en una vista, puede que otro activador dependa de éste a través de una actualización que debe realizarse para la vista.

TRANSFORM ALL FOR *nombre-tipo*

Indica que se deben descartar 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 descartan. 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. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704).

Si no existen transformaciones definidas para *nombre-tipo*, se devuelve 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 concreto, 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 sistema principal.

El grupo de transformación no se descarta 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 descartarse 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 descartan. 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. 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-activador*

Identifica el activador que se debe descartar. El *nombre-activadores* debe identificar un activador que se haya descrito en el catálogo (SQLSTATE 42704). Se suprime el activador especificado.

El descarte de activadores hace que algunos paquetes se marquen como no válidos.

Si *nombre-activador* especifica un activador INSTEAD OF en una vista, puede que otro activador dependa de éste a través de una actualización que debe realizarse para la vista.

TRUSTED CONTEXT *nombre-contexto*

Identifica el contexto fiable que debe descartarse. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). Si el contexto fiable se descarta mientras están activas las conexiones fiables para este contexto, dichas conexiones seguirán siendo fiables hasta que terminen o hasta que se produzca el siguiente intento de volver a utilizarlas. Si

DROP

se intenta cambiar el usuario de estas conexiones fiables, se devuelve un error (SQLSTATE 42517). El contexto fiable especificado se suprime del catálogo.

TYPE *nombre-tipo*

Identifica el tipo definido por el usuario que se debe descartar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. Para un tipo estructurado, también se descarta el tipo de referencia asociado. El *nombre-tipo* debe identificar un tipo definido por el usuario descrito en el catálogo.

RESTRICT

El tipo no se descarta (SQLSTATE 42893) si se da cualquiera de las condiciones 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 descartar 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 un método.
- El tipo es un tipo de parámetro o se utiliza en el cuerpo de un procedimiento SQL.
- 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 activador, una definición de vista o en una extensión de índice.

Si no se especifica RESTRICT, el comportamiento es el mismo que RESTRICT, excepto por las funciones y los métodos que utilizan el tipo. La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.

Funciones que utilizan el tipo: Si puede descartarse 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 descarta, o una referencia al tipo que se descarta, la siguiente sentencia DROP FUNCTION se ejecuta eficazmente:

```
DROP SPECIFIC FUNCTION SF
```

Es posible que esta sentencia también descarte en cascada las funciones dependientes. Si todas estas funciones también están en la lista que se debe

descartar debido a una dependencia del tipo definido por el usuario, la operación de descartar el tipo definido por el usuario será satisfactoria (de lo contrario, falla con SQLSTATE 42893).

Métodos que utilizan el tipo: Si puede descartarse 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 descarta, o una referencia al tipo que se descarta, 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 dar lugar a que la operación DROP TYPE no se ejecute satisfactoriamente.

Se invalidan todos los paquetes que dependen de métodos que se han definido en supertipos del tipo que está descartándose y que pueden elegirse para la alteración temporal.

Si se hace referencia al tipo en la definición de un permiso de fila o una máscara de columna, el tipo no se puede descartar (SQLSTATE 42893).

TYPE MAPPING *nombre-correlación-tipos*

Identifica la correlación de tipos de datos definida por el usuario que se debe descartar. 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 descartan objetos adicionales.

USAGE LIST *nombre-lista-uso*

Identifica la lista de uso que se debe descartar. El *nombre-lista-uso*, incluido el calificador implícito o explícito, debe identificar una lista de uso descrita en el catálogo (SQLSTATE 42704). La memoria asignada para la lista de uso se libera y no se encuentra bajo control de transacciones.

USER MAPPING FOR *nombre-autorización* | **USER SERVER** *nombre-servidor*

Identifica la correlación de usuarios que se debe descartar. 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 aparecer en la lista del 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 descartan objetos adicionales.

VARIABLE *nombre-variable*

Identifica la variable global que se debe descartar. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

Si se hace referencia a la variable en la definición de un permiso de fila o una máscara de columna, la variable no se puede descartar (SQLSTATE 42893).

RESTRICT

La palabra clave RESTRICT impide que se descarte la variable global si a esta se hace referencia en una definición de rutina de SQL, definición de

DROP

activador o definición de vista (SQLSTATE 42893). La norma restrictiva se aplicará por omisión a las mismas dependencias que las de la versión 9.5 si se cumplen las condiciones siguientes:

- Se ha inhabilitado el parámetro de configuración de base de datos **auto_reval**.
- Una definición de activador en línea, definición de función de SQL en línea, definición de método de SQL en línea o vista hace referencia a la variable.

VIEW *nombre-vista*

Identifica la vista que se debe descartar. 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 descartarse todas las subvistas antes de poder descartar una supervista (SQLSTATE 42893).

Se suprime la vista especificada. La definición de cualquier vista o activador que sea dependiente directa o indirectamente de esta vista se marca como no operativa. Se descarta cualquier tabla de consulta materializada o tabla de etapas que dependa de cualquier vista que se ha marcado como no operativa. Se invalidará cualquier paquete que dependa de una vista que se descarte 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 descartada se defina como ámbito de la referencia se quedan sin ámbito.

Si se hace referencia a la vista en la definición de un permiso de fila o una máscara de columna, la vista no se puede descartar (SQLSTATE 42893).

VIEW HIERARCHY *nombre-vista-raíz*

Identifica la jerarquía de vistas con tipo que se debe descartar. 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 activador que dependa, directa o indirectamente, de cualquiera de las vistas descartadas se marcará como no operativa. Cualquier paquete que dependa de cualquier vista o activador que se descarte o se marque como no operativo será inválido. Cualquier columna de referencia para la que una vista descartada o vista marcada como no operativa se define como ámbito de la referencia se queda sin ámbito.

WORK ACTION SET *nombre-conjunto-acciones-trabajo*

Identifica el conjunto de acciones de trabajo que se debe descartar. El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704). También se descartan todas las acciones de trabajo contenidas en *nombre-conjunto-acciones-trabajo*.

WORK CLASS SET *nombre-conjunto-clases-trabajo*

Identifica la clase de trabajo que se debe descartar. El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704). También se descartan todas las clases de trabajo contenidas en *nombre-conjunto-clases-trabajo*.

WORKLOAD *nombre-carga-trabajo*

Identifica la carga de trabajo que se debe descartar. Este nombre consta de una sola parte. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). No se puede descartar

SYSDEFAULTUSERWORKLOAD o SYSDEFAULTADMWORKLOAD (SQLSTATE 42832). Para poder descartar una carga de trabajo, es preciso inhabilitarla y que no tenga ninguna ocurrencia de carga de trabajo activa asociada (SQLSTATE 5U023). Para descartar una carga de trabajo con un umbral asociado (SQLSTATE 5U031), debe descartar el umbral en primer lugar. La carga de trabajo especificada se suprime del catálogo.

WRAPPER *nombre-derivador*

Identifica el derivador que se debe descartar. El *nombre-derivador* debe identificar un derivador que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el derivador.

Se descartan 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 derivador. También se descartan 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 descarta cualquier especificación de índice dependiente de estos apodos descartados y se marca como no operativa cualquier vista dependiente de estos apodos. Se invalidan todos los paquetes dependientes de los objetos descartados y vistas no operativas. Todos los procedimientos federados que dependen de la definición del servidor también se descartan.

XSRBJECT *nombre-objetoXSR*

Identifica el objeto XSR que se descartará. El *nombre-objetoXSR* debe identificar un objeto XSR que se describe en el catálogo (SQLSTATE 42704).

Se descartan las restricciones de comprobación que hacen referencia al objeto XSR. Se marcan como inoperativos todos los activadores y vistas que hacen referencia al objeto XSR. Los paquetes dependen de un objeto XSR descartado se invalidan.

En un entorno de base de datos particionada, se puede emitir esta sentencia sobre un objeto XSR conectándose a cualquier partición.

Normas

Dependencias: Tabla 31 en la página 1063 muestra las dependencias que tienen los objetos entre sí. No todas las dependencias se graban explícitamente en el catálogo. Por ejemplo, no existe ningún registro de las restricciones de las que depende un paquete. Se muestran cuatro tipos de dependencias distintas:

- R** Semántica de restricción. El objeto principal no puede descartarse mientras exista el objeto que depende de él.
- C** Semántica en cascada. El descarte del objeto principal hace que el objeto que depende de él (objeto dependiente) se descarte también. Sin embargo, si el objeto dependiente no puede descartarse debido a que tiene una dependencia de restricción en otro objeto, la operación de descartar el objeto principal fallará.
- X** Semántica de no operativo. El descarte 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 y revalidación automática. El descarte 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.

DROP

Un paquete que una función o un método utiliza, o que utiliza un procedimiento que se llama directa o indirectamente desde una función o desde un método, sólo volverá a validarse automáticamente si la rutina se ha definido como MODIFIES SQL DATA. Si la rutina no es MODIFIES SQL DATA, se devuelve un error (SQLSTATE 56098).

Por lo general, el gestor de la base de datos intenta revalidar los objetos no válidos la siguiente vez que se utiliza el objeto. No obstante, en los casos en que **auto_reval** se establece en IMMEDIATE, los objetos dependientes afectados se revalidarán inmediatamente después de volverse no válidos. A continuación, indicamos los casos:

- ALTER TABLE ... ALTER COLUMN
- ALTER TABLE ... DROP COLUMN
- ALTER TABLE ... RENAME COLUMN
- ALTER TYPE ... ADD ATTRIBUTE
- ALTER TYPE ... DROP ATTRIBUTE
- Cualquier sentencia CREATE que especifique "OR REPLACE"

Algunas de las dependencias que se muestran en la Tabla 31 en la página 1063 cambian a "A" (Semántica de invalidación/revalidación automática cuando el parámetro de configuración de la base de datos **auto_reval** se establece en IMMEDIATE o DEFERRED. En la Tabla 32 en la página 1070 se resumen los objetos dependientes afectados. Los objetos que figuran en la columna "Objetos dependientes afectados" se invalidarán cuando se ejecute la sentencia correspondiente que figura en la columna "Sentencia".

Algunos objetos y parámetros de la sentencia DROP no se muestran en la Tabla 31 en la página 1063 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.
- Los tipos de objeto de alias, agrupación de almacenamientos intermedios, clave de distribución, privilegio y procedimiento no dependen de la sentencia DROP.
- 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 SELECT 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 SELECT que hace referencia a un apodo para una tabla o vista dentro de estas fuentes de datos (SQLSTATE 55006).

DROP

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 el descarte de cualquier vista de la misma jerarquía de vistas invalidará el paquete.

- ³ Si un paquete, tabla de consulta materializada, tabla de etapas, vista o activador utiliza un alias, pasa a depender del alias y del objeto al que hace referencia el alias. Si el alias está en una cadena, se crea una dependencia en cada alias de la cadena.

Los propios alias no son dependientes de nada. Es posible definir un alias 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 Si se hace referencia al tipo definido por el usuario como tipo de parámetro de función o tipo de retorno, el tipo se descartará y sus datos de catálogo se mantendrán debido a la dependencia del parámetro de rutina. Un valor 'X' en la columna VALID de la vista de catálogo SYSCAT.DATATYPES indica este tipo descartado. La sentencia DROP FUNCTION suprimirá sus datos de catálogo si la sentencia DROP FUNCTION también ha descartado la última dependencia de parámetro de rutina de este tipo o la suprimirá una sentencia CREATE TYPE con el mismo nombre de esquema, nombre de módulo y nombre de tipo. Si el tipo definido por el usuario es un tipo estructurado, también se descartan todos los métodos asociados con el tipo.

- ⁶ Descartar un espacio de tablas o una lista de espacios de tablas hace que se descarten todas las tablas que están totalmente contenidas dentro de este espacio de tablas o lista. Sin embargo, si una tabla fragmenta espacios de tablas (índices, columnas largas o particiones de datos en distintos espacios de tablas) dichos espacios de tablas no se encuentran en la lista que se está descartando, los espacios de tablas no se pueden descartar mientras la tabla siga existiendo.

- 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 dará lugar a que se descarte una tabla de consulta materializada o a que una vista se convierta en no operativa. 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 el descarte 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 activador 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 activador o una vista incluye la utilización de 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 consulta materializada depende de las tablas o apodos subyacentes que se han especificado en la selección completa de la definición de tabla.

La semántica de la cascada se aplica a las tablas de consulta materializada dependientes.

Una subtabla depende de sus supertablas hasta la tabla raíz. Una supertabla no puede descartarse hasta que se han descartado todas sus subtablas.

Una tabla histórica depende de la tabla temporal de período de sistema a la que está asociada. La semántica en cascada se aplica a la tabla histórica cuando se descarta la tabla temporal de período de sistema de la que depende.

- 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. El descarte o la creación de un tipo estructurado que altera los subtipos de los que el paquete depende causa la invalidación.

Se invalidan todos los paquetes que dependen de métodos que se han definido en supertipos del tipo que está descartándose y que pueden elegirse para la alteración temporal.

- ¹³ Una restricción de comprobación o un activador depende de un tipo si el tipo se utiliza en cualquier parte dentro de la restricción o del activador. No hay dependencia de los subtipos de un tipo estructurado utilizado en un predicado TYPE dentro de una restricción de comprobación o un activador.

- 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 descartar una supervista hasta que se hayan descartado todas sus subvistas. Consulte el número ¹⁶ para obtener dependencias de vista adicionales.

- ¹⁶ Un activador o una vista también depende de la tabla de destino o vista de destino de una operación de eliminación de referencia o función Deref. Un activador o una vista con una cláusula FROM que incluya OUTER(Z)

DROP

- depende de todas las subtablas o subvistas de Z que existían en el momento de crearse el activador 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 El descarte de un servidor produce el descarte en cascada de las correlaciones de funciones y tipo de correlaciones creadas para ese servidor especificado.
- 20 Si la clave de distribución se define en una tabla en un grupo de particiones de base de datos de varias particiones, la clave de distribución es necesaria.
- 21 Si una función de tabla dependiente OLE DB tiene "R" objetos dependientes (véase DROP FUNCTION), el servidor no se puede descartar.
- 22 Una función o método SQL puede depender de los objetos referenciados por su cuerpo.
- 23 Cuando se descarta 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 sentencia REVOKE en un privilegio SELECT para una tabla o vista que se utiliza en el cuerpo de una función o método de SQL da lugar a que se realice un intento de descartar el cuerpo de la función o método, si el cuerpo de la función o método definido ya no dispone del privilegio SELECT. Si se utiliza un cuerpo de función o método de este tipo en el cuerpo de una vista, activador o método, no podrá descartarse y, como resultado de ello, la sentencia REVOKE quedará restringida. En otro caso, la revocación se propaga y descarta esas funciones.
- 26 Un activador depende de un activador INSTEAD OF cuando éste modifica la vista en la que se ha definido el activador INSTEAD OF y el activador INSTEAD OF se activa.

- 27 No se puede descartar una declaración de método de un método original al que otros métodos alteran temporalmente (SQLSTATE 42893).
- 28 Si el método del cuerpo de método que está creándose se ha declarado de modo que altere temporalmente a otro método, se invalidan todos los paquetes que dependen del método alterado temporalmente y de los métodos que alteran temporalmente a este método en los supertipos del método que está creándose.
- 29 Cuando se crea un nuevo subtipo de un tipo existente, se invalidan todos los paquetes que dependen de los métodos que se han definido en los supertipos del tipo que está creándose y que pueden elegirse para la alteración temporal (por ejemplo, no los métodos de mutación o de observación).
- 30 Si el método específico del cuerpo de método que está descartándose se ha declarado de modo que altere temporalmente a otro método, se invalidan todos los paquetes que dependen del método alterado temporalmente y de los métodos que alteran temporalmente a este método en los supertipos del método específico que está descartándose.
- 31 El SQL dinámico colocado en la antememoria tiene la misma semántica que los paquetes.
- 32 Cuando se descarta una tabla base remota utilizando la sentencia DROP TABLE, se descartan el apodo y la tabla base remota.
- 33 Una clave primaria o las claves exclusivas a las que una clave foránea no hace referencia no restringen la modificación de un nombre local de apodo ni del tipo local.
- 34 Un XSROBJECT puede ser inoperativo para la descomposición como consecuencia de los cambios realizados en una tabla asociada con el esquema XML para descomposición. Los cambios que pueden afectar a la descomposición son los siguientes: descartar la tabla o una columna de la tabla, o cambiar una columna de la tabla. El estado de descomposición de un esquema XML se puede restablecer ejecutando una sentencia ALTER XSROBJECT para habilitar o inhabilitar la descomposición del esquema XML.
- 35
- No se puede descartar una clase de servicio si está correlacionada con un umbral (SQLSTATE 5U031).
  - No se puede descartar una clase de servicio si está correlacionada con una carga de trabajo (SQLSTATE 5U031).
  - No se puede descartar una superclase de servicio hasta que se hayan descartado todas sus subclases de servicio definidas por el usuario (SQLSTATE 5U031).
  - No se puede descartar una superclase de servicio si está correlacionada con un conjunto de acciones de trabajo (SQLSTATE 5U031).
  - No se puede descartar una subclase de servicio si está correlacionada con una acción de trabajo (SQLSTATE 5U031).
- 36 No se puede descartar un conjunto de clases de trabajo hasta que se haya descartado el conjunto de acciones de trabajo definido en él.
- 37 Cuando se descarte el índice o la tabla, su lista de uso quedará invalidada en el catálogo. La revalidación tendrá lugar en la próxima activación de la lista o puede revalidarse explícitamente mediante el procedimiento ADMIN\_REVALIDATE\_DB\_OBJECTS.

## DROP

- 38 La revocación de un privilegio queda restringida si provoca que un objeto quede descartado o invalidado y un permiso o máscara depende de él. Por ejemplo, si tiene una vista que depende de una tabla, y un permiso o máscara que hace referencia a la vista, REVOKE SELECT en la tabla invalida la vista, aunque provoca un error.
- 39 Los paquetes se invalidan cuando una tabla en la que se ha definido el permiso habilitado tiene el control de acceso de nivel de fila activado. Los paquetes no se verán afectados cuando se descarta un permiso que está inhabilitado o que se ha definido en una tabla con el control de acceso a filas desactivado.
- 40 Los paquetes se invalidan cuando una tabla en la que se ha definido el permiso habilitado tiene el control de acceso de nivel de fila activado. Los paquetes no se verán afectados cuando se descarta un permiso que está inhabilitado o que se ha definido en una tabla con el control de acceso a filas desactivado.

Tabla 32. Objetos dependientes afectados por `auto_reval`

| Sentencia                                                      | Objetos dependientes afectados                                                                                                                 |
|----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| ALTER NICKNAME (modificación del nombre local o el tipo local) | Tipo de anclaje, función, método, procedimiento, tipo definido por el usuario, variable, vista                                                 |
| ALTER TABLE ALTER COLUMN                                       | Tipo de anclaje, función, método, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista, XSROBJECT              |
| ALTER TABLE DROP COLUMN <sup>2</sup>                           | Tipo de anclaje, función, método,, índice, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista, XSROBJECT     |
| ALTER TABLE RENAME COLUMN <sup>1, 3</sup>                      | Tipo de anclaje, función, método,, índice, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista, XSROBJECT     |
| ALTER TYPE ADD ATTRIBUTE                                       | Vista                                                                                                                                          |
| ALTER TYPE DROP ATTRIBUTE                                      | Vista                                                                                                                                          |
| DROP ALIAS                                                     | Tipo de anclaje, función, método, procedimiento, activador, tipo definido por el usuario, variable, vista                                      |
| DROP FUNCTION (ALTER MODULE DROP FUNCTION)                     | Función, correlación de función, extensión de índice, método, procedimiento, activador, variable, vista                                        |
| DROP METHOD                                                    | Función, correlación de función, extensión de índice, método, procedimiento, activador, variable, vista                                        |
| DROP NICKNAME                                                  | Tipo de anclaje, función, método, activador, tipo definido por el usuario, variable, vista                                                     |
| DROP PROCEDURE (ALTER MODULE DROP PROCEDURE)                   | Función, método, procedimiento, activador                                                                                                      |
| DROP SEQUENCE                                                  | Función, método, procedimiento, activador, variable, vista                                                                                     |
| DROP TABLE                                                     | Tipo de anclaje, función, método, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista, XSROBJECT              |
| DROP TABLE HIERARCHY                                           | Función, método, procedimiento, activador, variable, vista                                                                                     |
| DROP TRIGGER                                                   | Activador                                                                                                                                      |
| DROP TYPE (ALTER MODULE DROP TYPE)                             | Tipo de anclaje, tipo de cursor, función, método, procedimiento, extensión de índice, activador, tipo definido por el usuario, variable, vista |

Tabla 32. Objetos dependientes afectados por `auto_reval` (continuación)

| Sentencia                                  | Objetos dependientes afectados                                                                                                      |
|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| DROP VARIABLE (ALTER MODULE DROP VARIABLE) | Tipo de anclaje, función, correlación de funciones, método, procedimiento, activador, tipo definido por el usuario, variable, vista |
| DROP VIEW                                  | Tipo de anclaje, función, método, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista              |
| DROP VIEW HIERARCHY                        | Función, procedimiento, activador, variable, vista                                                                                  |
| DROP XSROBJECT                             | Activador, vista                                                                                                                    |
| RENAME TABLE                               | Tipo de anclaje, función, método, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista, XSROBJECT   |
| REVOKE un privilegio                       | Función, método, procedimiento, activador, variable, vista                                                                          |
| CREATE OR REPLACE ALIAS <sup>1</sup>       | Función, activador, procedimiento, variable, vista                                                                                  |
| CREATE OR REPLACE VIEW <sup>1</sup>        | Tipo de anclaje, función, método, procedimiento, activador <sup>4</sup> , tipo definido por el usuario variable, vista              |
| CREATE OR REPLACE FUNCTION <sup>1</sup>    | Función, correlación de función, extensión de índice, método, procedimiento, variable, vista                                        |
| CREATE OR REPLACE PROCEDURE <sup>1</sup>   | Función, método, procedimiento, activador                                                                                           |
| CREATE OR REPLACE NICKNAME <sup>1</sup>    | Función, método, procedimiento, variable, vista                                                                                     |
| CREATE OR REPLACE SEQUENCE <sup>1</sup>    | Función, método, procedimiento, activador, variable, vista                                                                          |
| CREATE OR REPLACE VARIABLE <sup>1</sup>    | Función, método, activador, tipo definido por el usuario, variable, vista                                                           |
| CREATE OR REPLACE TRIGGER <sup>1</sup>     | Activador                                                                                                                           |

<sup>1</sup> La semántica de revalidación inmediata se aplica a estas sentencias (para las sentencias CREATE, sólo si se especifica OR REPLACE) con independencia de la configuración del parámetro de configuración de base de datos `auto_reval`.

<sup>2</sup> Los objetos dependientes de la lista se revalidarán la siguiente vez que se utilice el objeto, salvo en el caso de los objetos siguientes, que se revalidarán inmediatamente como parte de la sentencia:

- ANCHOR TYPE
- CURSOR TYPE
- VIEW (donde la lista de selección sólo consta de SELECT \* y no contiene ninguna columna de vista definida explícitamente).

En el caso de una revalidación de vista inmediata, la lista de nombres de columnas para la lista de selección se volverá a establecer durante la revalidación.

<sup>3</sup> Los objetos dependientes de la lista volverá a validarse la próxima vez que se utilice el objeto, excepto en el caso de los objetos siguientes, que se revalidarán inmediatamente como parte de la sentencia:

- Tipo definido por el usuario
- VIEW (donde la lista de selección sólo consta de SELECT \* y no contiene ninguna columna de vista definida explícitamente).

En el caso de una revalidación de vista inmediata, la lista de nombres de columnas para la lista de selección se volverá a establecer durante la revalidación.

## DROP

- 4 Si la dependencia se debe a que el activador está definido en la tabla o la vista, se seguirá aplicando la semántica no operativa de la Tabla 1. Si la dependencia se debe a que el cuerpo del activador hace referencia a la tabla o la vista, se aplica la semántica de invalidación y revalidación automática.

La sentencia `DROP DATABASE PARTITION GROUP` puede resultar anómala (SQLSTATE 55071) si una petición para añadir un servidor de particiones de base de datos está pendiente o en curso. Esta sentencia puede también resultar anómala (SQLSTATE 55077) si se añade en línea un servidor de particiones de base de datos nuevo a la instancia y no todas las aplicaciones saben de la existencia del servidor de particiones de base de datos nuevo.

### Notas

- Es válido descartar una función definida por el usuario mientras se está utilizando. También, un cursor puede abrirse en una sentencia que contenga una referencia a una función definida por el usuario y mientras el cursor está abierto la función puede descartarse 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 descarte la función hasta que el paquete complete su unidad de trabajo actual. En dicho momento, se descarta 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 del cuerpo de una función (que es muy diferente de descartar 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.
- 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 estas describen 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 activadores, 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 descartando (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, no es posible descartar la propiedad de

transformación de la función, de lo contrario, los paquetes se marcarán como inoperativos, por el mero hecho que contienen invocaciones explícitas en una expresión SQL.

- Las secuencias creadas por el sistema para las columnas IDENTITY no pueden descartarse utilizando la sentencia DROP SEQUENCE.
- Cuando se descarta una secuencia, también se descartan todos los privilegios de ésta y los paquetes que hacen referencia a la secuencia se invalidan.
- Para apodos relacionales, la sentencia DROP NICKNAME de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las siguientes condiciones (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Para apodos no relacionales, la sentencia DROP NICKNAME dentro de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Una sentencia SELECT de la misma UOW ya hace referencia a un apodo al que se hace referencia en esta sentencia
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Una sentencia DROP SERVER (SQLSTATE 55006), o las sentencias DROP FUNCTION MAPPING o DROP TYPE MAPPING (SQLSTATE 55007) de una unidad de trabajo (UOW) determinada no pueden procesarse en bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o 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 uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- La sentencia DROP WORKLOAD no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.
- Una aplicación sólo puede emitir una de estas sentencias cada vez y sólo se permite una de estas sentencias dentro de una unidad de trabajo. Cada sentencia debe ir seguida de una sentencia COMMIT o ROLLBACK antes de que pueda emitirse otra de estas sentencias (SQLSTATE 5U021).

## DROP

- CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
- CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
- CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
- CREATE WORK ACTION, ALTER WORK ACTION o DROP (WORK ACTION)
- CREATE WORK CLASS, ALTER WORK CLASS o DROP (WORK CLASS)
- CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
- GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- **Invalidación modificable:** después del cambio o el descarte efectuado por las sentencias siguientes, proseguirá el acceso activo al objeto descartado o cambiado, hasta que finalice el acceso.
  - ALTER FUNCTION
  - ALTER MODULE ... DROP FUNCTION
  - ALTER MODULE ... DROP VARIABLE
  - ALTER TABLE ... DETACH PARTITION
  - ALTER VIEW
  - DROP ALIAS
  - DROP FUNCTION
  - DROP TRIGGER
  - DROP VARIABLE
  - DROP VIEW
  - Todas las sentencias CREATE OR REPLACE salvo CREATE OR REPLACE SEQUENCE.

Este es el caso cuando la variable de registro de la base de datos `DB2_DLL_SOFT_INVALID` está establecida en ON. Cuando está establecida en OFF, la operación de descartar o cambiar estos objetos sólo se completará después de que todo el acceso activo al objeto que se descarta o cambia haya finalizado.

- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
  - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
  - DISTINCT TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
  - DATA TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
  - SYNONYM puede especificarse en lugar de ALIAS
  - PROGRAM puede especificarse en lugar de PACKAGE
- **Invalidación de paquetes y sentencias colocadas dinámicamente en antememoria tras descartar permisos de fila o máscaras de columna:** Si el control de acceso de nivel de fila está activado en la tabla, al descartar un permiso de fila definido para esa tabla se invalidan todos los paquetes y las sentencias colocadas dinámicamente en antememoria que hacen referencia a la misma tabla. Si el control de acceso de nivel de columna está activado en la tabla, al descartar una máscara de columna habilitada definida para esa tabla se invalidan todos los paquetes y las sentencias colocadas dinámicamente en



antememoria que hacen referencia a la misma tabla. No existe invalidación por descartar máscaras o permisos inhabilitados.

- **Dependencia circular:** en el ejemplo siguiente existe una dependencia circular:

```
CREATE PERMISSION RP1 ON T1 FOR ROWS
 WHERE C1>(SELECT MAX(C1) FROM T2)
ENFORCED FOR ALL ACCESS
ENABLE;
```

```
CREATE PERMISSION RP2 ON T2 FOR ROWS
 WHERE C1>(SELECT MAX(C1) FROM T1)
ENFORCED FOR ALL ACCESS
ENABLE
```

Las sentencias DROP TABLE T1 y DROP TABLE T2 fallan porque **RP1** depende de **T2** y **RP2** depende de **T1**. El usuario con autorización SECADM debe descartar uno de los permisos de fila en primer lugar y, a continuación, emitir la sentencia **DROP TABLE**.

## Ejemplos

- *Ejemplo 1:* Descarte la tabla TDEPT.

```
DROP TABLE TDEPT
```

- *Ejemplo 2:* Elimine la vista VDEPT.

```
DROP VIEW VDEPT
```

- *Ejemplo 3:* El ID de autorización HEDGES intenta descartar un alias.

```
DROP ALIAS A1
```

El alias HEDGES.A1 se elimina de los catálogos.

- *Ejemplo 4:* Hedges intenta descartar un alias, pero especifica T1 como el nombre-alias, cuando T1 es el nombre de una tabla existente (no el nombre de un alias).

```
DROP ALIAS T1
```

Esta sentencia falla (SQLSTATE 42809).

- *Ejemplo 5:* descartar el grupo de particiones de base de datos BUSINESS\_OPS. Para descartar el grupo de particiones de base de datos, primero deben descartarse los dos espacios de tablas (ACCOUNTING y PLANS) del grupo de particiones de base de datos.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP DATABASE PARTITION GROUP BUSINESS_OPS
```

- *Ejemplo 6:* Pellow desea descartar la función CENTRE, que ha creado en su esquema PELLOW, usando la signatura para identificar la instancia de la función que se debe descartar.

```
DROP FUNCTION CENTRE (INT,FLOAT)
```

- *Ejemplo 7:* McBride desea descartar la función FOCUS92, que ha creado en el esquema PELLOW, utilizando el nombre específico para identificar la instancia de función que se debe descartar.

```
DROP SPECIFIC FUNCTION PELLOW.FOCUS92
```

- *Ejemplo 8:* Descarte la función ATOMIC\_WEIGHT del esquema CHEM, donde se sabe que sólo existe una función con ese nombre.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

- *Ejemplo 9:* Descarte el activador SALARY\_BONUS, que ha dado lugar a que los empleados de la condición especificada recibieran una bonificación en sus salarios.

## DROP

**DROP TRIGGER SALARY\_BONUS**

- *Ejemplo 10:* Descarte el tipo de datos diferenciado denominado shoesize, si no se utiliza actualmente.

**DROP TYPE SHOESIZE**

- *Ejemplo 11:* Descarte el supervisor de sucesos SMITHPAY.

**DROP EVENT MONITOR SMITHPAY**

- *Ejemplo 12:* Descarte el esquema del Ejemplo 2 bajo CREATE SCHEMA utilizando RESTRICT. Tenga en cuenta que primero debe descartarse la tabla llamada PART.

**DROP TABLE PART**

**DROP SCHEMA INVENTORY RESTRICT**

- *Ejemplo 13:* Macdonald desea descartar el procedimiento DESTROY, que ha creado en el esquema EIGLER, utilizando el nombre específico encontrado en el catálogo del sistema para identificar el procedimiento que se debe descartar.

**DROP SPECIFIC PROCEDURE EIGLER.SQL100506102825100**

- *Ejemplo 14:* Descarte 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. Descarte 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. Descarte la especificación de índice que se ha creado para dejar que optimizador conozca este índice.

**DROP INDEX INDEXSPEC**

- *Ejemplo 17:* Descarte del grupo de transformación MYSTRUCT1.

**DROP TRANSFORM MYSTRUCT1 FOR POLYGON**

- *Ejemplo 18:* Descarte 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:* Descarte la secuencia ORG\_SEQ, con restricciones.

**DROP SEQUENCE ORG\_SEQ**

- *Ejemplo 20:* Se ha creado la tabla remota EMPLOYEE en un sistema federado utilizando un DDL transparente. Ya no es necesario acceder a la tabla. Descarte la tabla remota EMPLOYEE.

**DROP TABLE EMPLOYEE**

- *Ejemplo 21:* Descarte la correlación de funciones BONUS\_CALC y restituya la correlación de funciones por omisión (si existe).

**DROP FUNCTION MAPPING BONUS\_CALC**

- *Ejemplo 22:* Descarte el componente de la etiqueta de seguridad LEVEL.

**DROP SECURITY LABEL COMPONENT LEVEL**

- *Ejemplo 23:* Descarte la etiqueta de seguridad EMPLOYEESECLABEL de la política de seguridad DATA\_ACCESS.

**DROP SECURITY LABEL DATA\_ACCESS.EMPLOYEESECLABEL**

- *Ejemplo 24:* Descarte la política de seguridad DATA\_ACCESS.

**DROP SECURITY POLICY DATA\_ACCESS**

- *Ejemplo 25:* Descarte el componente de la etiqueta de seguridad GROUPS.

**DROP SECURITY LABEL COMPONENT GROUPS**

- *Ejemplo 26:* Descarte el esquema XML EMPLOYEE que se encuentra en el esquema SQL HR.

**DROP XSROBJECT** HR.EMPLOYEE

- *Ejemplo 27:* descarte la subclase de servicio DOGSALES de la superclase de servicio PETALES.

**DROP SERVICE CLASS** DOGSALES **UNDER** PETALES

- *Ejemplo 28:* descarte la superclase de servicio PETALES, que no tiene subclases de servicio definidas por el usuario. La subclase por omisión para la clase de servicio PETALES se descarta automáticamente.

**DROP SERVICE CLASS** PETALES

- *Ejemplo 29:* descartar (DROP) el permiso P1.

**DROP PERMISSION** P1

- *Ejemplo 30:* descartar (DROP) la máscara M1.

**DROP MASK** M1

- *Ejemplo 31:* descartar un grupo de almacenamiento denominado TEST\_SG.

**DROP STOGROUP** TEST\_SG

- *Ejemplo 32:* descartar la lista de uso MON\_PAYROLL

**DROP USAGE LIST** MON\_PAYROLL

---

## END DECLARE SECTION

La sentencia END DECLARE SECTION marca el final de una sección de declaración de variables del lenguaje principal.

### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. No se debe especificar 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 normas del sistema principal. Indica el final de una sección de declaración de variables del lenguaje principal. Una sección de variables del lenguaje principal empieza con una sentencia BEGIN DECLARE SECTION.

Las sentencias BEGIN DECLARE SECTION y END DECLARE SECTION deben especificarse por pares y no pueden anidarse.

Las declaraciones de variables del lenguaje principal pueden especificarse utilizando la sentencia de SQL INCLUDE. De lo contrario, la sección de declaración de variables del lenguaje principal no debe contener ninguna sentencia que no sean declaraciones de variables del lenguaje principal.

Las variables del lenguaje principal a las que se hace referencia en las sentencias de SQL se deben declarar en una sección de declaración de variable del lenguaje principal en todos los lenguajes de sistema principal, distintos de REXX. 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.

## EXECUTE

La sentencia EXECUTE ejecuta una sentencia de SQL preparada.

### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Para todas las variables globales utilizadas como *expresión* en la cláusula USING o en la expresión para un *índice-matriz*, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes:

- el privilegio READ sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

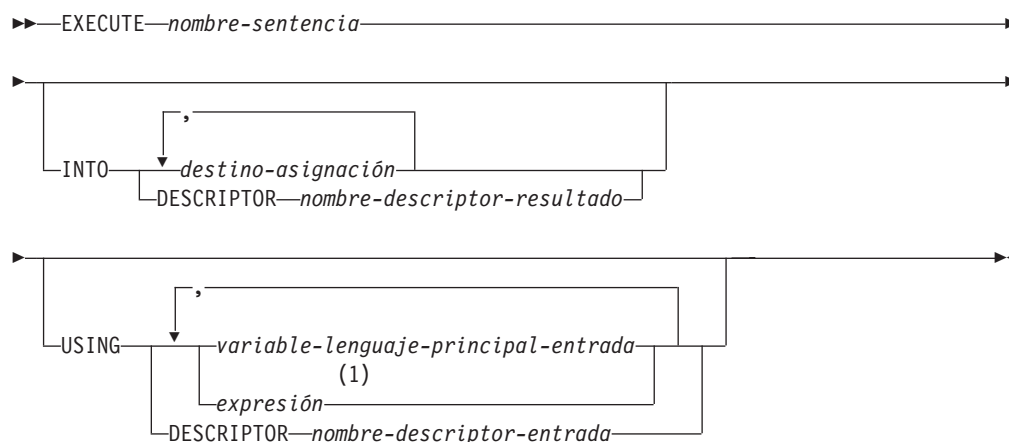
Para todas las variables globales utilizadas como *destino-asignación*, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes:

- el privilegio WRITE sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

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 podría verse afectado por la opción de enlace DYNAMICRULES.

Para las sentencias en las que la comprobación de la autorización se lleva a cabo en el momento de preparar la sentencia (DML), no se ejecuta ninguna comprobación de autorización adicional en la sentencia de SQL especificada por la sentencia PREPARE.

### Sintaxis



**destino-asignación:**

|                                                        |  |
|--------------------------------------------------------|--|
| <i>nombre-variable-global</i>                          |  |
| <i>nombre-variable-lenguaje-principal</i>              |  |
| <i>nombre-parámetro-SQL</i>                            |  |
| <i>nombre-variable-SQL</i>                             |  |
| <i>nombre-variable-transición</i>                      |  |
| <i>nombre-variable-matriz</i> [ <i>índice-matriz</i> ] |  |
| <i>referencia-campo</i>                                |  |

**Notas:**

- 1 Una expresión distinta de la *variable-lenguaje-principal* sólo se puede utilizar cuando la sentencia EXECUTE se emplea dentro de una sentencia de SQL compuesto (compilado).

**Descripción***nombre-sentencia*

Identifica la sentencia preparada que se debe ejecutar. El *nombre-sentencia* debe identificar una sentencia que se ha preparado anteriormente, y la sentencia que se ha preparado anteriormente no puede ser una sentencia SELECT.

**INTO**

Incluye una lista de destinos que se utilizan para recibir valores de los marcadores de parámetro de salida en la sentencia preparada. Cada asignación a un destino se realiza secuencialmente y siguiendo la lista. Si se produce un error en cualquier asignación, no se asigna el valor al destino y no se asignan más valores a los destinos. Cualquier valor que ya se haya asignado a los destinos continúa asignado.

En el caso de una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos OUT e INOUT para el procedimiento son marcadores de parámetro de salida. Si en la sentencia aparece algún marcador de parámetro de salida, debe especificarse la cláusula INTO (SQLSTATE 07007).

*destino-asignación*

Identifica uno o varios destinos para la asignación de los valores de salida. El primer valor de la fila del resultado se asigna al primer destino de la lista, el segundo valor al segundo destino, etcétera.

Si el tipo de datos de un *destino-asignación* es un tipo de fila, debe haber exactamente un *destino-asignación* especificado (SQLSTATE 428HR), el número de columnas debe coincidir con el número de campos en el tipo de fila y los tipos de datos de las columnas de la fila captada deben poder asignarse a los campos correspondientes del tipo de fila (SQLSTATE 42821).

Si el tipo de datos de un *destino-asignación* es un elemento de matriz, debe haber exactamente un *destino-asignación* especificado.

*nombre-variable-global*

Identifica la variable global que es el sujeto de la asignación.

*nombre-variable-lenguaje-principal*

Identifica la variable del lenguaje principal que es el sujeto de la asignación. Para los valores de salida LOB, el destino puede ser una variable del lenguaje principal normal (si es lo suficientemente grande), una variable de localizador LOB o una variable de referencia a archivos LOB.

*nombre-parámetro-SQL*

Identifica el parámetro de rutina que es el destino de la asignación.

*nombre-variable-SQL*

Identifica la variable de SQL que es el sujeto de la asignación. Las variables de SQL se deben declarar antes de utilizarlas.

*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 activador, calificado opcionalmente por un nombre de correlación que identifica el nuevo valor.

*nombre-variable-matriz*

Identifica una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz.

*índice-matriz*

Expresión que especifica qué elemento de la matriz será el destino de la asignación. Para una matriz común, la expresión de *índice-matriz* se debe poder asignar a INTEGER (SQLSTATE 428H1) y no puede ser un valor nulo. Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz (SQLSTATE 2202E). Para una matriz asociativa, la expresión de *índice-matriz* se debe poder asignar al tipo de datos de índice de la matriz asociativa (SQLSTATE 428H1) y no puede ser un valor nulo.

*referencia-campo*

Identifica el campo de un valor de tipo de fila que es el destino de la asignación. La *referencia-campo* debe especificarse como un *nombre-campo* calificado donde el calificador identifica el valor de fila donde está definido el campo.

**DESCRIPTOR** *nombre-descriptor-resultado*

Identifica una SQLDA de salida que debe contener una descripción válida de las 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.

SQLDA debe tener suficiente almacenamiento para contener todas las apariciones 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 aparición SQLVAR.

Si deben incluirse datos de salida de tipo de datos estructurados o LOB, deberán existir dos entradas de SQLVAR para cada marcador de parámetro de salida.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

## EXECUTE

### USING

Incluye una lista de variables o expresiones para los que se sustituyen valores para los marcadores de parámetro de entrada en la sentencia preparada.

En el caso de una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos IN e INOUT para el procedimiento son marcadores de parámetro de entrada. Para todas las demás sentencias dinámicas, todos los marcadores de parámetro son marcadores de parámetro de entrada. Si en la sentencia aparece algún marcador de parámetro de entrada, debe especificarse la cláusula USING (SQLSTATE 07004).

*variable-lenguaje-principal-entrada, ...*

Identifica una variable del lenguaje principal que se declara en el programa de acuerdo a las normas para la declaración de variables del lenguaje principal. El número de variables debe ser el mismo que el número de marcadores de parámetro de entrada de la sentencia preparada. La variable número *n* corresponde al marcador de parámetro número *n* de la sentencia preparada. Las variables de localizador y las variables de referencia a archivo, cuando deben utilizarse, pueden proporcionarse como la fuente de los valores de los marcadores de parámetro.

*expresión*

Identifica una expresión que se utilizará como entrada para el marcador de parámetro de entrada correspondiente de la sentencia preparada. Una expresión distinta de una *variable-lenguaje-principal* sólo se puede especificar cuando la sentencia EXECUTE se emite dentro de una sentencia de SQL compuesto (compilado).

**DESCRIPTOR** *nombre-descriptor-entrada*

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.

SQLDA debe tener suficiente almacenamiento para contener todas las apariciones 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 aparición SQLVAR.

Si deben incluirse datos de entrada de tipo de datos estructurados o LOB, deberán existir dos entradas de SQLVAR para cada marcador de parámetro.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

### Notas

- Antes de ejecutarse la sentencia preparada, cada marcador de parámetro de entrada se sustituye eficazmente por el valor de su variable o expresión correspondiente. Para un marcador de parámetro con tipo, los atributos de la expresión o variable de destino son aquellos determinados por la especificación



CAST. Para un marcador de parámetro sin tipo, los atributos de la expresión o variable de destino se determinan de acuerdo con el contexto del marcador de parámetro.

Supongamos que V indica una variable de entrada que se corresponde con el marcador de parámetro P. El valor de V se asigna a la variable de destino para P de acuerdo con las normas para asignar 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 ejecuta la sentencia preparada, el valor que se utiliza en el lugar de P es el valor de la variable de destino para P o el resultado de la expresión 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 V con dos espacios en blanco.

- Para una sentencia CALL dinámica, tras haberse ejecutado la sentencia preparada, el valor que se devuelve de cada argumento OUT e INOUT se asigna a la asignación de destino que corresponde con el marcador de parámetro de salida que se ha utilizado para el argumento. Para un marcador de parámetro con tipo, los atributos de la variable de destino son aquellos determinados por la especificación CAST. Para un marcador de parámetro sin tipo, los atributos de la variable de destino son los que especifica la definición del parámetro del procedimiento.

Supongamos que V es un destino de asignación de salida que corresponde con el marcador de parámetro P, el cual se utiliza para el argumento A de un procedimiento. El valor de A se asigna a V de acuerdo con las normas para la recuperación de un valor de una columna. Por lo tanto:

- V debe ser compatible con A.
- Si V es una serie de caracteres, su longitud no debe ser menor que la longitud de A o el valor de A se truncará.
- Si V es un número, el valor absoluto máximo de su parte integrante no debe ser menor que el valor absoluto de la parte integrante de A.
- Si los atributos de V no son idénticos a los atributos de A, el valor de A se convierte para ajustarse a los atributos de V.

- **Almacenamiento en antememoria de sentencias de SQL dinámico:** La información necesaria para ejecutar sentencias de SQL dinámico 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 estático o cuando se preparan por primera vez las sentencias de SQL dinámico. 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 de la sentencia de SQL individual también se coloca en la antememoria: las secciones de SQL estático se leen desde el catálogo del sistema y se colocan en la antememoria del paquete cuando se hace referencia por primera vez a la sentencia; las secciones de SQL dinámico se colocan directamente en la antememoria tras haberse creado. Las secciones de SQL dinámico pueden crearse mediante una sentencia explícita

## EXECUTE

como, por ejemplo, PREPARE o EXECUTE IMMEDIATE. Una vez creadas, las secciones de las sentencias de SQL dinámico pueden volver a crearse por medio de una preparación implícita de la sentencia que realiza el sistema si la sección original se ha suprimido por razones relacionadas con la gestión del espacio o si ha dejado de ser válida debido a que se han realizado cambios en el entorno.

Cada sentencia de SQL se coloca en antememoria en el nivel de la base de datos y las aplicaciones pueden compartirla. Las sentencias de SQL estático se comparten entre las aplicaciones que utilizan el mismo paquete; las sentencias de SQL dinámico se comparten entre las aplicaciones que utilizan el mismo entorno de compilación y exactamente el mismo texto de sentencia. El texto de cada sentencia de SQL que una aplicación emite se coloca en antememoria localmente dentro de la aplicación para poder utilizarlo si se necesita 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 colocada en antememoria para todas estas sentencias EXECUTE IMMEDIATE al mismo tiempo. 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 consiste en emitir varias sentencias PREPARE distintas una vez durante el inicio de la aplicación y en emitir posteriormente una sentencia EXECUTE o OPEN en función de las necesidades.

Cuando se colocan en antememoria sentencias de SQL dinámico, se puede reutilizar una sentencia en varias unidades de trabajo sin tener que volver a preparar la sentencia, a menos que las sentencias SQL preparadas en un paquete estén vinculadas con la opción KEEP DYNAMIC NO. El sistema vuelve a compilar la sentencia, si es necesario, cuando se producen cambios en el entorno.

Los sucesos siguientes son ejemplos de cambios en el entorno o en objetos de datos que pueden dar lugar a que las sentencias dinámicas colocadas en antememoria se preparen implícitamente en la siguiente petición PREPARE, EXECUTE, EXECUTE IMMEDIATE u OPEN:

- ALTER FUNCTION
- ALTER METHOD
- ALTER NICKNAME
- ALTER PROCEDURE
- ALTER SERVER
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE METHOD
- CREATE PROCEDURE
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP (todos los objetos)

- RUNSTATS en cualquier tabla o índice
- Cualquier acción que dé lugar a que una vista pase a ser 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 dinámico en antememoria:

- *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 devuelven 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 tiene que emitir una sentencia PREPARE después de una sentencia COMMIT o ROLLBACK, a menos que la sentencia esté asociada con un paquete vinculado con KEEP DYNAMIC NO.
- *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 invalidan las sentencias de SQL dinámico preparadas o implícitamente preparadas durante la unidad de trabajo afectada por la operación de retroacción. Las sentencias de SQL dinámico asociadas con un paquete vinculado con KEEP DYNAMIC NO se eliminan del contexto SQL de aplicación tras una operación ROLLBACK y deben volver prepararse de forma explícita para que la aplicación pueda ejecutarlas. Las sentencias de SQL dinámico se siguen colocando en antememoria a nivel de base de datos, por lo que una petición PREPARE posterior no incurre en el coste de compilación de la sentencia si la sección aún es válida.
- *COMMIT*: las sentencias de SQL dinámico no se invalidan, pero se libera cualquier bloqueo que se haya adquirido. Los cursores no definidos con la opción WITH HOLD se cierran y sus bloqueos se liberan. Los cursores definidos con la opción WITH HOLD mantendrán sus bloqueos de paquete y sección para proteger la sección activa durante el proceso de confirmación y después de éste. Las sentencias de SQL dinámico vinculadas con la opción KEEP DYNAMIC NO no están en estado preparado después de un límite de transacciones y deben volver a prepararse de forma explícita para que la aplicación pueda ejecutarlas. Las sentencias SELECT preparadas para un cursor abierto definido con la opción WITH HOLD permanecen en un estado preparado hasta que se alcanza un límite de transacción, en cuyo caso del cursor se cierra. Las

## EXECUTE

sentencias de SQL dinámico asociadas con un paquete vinculado con `KEEPDYNAMIC NO` se eliminan del contexto SQL de aplicación tras una operación de confirmación y deben volver prepararse de forma explícita para que la aplicación pueda ejecutarlas.

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 parámetro. Las variables del lenguaje principal `h1 - h4` corresponden al formato de `TDEPT`.

```
strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
.
.
(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
```

*Ejemplo 3:* Dado un procedimiento para otorgar una bonificación a un empleado:

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
IN DEPTNO INTEGER,
OUT CHEQUE INTEGER,
INOUT BONUS DEC(6,0))
...
```

Llame dinámicamente al procedimiento desde una aplicación en C. El procedimiento toma las siguientes variables del lenguaje principal como entrada:

- *employee*, el número de ID del empleado
- *dept*, el número del departamento
- *bonus*, la bonificación que ha de otorgarse al empleado.

El procedimiento devuelve los valores siguientes a las variables del lenguaje principal:

- *cheque\_no*, el número de ID del cheque
- *bonus*, el importe real de la bonificación (tras realizarse cualquier ajuste necesario)

```
strcpy (s, "CALL GIVE_BONUS(?, ?, ?, ?)");
EXEC SQL PREPARE DO_BONUS FROM :s;
.
.
/* Comprobar la ejecución satisfactoria y colocar valores en
:employee, :dept y :bonus */
.
EXEC SQL EXECUTE DO_BONUS INTO :cheque_no, :bonus
USING :employee, :dept, :bonus;
.
.
/* Comprobar la ejecución satisfactoria y procesar los
valores devueltos en :cheque_no y :bonus */
```

## EXECUTE IMMEDIATE

La sentencia EXECUTE IMMEDIATE prepara una forma ejecutable de una sentencia de SQL a partir de una forma de serie de caracteres de la sentencia y 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 parámetro.

### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

Las normas de autorización son las que se han definido para la sentencia de SQL especificada.

El ID de autorización de la sentencia podría verse afectado por la opción de enlace DYNAMICRULES.

### Sintaxis

► EXECUTE IMMEDIATE *expresión* ◄

### Descripción

*expresión*

Expresión que devuelve la serie de sentencia que se va a ejecutar. La expresión debe devolver un tipo de serie de caracteres que sea inferior al tamaño de sentencia máximo de 2.097.152 bytes. Tenga en cuenta que un CLOB(2097152) puede contener una sentencia de tamaño máximo, pero VARCHAR no puede.

La serie de caracteres de la sentencia debe ser una de las sentencias de SQL siguientes:

- ALTER
- CALL
- COMMENT
- COMMIT
- SQL compuesto (compilado)
- SQL compuesto (en línea)
- CREATE
- DECLARE GLOBAL TEMPORARY TABLE
- DELETE
- DROP
- EXPLAIN
- FLUSH EVENT MONITOR
- FLUSH PACKAGE CACHE
- GRANT
- INSERT

## EXECUTE IMMEDIATE

- LOCK TABLE
- MERGE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET COMPILATION ENVIRONMENT
- SET CURRENT DECFLOAT ROUNDING MODE
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT FEDERATED ASYNCHRONY
- SET CURRENT IMPLICIT XMLPARSE OPTION
- SET CURRENT ISOLATION
- SET CURRENT LOCALE LC\_MESSAGES
- SET CURRENT LOCALE LC\_TIME
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT MDC ROLLOUT MODE
- SET CURRENT OPTIMIZATION PROFILE
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET CURRENT TEMPORAL BUSINESS\_TIME
- SET CURRENT TEMPORAL SYSTEM\_TIME
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET ROLE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- SET SCHEMA
- SET SERVER OPTION
- SET SESSION AUTHORIZATION
- SET SQL\_CCFLAGS
- SET USAGE LIST STATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- SET variable
- TRANSFER OWNERSHIP (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- TRUNCATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)

- UPDATE

La serie de sentencia no debe incluir marcadores de parámetro ni referencias a variables del lenguaje principal, y no debe empezar por EXEC SQL. No debe contener un terminador de sentencia, a excepción de las sentencias de SQL compuesto, que pueden contener puntos y coma (;) para separar sentencias dentro del bloque compuesto. En algunas sentencias CREATE y ALTER se utiliza una sentencia de SQL compuesto, por lo que dichas sentencias también pueden contener puntos y coma.

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 en la SQLCA se informa acerca de la condición de error que evita que pueda ejecutarse. 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 almacenamiento de sentencias en antememoria afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE.

## 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 se encuentra 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 sobre el plan de acceso elegido para la sentencia explicable proporcionada y coloca esta información en las tablas de Explain.

Una *sentencia explicable* puede ser una sentencia XQuery válida o una de las siguientes sentencias de SQL: CALL, SQL compuesto (dinámico), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES o VALUES INTO.

## Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

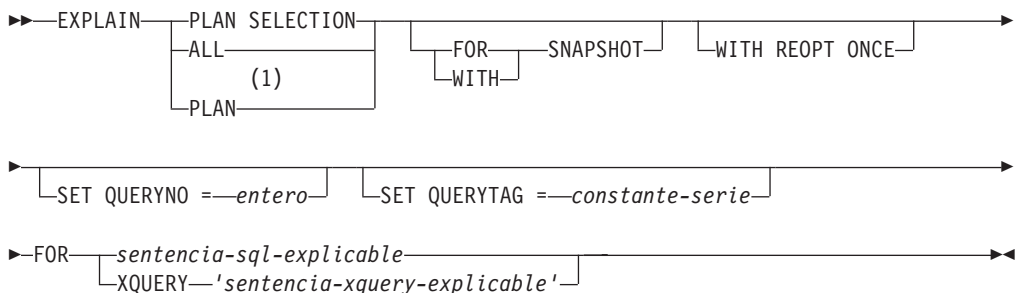
La sentencia que se debe explicar no se ejecuta.

## Autorización

El ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Autorización DATAACCESS que permite una sentencia INSERT, UPDATE, DELETE o SELECT.
- El privilegio INSERT sobre las tablas de Explain y al menos en una de las autorizaciones siguientes:
  - Todos los privilegios necesarios para ejecutar la sentencia explicable especificada en la sentencia EXPLAIN (por ejemplo, si se utiliza una sentencia DELETE como la sentencia explicable, se aplican las normas de autorización de la sentencia DELETE cuando ésta se explica).
  - Autorización EXPLAIN
  - Autorización SQLADM
  - Autorización DBADM

## Sintaxis



### Notas:

- 1 Sólo se da soporte a la opción PLAN para la tolerancia de la sintaxis de sentencias EXPLAIN existentes de DB2 para z/OS. 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 la compilación de consultas debe insertarse en las tablas de 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 debe tomarse una instantánea de explicación y colocarla en la columna SNAPSHOT de la tabla EXPLAIN\_STATEMENT. No se captura ninguna otra información distinta de la que se encuentra en las tablas EXPLAIN\_INSTANCE y EXPLAIN\_STATEMENT.

### WITH SNAPSHOT

Esta cláusula indica que, además de la información de explicación normal, se debe tomar una instantánea de explicación.

El comportamiento por omisión de la sentencia EXPLAIN es reunir únicamente información de explicación normal y no la instantánea de explicación.

### por omisión (no se especifican FOR SNAPSHOT ni WITH SNAPSHOT)

Coloca la información de explicación en las tablas de Explain.

### WITH REOPT ONCE

Esta cláusula indica que la *sentencia explicable* especificada se debe volver a optimizar mediante los valores para variables del lenguaje principal, marcadores de parámetro, registros especiales o variables globales que se utilizaron anteriormente para volver a optimizar esta sentencia con REOPT ONCE. Las tablas de Explain se llenarán con los datos del nuevo plan de acceso. Si el usuario tiene autorización DBADM, o la variable de registro de base de datos DB2\_VIEW\_REOPT\_VALUES se ha establecido en YES, la tabla EXPLAIN\_PREDICATE también se rellenará con los valores si se han utilizado para reoptimizar la sentencia.

### SET QUERYNO = entero

Asocia el *entero*, a través de la columna QUERYNO de la tabla EXPLAIN\_STATEMENT, con la *sentencia 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*, a través de la columna QUERYTAG de la tabla EXPLAIN\_STATEMENT, con la *sentencia explicable*. La *constante-serie* puede ser cualquier serie de caracteres de hasta 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.

## EXPLAIN

### **FOR** *sentencia-sql-explicable*

Especifica la sentencia de SQL que se debe explicar. Esta sentencia puede ser cualquier sentencia CALL, SQL compuesto (dinámico), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES o VALUES INTO SQL válida. Si la sentencia EXPLAIN está incorporada en un programa, la *sentencia-sql-explicable* puede contener referencias a las variables del lenguaje principal (estas variables deben estar definidas en el programa). De manera similar, si EXPLAIN se prepara de forma dinámica, la *sentencia-sql-explicable* puede contener marcadores de parámetro.

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 se pueden utilizar con esta sentencia.

Para explicar el SQL dinámico dentro de una aplicación, toda la sentencia EXPLAIN debe estar preparada dinámicamente.

### **FOR XQUERY** '*sentencia-xquery-explicable*'

Especifica la sentencia de XQUERY que se debe explicar. Esta sentencia puede ser cualquier sentencia XQUERY válida.

Si la sentencia EXPLAIN está incorporada en un programa, la '*sentencia-xquery-explicable*' puede contener referencias a variables del lenguaje principal, siempre y cuando las variables del lenguaje principal no se utilicen en el nivel superior de la sentencia XQUERY, sino que se pasen a través de una función XMLQUERY, mediante un predicado XMLEXISTS, o una función XMLTABLE. Las variables del lenguaje principal deben definirse en el programa.

De manera similar, si EXPLAIN se prepara de forma dinámica, la '*sentencia-xquery-explicable*' puede contener marcadores de parámetro, siempre y cuando se sigan las mismas restricciones que para pasar variables del lenguaje principal.

Como alternativa, se puede utilizar db2-fn:sqlquery de la función XQUERY de DB2 para incorporar sentencias de SQL con referencias a variables del lenguaje principal y marcadores de parámetro.

La *sentencia-xquery-explicable* debe ser una sentencia de XQUERY válida que podría prepararse y ejecutarse independientemente de la sentencia EXPLAIN. Las sentencias de consulta que hacen referencia a los cursores definidos a través de CLP no se pueden utilizar con esta sentencia.

## Notas

- El recurso Explain utiliza los ID siguientes como esquema al calificar tablas de Explain que se están llenando de datos:
  - El ID de autorización de sesión SQL dinámico
  - El ID de autorización de sentencia para SQL estático

El esquema se puede asociar con un conjunto de tablas de Explain o alias que apuntan a un conjunto de tablas de Explain en un esquema distinto. Si no se encuentran tablas de Explain en el esquema, el recurso Explain comprueba la existencia de tablas de Explain en el esquema SYSTOOLS e intenta utilizar dichas tablas.

- La tabla siguiente muestra la interacción de las palabras clave de instantánea y la información de explicación.

| Palabra clave especificada | ¿Capturar información de Explain? |
|----------------------------|-----------------------------------|
| ninguna                    | Sí                                |
| FOR SNAPSHOT               | No                                |
| WITH SNAPSHOT              | Sí                                |

Si no están especificadas las cláusulas FOR SNAPSHOT ni WITH SNAPSHOT, no se toma una instantánea de explicación.

- Antes de realizar la invocación de la sentencia EXPLAIN, el usuario deberá haber creado las tablas de Explain. La información que genera esta sentencia se almacena en las tablas de Explain, en el esquema que se ha designado en el momento de compilarse la sentencia.
- Si se producen errores durante la compilación de la *sentencia explicable* proporcionada, entonces no se almacena información en las tablas de Explain.
- El plan de acceso para la *sentencia explicable* no se guarda y, por consiguiente, no se podrá invocar posteriormente. La información de explicación para la *sentencia explicable* se inserta cuando se compila la sentencia EXPLAIN misma.
- Para una sentencia de consulta EXPLAIN estática, la información se inserta en las tablas de Explain en el momento de la vinculación o al volver a vincular explícitamente. 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 o PREP. Las sentencias EXPLAIN estáticas se pueden utilizar para hacer que las tablas de Explain se llenen para una sentencia de consulta estática específica entre muchas; simplemente, ponga un prefijo en la sentencia de destino con la sintaxis de sentencia EXPLAIN adecuada y vincule la aplicación sin utilizar las opciones BIND ni PREP de explicación. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de explicación real.
- Las sentencias EXPLAIN estáticas de un procedimiento de SQL se evalúan cuando se compila el procedimiento.
- Para las sentencias EXPLAIN de consulta con vinculación incremental, las tablas de Explain se llenan de datos cuando se envía a compilación la sentencia EXPLAIN. Cuando se ejecuta el paquete, la sentencia EXPLAIN no realiza ningún proceso (aunque se ejecutará correctamente). Cuando las tablas de Explain se llenan con datos, el calificador de la tabla de Explain y el ID de autorización utilizado durante el llenado de datos serán los pertenecientes 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 explicación real.
- Para sentencias EXPLAIN dinámicas, las tablas de Explain se llenan en el momento que la sentencia EXPLAIN se envía 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 consulta dinámica. La sentencia EXPLAIN debe utilizarse cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de EXPLAIN real.

## EXPLAIN

- Si la opción de vinculación REOPT se establece en ONCE y el registro especial CURRENT EXPLAIN MODE o CURRENT EXPLAIN SNAPSHOT se establece en REOPT, la ejecución de sentencias de consulta dinámica y estática que contienen variables del lenguaje principal, registros especiales, marcadores de parámetro o variables globales hará que sólo se capture la información de explicación para la sentencia cuando se vuelva a optimizar la sentencia. De forma alternativa, si la opción de vinculación REOPT se establece en ALWAYS, la información de explicación se capturará cada vez que se ejecuten estas sentencias.

### Ejemplos

- *Ejemplo 1:* Explique una sentencia SELECT simple y póngale el distintivo QUERYNO = 13.

```
EXPLAIN PLAN SET QUERYNO = 13
FOR SELECT C1
FROM T1
```

- *Ejemplo 2:* Explique una sentencia SELECT simple y póngale el distintivo QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
FOR SELECT C1
FROM T1
```

- *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
```

- *Ejemplo 4:* Intente obtener información de explicación cuando no existan tablas de Explain.

```
EXPLAIN ALL FOR SELECT C1
FROM T1
```

Esta sentencia fallará porque no se han definido las tablas de Explain (SQLSTATE 42704).

- *Ejemplo 5:* La siguiente sentencia será satisfactoria si se encuentra en la antememoria del paquete y ya se ha compilado utilizando REOPT ONCE.

```
EXPLAIN ALL WITH REOPT ONCE FOR SELECT C1
FROM T1
WHERE C1 = :<variable del lenguaje principal>
```

- *Ejemplo 6:* el ejemplo siguiente utiliza la función db2-fn:xmlcolumn, que toma el nombre que distingue entre mayúsculas y minúsculas de una columna XML como un argumento y devuelve una secuencia XML que es la concatenación de valores de columna XML.

Imaginemos una tabla denominada BUSINESS.CUSTOMER con una columna XML denominada INFO. Una XQuery simple que devuelve todos los documentos de la columna INFO es:

```
EXPLAIN PLAN SELECTION
FOR XQUERY 'db2-fn:xmlcolumn ("BUSINESS.CUSTOMER.INFO")'
```

Si un valor de la columna es nulo, la secuencia que se obtendrá para esa fila estará vacía.

## FETCH

La sentencia FETCH coloca un cursor en la siguiente fila de su tabla de resultados y asigna los valores de dicha fila a las variables de destino.

### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente. Cuando se invoca utilizando el procesador de línea de mandatos, la sintaxis que sigue a *nombre-cursor* es opcional y diferente de la sintaxis de SQL. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

### Autorización

En todas las variables globales utilizadas como *nombre-variable-cursor* o en la expresión para un *índice-matriz*, los privilegios con los que cuenta el ID de autorización de la sentencia deben incluir uno de los privilegios siguientes:

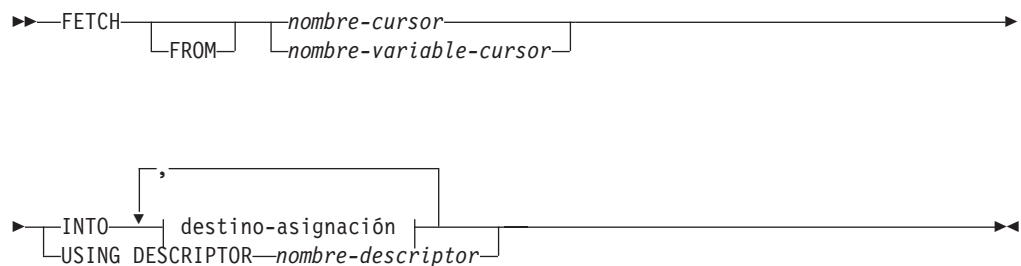
- el privilegio READ sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

En todas las variables globales utilizadas como *destino-asignación*, los privilegios con los que cuenta el ID de autorización de la sentencia deben incluir uno de los privilegios siguientes:

- el privilegio WRITE sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

### Sintaxis





*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 activador, calificado opcionalmente por un nombre de correlación que identifica el nuevo valor.

*nombre-variable-matriz*

Identifica una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz.

*[índice-matriz]*

Expresión que especifica qué elemento de la matriz será el destino de la asignación. Para una matriz común, la expresión de *índice-matriz* se debe poder asignar a INTEGER (SQLSTATE 428H1) y no puede ser un valor nulo. Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz (SQLSTATE 2202E). Para una matriz asociativa, la expresión de *índice-matriz* se debe poder asignar al tipo de datos de índice de la matriz asociativa (SQLSTATE 428H1) y no puede ser un valor nulo.

*referencia-campo*

Identifica el campo de un valor de tipo de fila que es el destino de la asignación. La *referencia-campo* debe especificarse como un *nombre-campo* calificado donde el calificador identifica el valor de fila donde está definido el campo.

**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 de SQLVAR, para indicar los atributos de las variables.

SQLDA debe tener suficiente almacenamiento para contener todas las apariciones 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 aparición 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 de resultados).

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

La variable número *n* descrita en el SQLDA corresponde a la columna número *n* de la tabla de resultados del cursor. El tipo de datos de cada variable debe ser compatible con su columna correspondiente.

Cada asignación que se realiza para una variable se realiza de acuerdo con normas específicas. 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 cuenta que no hay ningún aviso si hay más variables que el número de columnas del resultado.

## FETCH

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

- *Posición del cursor*: un cursor abierto tiene tres posiciones posibles:

- Antes de una fila
- En una fila
- Después de la última fila.

Un cursor sólo puede estar en una fila como resultado de una sentencia FETCH. Si el cursor está situado actualmente en la última fila o después de ella en la tabla de resultados:

- 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 destinos de asignación.

Si el cursor está situado actualmente antes de una fila, se volverá a situar en dicha fila y se asignarán los valores a los destinos tal como especifican las cláusulas 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 los destinos tal como se especifica en la cláusula INTO o USING.

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.

Es posible que se produzca un error que haga que el estado del cursor sea imprevisible.

- 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 FETCH no devuelva un aviso. También es posible que el aviso devuelto corresponda a una fila recuperada anteriormente. Esto se produce como resultado de las optimizaciones como, por ejemplo, la utilización de tablas temporales del sistema o de operadores de desplazamiento descendente.
- El almacenamiento de sentencias en antememoria afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE.
- DB2 CLI da soporte a capacidades de búsqueda adicionales. Por ejemplo cuando la tabla de resultados de un cursor es de sólo lectura, se puede utilizar la función SQLFetchScroll() para situar el cursor en cualquier punto dentro de dicha tabla de resultados.
- Para un cursor actualizable, se obtiene un bloqueo en una fila cuando se capta.
- Si la definición del cursor contiene una sentencia de cambio de datos de SQL o invoca una rutina que modifica datos de SQL, un error durante la operación de recuperación no hace que las filas modificadas se retrotraigan, aunque el error haga que se cierre el cursor.

### 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 BUFFERPOOLS

La sentencia FLUSH BUFFERPOOLS graba las páginas sucias de todas las agrupaciones de almacenamientos intermedios de una base de datos concreta de forma síncrona en el disco. En entornos DB2 pureScale, las páginas sucias de la agrupación de almacenamientos intermedios del grupo también se graban de forma síncrona en el disco.

La sentencia no está bajo el control de la transacción.

La sentencia FLUSH BUFFERPOOLS puede utilizarse de las formas siguientes:

- Para reducir el tiempo de recuperación de una base de datos en caso de que se produzca un error.
- Para reducir el tamaño de las anotaciones cronológicas grabadas en una imagen de copia de seguridad antes de las operaciones de base de datos, como las copias de seguridad en línea.
- Para minimizar el tiempo de recuperación de una base de datos de duplicación dividida.

### Invocación

La sentencia se puede incorporar a un programa de aplicación o emitir a través del uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que ostenta el ID de autorización de la sentencia deben incluir la autorización SQLADM, DBADM, SYSMANT, SYSCTRL o SYSADM.

### Sintaxis

```
▶▶ FLUSH [BUFFERPOOL | ALL | BUFFERPOOLS] ▶▶
```

### Descripción

#### ALL

Vacía las páginas sucias de todas las agrupaciones de almacenamientos intermedios (locales y grupos).

### Notas

- **Proceso de páginas sucias:** sólo se graban en el disco las páginas sucias que están en las agrupaciones de almacenamientos intermedios cuando la sentencia empieza el proceso. Las páginas sucias que se añaden a las agrupaciones de almacenamientos intermedios antes de que la sentencia acabe el proceso no se graban en el disco.
- **Sintaxis alternativa:** se puede especificar BUFFERPOOL en lugar de BUFFERPOOLS.

## 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.

En cualquier momento existe un registro de sucesos parcial disponible para los supervisores de sucesos que tienen una frecuencia de generación de registros baja (como un supervisor de sucesos de base de datos). Estos registros se indican en las anotaciones cronológicas 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 incorporarse 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 que tiene el ID de autorización de la sentencia deben incluir la autorización SQLADM o DBADM.

### Sintaxis

```
▶▶—FLUSH—EVENT—MONITOR—nombre-supervisor-sucesos—BUFFER—▶▶
```

### Descripción

*nombre-supervisor-sucesos*

Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador ordinario.

#### **BUFFER**

Indica que se deben 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.
- La sentencia FLUSH EVENT MONITOR no provoca que se generen y graben errores para el supervisor de sucesos UNIT OF WORK.

# FLUSH FEDERATED CACHE

La sentencia FLUSH FEDERATED CACHE vacía la antememoria federada, lo que permite la obtención de nuevos metadatos la próxima vez que se emita una sentencia de SQL para la vista o tabla remota utilizando un nombre de tres partes federado.

Cuando se emite una sentencia de SQL para una tabla o vista remota utilizando un nombre de tres partes federado, si se hace referencia a la tabla o vista remota por primera vez, se recuperan los metadatos y estadísticas para el objeto remoto y éstos se almacenan en una antememoria federada.

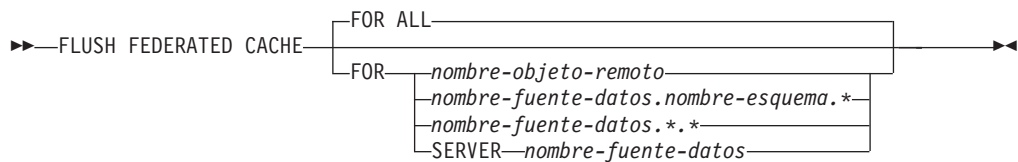
## Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. 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 incluir la autorización SQLADM o DBADM.

## Sintaxis



## Descripción

### FOR ALL

Vacía la información de la antememoria federada para todos los objetos de todas las fuentes de datos. Es el valor por omisión.

### FOR *nombre-objeto-remoto*

Vacía la información de la antememoria federada para una tabla o vista remota específica.

### FOR *nombre-fuente-datos.nombre-esquema.\**

Vacía la información de la antememoria federada para todos los objetos del esquema que se identifica mediante *nombre-esquema* de la fuente de datos específica que se identifica mediante *nombre-fuente-datos*.

### FOR *nombre-fuente-datos.\*.\**

Vacía la información de la antememoria federada para todos los objetos de las fuentes de datos específicas que se identifican mediante *nombre-fuente-datos*.

### FOR SERVER *nombre-fuente-datos*

Vacía la información de la antememoria federada para todos los objetos de las fuentes de datos específicas que se identifican mediante *nombre-fuente-datos*.

## Notas

- **Invalidación de paquete:** El vaciado de la antememoria federada da lugar a que se invaliden los paquetes que dependen del nombre de tres partes. Esta acción

puede tener un impacto en el rendimiento, pues los paquetes invalidados han de volver a compilarse siempre que se ejecutan sentencias del paquete.

- **Invalidación de vista:** El vaciado de la antememoria federada no dará lugar a que se invaliden las vistas que dependen del nombre de tres partes. La próxima vez que se utilice la vista, revalidará la vista implícitamente. Si se han realizado cambios en el objeto remoto, es posible que la sentencia que utiliza la vista devuelva un error.

### Ejemplos

- *Ejemplo 1:* Vaciar la información de la antememoria federada para el nombre-tabla-remota *t1* en el nombre-esquema-remoto *nombre\_esquema\_remoto* en la fuente de datos *rudb*.

```
FLUSH FEDERATED CACHE FOR rudb.rschema.t1
```

- *Ejemplo 2:* Vaciar la información de la antememoria federada para todos los objetos del nombre-esquema-remoto *rschema* en la fuente de datos *rudb*.

```
FLUSH FEDERATED CACHE FOR rudb.rschema.*
```

- *Ejemplo 3:* Vaciar la información de la antememoria federada para todos los objetos de la fuente de datos *rudb*.

```
FLUSH FEDERATED CACHE FOR rudb.*.*
```

A continuación se muestra una alternativa a esta sintaxis:

```
FLUSH FEDERATED CACHE FOR SERVER rudb
```

### FLUSH OPTIMIZATION PROFILE CACHE

Se pueden compilar diversas sentencias utilizando el mismo perfil de optimización. Para conseguir que el proceso de perfiles de optimización sea más eficaz, el perfil de optimización se procesa la primera vez que se utiliza para optimizar una sentencia y la salida se almacena en la antememoria de perfiles de optimización. Las posteriores referencias al perfil de optimización utilizan la versión procesada de la antememoria de perfiles de optimización.

Un perfil de optimización se debe eliminar de la antememoria de perfiles de optimización cuando la versión almacenada en SYSTOOLS.OPT\_PROFILE se haya actualizado. Cuando la versión antigua se elimine de la antememoria, al optimizar las posteriores sentencias que utilicen el perfil de optimización se utilizará la versión nueva.

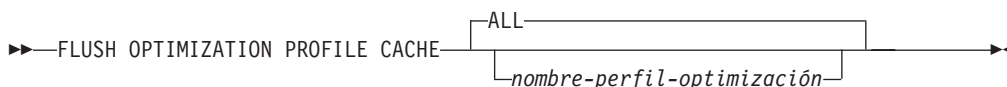
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. 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 incluir la autorización SQLADM o la DBADM (SQLSTATE 42502).

#### Sintaxis



#### Descripción

##### *nombre-perfil-optimización*

Especifica el nombre del perfil de optimización que se debe desechar de la antememoria de perfiles de optimización. Si el nombre especificado no está calificado, se utilizará el valor del registro CURRENT DEFAULT SCHEMA como calificador implícito.

##### **ALL**

Especifica que todos los perfiles de todas las particiones de base de datos activas se deben desechar de la antememoria de perfiles de optimización.

#### Notas

- La sentencia FLUSH OPTIMIZATION PROFILE CACHE elimina todos o un solo perfil de optimización de la antememoria de perfiles de optimización. También causa la invalidación lógica de cualquier sentencia de SQL dinámico en antememoria que se haya preparado con el perfil de optimización en cuestión.
- Cuando se realiza la siguiente petición para la misma sentencia de SQL, se vuelven a generar planes de acceso nuevos para los planes dinámicos invalidados.
- Los paquetes que hacen referencia a un perfil de optimización eliminado de la antememoria de perfiles de optimización por esta sentencia se deben volver a enlazar explícitamente para permitir que se generen nuevos planes de acceso.

### Ejemplos

- *Ejemplo 1:* El perfil de optimización "Rick"."Foo" se desecha de la antememoria de perfiles de optimización.

```
SET CURRENT SCHEMA = 'Rick'
FLUSH OPTIMIZATION PROFILE CACHE "Foo"
```

- *Ejemplo 2:* El perfil de optimización JOHN.ALL se elimina de la antememoria de perfiles de optimización.

```
SET CURRENT SCHEMA = 'Rick'
FLUSH OPTIMIZATION PROFILE CACHE JOHN.ALL
```

### Mensajes

- No se emitirá ningún error si la antememoria de perfiles de optimización está vacía o si los perfiles de optimización especificados (ya sea explícita o implícitamente) no existen en la antememoria de perfiles de optimización.

---

# FLUSH PACKAGE CACHE

La sentencia FLUSH PACKAGE CACHE elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes. Esta sentencia da lugar a la invalidación lógica de cualquier sentencia de SQL dinámico colocada en antememoria y hace que DB2 compile implícitamente la siguiente petición para la misma sentencia de SQL.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. 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 incluir la autorización SQLADM o DBADM.

### Sintaxis

►►—FLUSH PACKAGE CACHE—DYNAMIC—◄◄

### Notas

- Esta sentencia afecta a todas las entradas de SQL dinámico colocadas en antememoria que están en la antememoria de paquetes de todas las particiones de base de datos activas.
- Puesto que las sentencias de SQL dinámico colocadas en antememoria se invalidan, la antememoria de paquetes que se utiliza para la entrada colocada en antememoria se liberará si la entrada no está utilizándose cuando se ejecuta la sentencia FLUSH PACKAGE CACHE.
- Cualquier sentencia de SQL dinámico colocada en antememoria que actualmente esté utilizándose podrá seguir existiendo en la antememoria del paquete hasta que el usuario actual ya no la necesite; el nuevo usuario siguiente de la misma sentencia hará que DB2 realice una preparación implícita de la sentencia y el nuevo usuario ejecutará la nueva versión de la sentencia de SQL dinámico colocada en antememoria.



## FOR

La sentencia FOR ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.

### Invocación

Esta sentencia se puede incluir en:

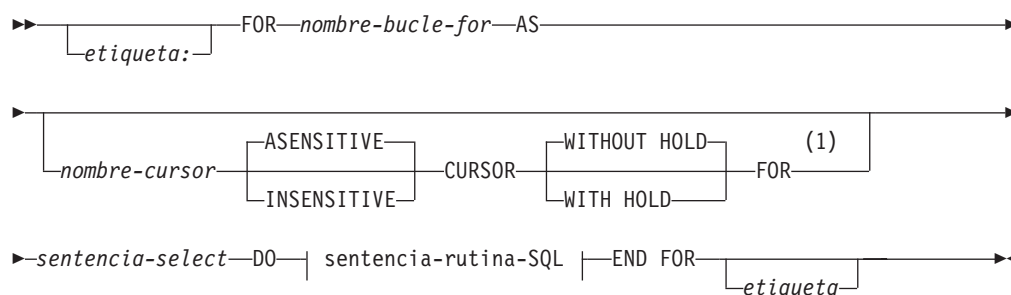
- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

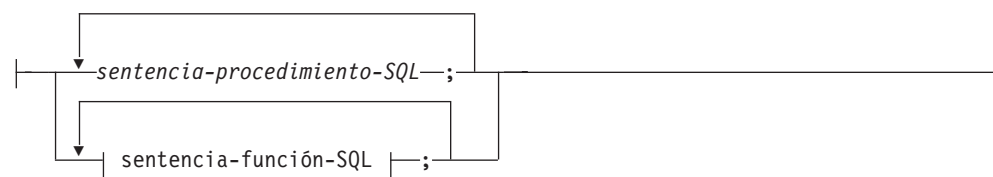
### Autorización

No se requieren privilegios para invocar la sentencia FOR. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia FOR. Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

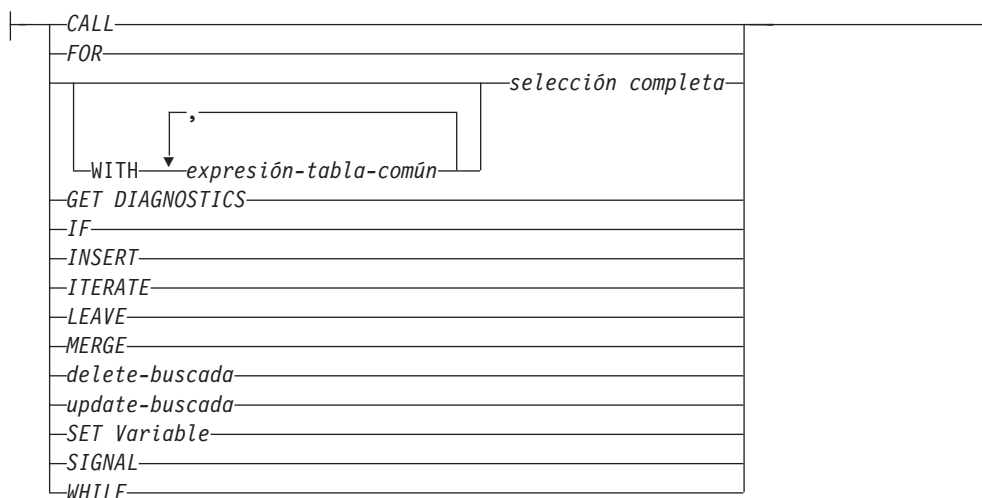
### Sintaxis



#### sentencia-rutina-SQL:



#### sentencia-función-SQL:



### Notas:

- 1 Esta opción sólo puede utilizarse en el contexto de un procedimiento de SQL o una sentencia de SQL compuesto (compilado).

## 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 una 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 normas 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 de resultados de la sentencia SELECT. Si no se especifica, DB2 genera un nombre de cursor exclusivo. Para obtener una descripción de ASENSITIVE, INSENSITIVE, WITHOUT HOLD o WITH HOLD, consulte "DECLARE CURSOR".

### *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 activador, una función, un método o una sentencia de SQL compuesto (en línea), la *sentencia-select* sólo debe constar de una *selección completa* con expresiones de tabla comunes opcionales.

### *sentencia-procedimiento-SQL*

Especifica una o varias sentencias que se deben invocar para cada fila de la tabla. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL o dentro de una sentencia de SQL compuesto (compilado). Consulte *sentencia-procedimiento-SQL* en la sentencia de "SQL compuesto (compilado)".

### *sentencia-función-SQL*

Especifica una o varias sentencias que se deben invocar para cada fila de la

tabla. No se da soporte a una operación de actualización-búsqueda, supresión-búsqueda ni INSERT en apodos. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL.

## Normas

- La lista de selección debe constar de nombres de columna exclusivos y los objetos especificados en la *sentencia-select* deben existir cuando se crea el procedimiento, o deben ser un objeto creado 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.

## Ejemplo

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 de 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 ATOMIC
 DECLARE fullname CHAR(40);
 FOR v1 AS
 SELECT firstnme, midinit, lastname FROM employee
 DO
 SET fullname = lastname CONCAT ', '
 CONCAT firstnme CONCAT ' ' CONCAT midinit;
 INSERT INTO tnames VALUES (fullname);
 END FOR;
END
```

## FREE LOCATOR

La sentencia FREE LOCATOR elimina la asociación entre una variable localizadora y su valor.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que 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 normas para la declaración de variables localizadoras.

La variable-localizadora debe tener actualmente un localizador asignado a ella. Es decir, deberá haberse asignado un localizador durante esta unidad de trabajo (por medio de una sentencia CALL, FETCH, SELECT INTO o VALUES INTO) y no deberá haberse liberado posteriormente (por medio de una sentencia FREE LOCATOR); de lo contrario, se devolverá 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.
```

## GET DIAGNOSTICS

La sentencia GET DIAGNOSTICS se utiliza para obtener información acerca del entorno de ejecución actual, incluida la información relacionada con la sentencia de SQL anterior (distinta de una sentencia GET DIAGNOSTICS) que se ha ejecutado. Parte de la información que está disponible por medio de la sentencia GET DIAGNOSTICS también está disponible en el SQLCA.

### Invocación

Esta sentencia se puede incluir en:

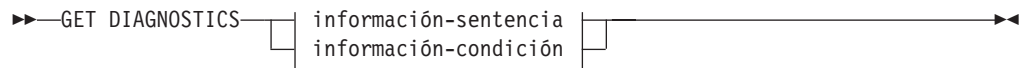
- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se necesita.

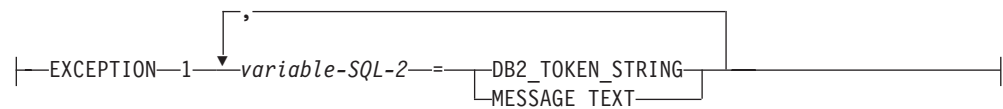
### Sintaxis



#### información-sentencia:



#### información-condición:



### Descripción

#### información-sentencia

Devuelve información sobre la última sentencia de SQL ejecutada.

##### *variable-SQL-1*

Identifica la variable que es el sujeto de la asignación. La variable no debe ser una variable global. Las variables de SQL se pueden definir en una sentencia compuesta. El tipo de datos de la variable debe ser compatible con el tipo de datos especificado en la Tabla 33 en la página 1113.

##### **DB2\_RETURN\_STATUS**

Identifica el valor de estado devuelto por el procedimiento asociado a la sentencia de SQL ejecutada anteriormente, siempre que la sentencia fuera

## GET DIAGNOSTICS

una sentencia CALL que invoca un procedimiento que devuelve un estado. Si la sentencia anterior no es una sentencia de este tipo, el valor que se devuelve carece de significado y puede ser cualquier entero.

### DB2\_SQL\_NESTING\_LEVEL

Identifica el nivel actual de anidamiento o recurrencia en vigor cuando se ejecutó la sentencia GET DIAGNOSTICS. Cada nivel de anidamiento corresponde a una invocación anidada o recursiva de una función de SQL compilado, un procedimiento de SQL compilado, un activador compilado o una sentencia de SQL compuesto (compilado) preparada dinámicamente. Si la sentencia GET DIAGNOSTICS se ejecuta fuera de un nivel de anidamiento, se devuelve el valor cero. Esta opción solamente puede especificarse en el contexto de una función de SQL compilado, un procedimiento de SQL compilado, un activador compilado o una sentencia de SQL compuesto (compilado) (SQLSTATE 42601).

### ROW\_COUNT

Identifica el número de filas que se asocian a la sentencia de SQL anterior. Si la sentencia de SQL anterior es una sentencia DELETE, INSERT o UPDATE, ROW\_COUNT identifica el número de filas que se han calificado para la operación. Si la sentencia anterior es una sentencia PREPARE, ROW\_COUNT identifica el número *estimado* de filas de resultados de la sentencia preparada.

### información-condición

Especifica que va a devolverse la información de error o de aviso para la sentencia de SQL que se ha ejecutado anteriormente. Si se necesita información acerca de un error, la sentencia GET DIAGNOSTICS debe ser la primera sentencia que debe especificarse en el manejador que se encargará de manejar el error. Si se necesita información acerca de un aviso y si el manejador va a recibir el control de la condición de aviso, la sentencia GET DIAGNOSTICS debe ser la primera sentencia que debe especificarse en ese manejador. Si el manejador *no* va a recibir el control de la condición de aviso, la sentencia GET DIAGNOSTICS debe ser la siguiente sentencia que debe ejecutarse. Esta opción sólo puede especificarse en el contexto de un Procedimiento de SQL (SQLSTATE 42601).

### variable-SQL-2

Identifica la variable que es el sujeto de la asignación. La variable no debe ser una variable global. Las variables de SQL se pueden definir en una sentencia compuesta. El tipo de datos de la variable debe ser compatible con el tipo de datos especificado en la Tabla 33 en la página 1113.

### DB2\_TOKEN\_STRING

Identifica cualquier símbolo de mensaje de error o de aviso que se devuelve de la sentencia de SQL ejecutada anteriormente. Si la sentencia se ha completado con un SQLCODE que es cero o si el SQLCODE no tiene ningún símbolo, para una variable VARCHAR se devuelve una serie de caracteres vacía o, en el caso de una variable CHAR, se devuelven blancos.

### MESSAGE\_TEXT

Identifica cualquier texto de mensaje de error o de aviso que se devuelve de la sentencia de SQL ejecutada anteriormente. El texto del mensaje se devuelve en el idioma del servidor de bases de datos en el que se ha procesado la sentencia. Si la sentencia se ha completado con un SQLCODE que es cero, para una variable VARCHAR se devuelve una serie de caracteres vacía o, en el caso de una variable CHAR, se devuelven blancos.

## Notas

- La sentencia GET DIAGNOSTICS no cambia el contenido del área de diagnósticos (SQLCA). Las variables especiales SQLSTATE o SQLCODE declaradas en un procedimiento de SQL se establecen en el SQLSTATE o SQLCODE devuelto por la ejecución de la sentencia GET DIAGNOSTICS.
- **Tipos de datos para elementos:** La tabla siguiente muestra los tipos de datos SQL para cada elemento de diagnóstico. Cuando un elemento de diagnóstico se asigna a una variable, el tipo de datos de la variable debe ser compatible con el tipo de datos del elemento de diagnóstico solicitado.

Tabla 33. Tipos de datos para los elementos GET DIAGNOSTICS

| Tipo de información      | Elemento              | Tipo de datos  |
|--------------------------|-----------------------|----------------|
| Información de sentencia | DB2_RETURN_STATUS     | INTEGER        |
| Información de sentencia | DB2_SQL_NESTING_LEVEL | INTEGER        |
| Información de sentencia | ROW_COUNT             | DECIMAL(31,0)  |
| Información de condición | DB2_TOKEN_STRING      | VARCHAR(1000)  |
| Información de condición | MESSAGE_TEXT          | VARCHAR(32672) |

- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
  - RETURN\_STATUS puede especificarse en lugar de DB2\_RETURN\_STATUS.

## Ejemplos

- *Ejemplo 1:* en un procedimiento de SQL, ejecute una sentencia GET DIAGNOSTICS para determinar cuántas filas se actualizaron.

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
LANGUAGE SQL
BEGIN
 DECLARE SQLSTATE CHAR(5);
 DECLARE rcount INTEGER;
 UPDATE CORPDATA.PROJECT
 SET PRSTAFF = PRSTAFF + 1.5
 WHERE DEPTNO = deptnbr;
 GET DIAGNOSTICS rcount = ROW_COUNT;
 -- En este momento, rcount contiene el número de filas que se actualizaron.
 ...
END
```

- *Ejemplo 2:* dentro de un procedimiento de SQL, procese el valor de estado devuelto por la invocación de un procedimiento denominado TRYIT, el cual puede devolver explícitamente un valor positivo, lo que indica un error de usuario, o encontrar errores de 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 = DB2_RETURN_STATUS;
 IF RETVAL <> 0 THEN
 ...
 LEAVE A1;
```

## GET DIAGNOSTICS

```
ELSE
 ...
END IF;
END A1
```



## GOTO

La sentencia GOTO se utiliza para definir una bifurcación a una etiqueta definida por el usuario dentro de un procedimiento de SQL.

### Invocación

Esta sentencia sólo puede incorporarse en un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se necesita.

### 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 manejador de condiciones, la *etiqueta* se debe definir en el mismo manejador de condiciones, de acuerdo con las demás normas sobre el ámbito
- Si la sentencia GOTO se define fuera de un manejador de condiciones, la *etiqueta* no se debe definir dentro de un manejador.

Si la *etiqueta* no está definida dentro de un ámbito que sea accesible para la sentencia GOTO, se produce un error (SQLSTATE 42736).

### Notas

- 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.

### Ejemplo

En la sentencia compuesta siguiente, los parámetros *rating* y *v\_empno* se pasan al procedimiento que, a continuación, devuelve el parámetro de salida *return\_parm* como una duración en forma de fecha. Si el tiempo que el empleado lleva prestando servicios a la empresa es inferior a seis meses, la sentencia GOTO transfiere el control al final del procedimiento y *new\_salary* no cambia.

```
CREATE PROCEDURE adjust_salary
 (IN v_empno CHAR(6),
 IN rating INTEGER,
```

## GOTO

```
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);
 ELSEIF 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
```

## GRANT (autorizaciones de bases 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

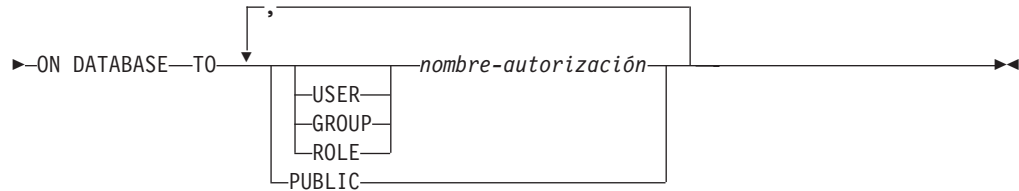
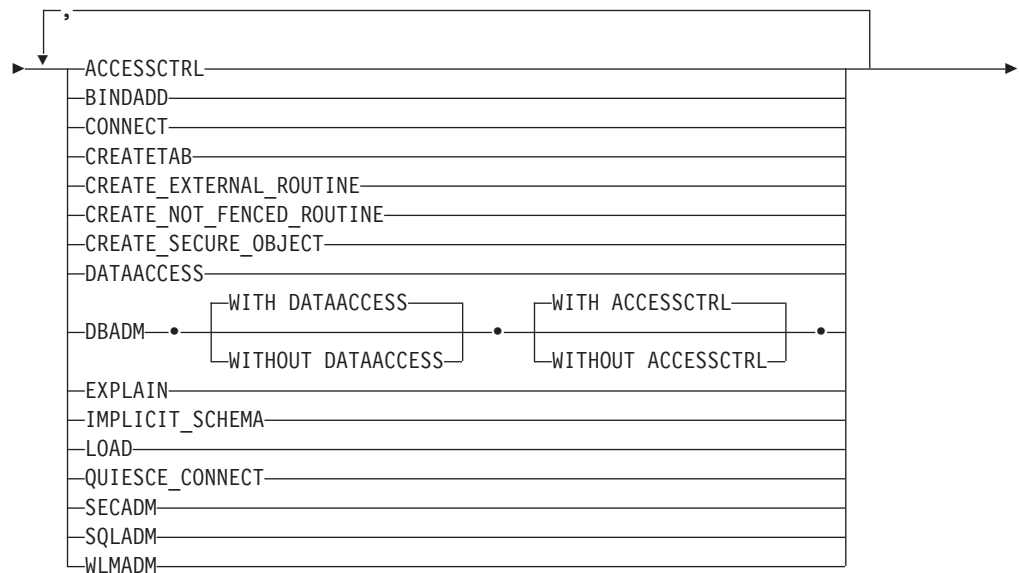
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Para otorgar autorización ACCESSCTRL, CREATE\_SECURE\_OBJECT, DATAACCESS, DBADM o SECADM, se necesita autorización SECADM. Para otorgar otras autorizaciones, se necesita la autorización ACCESSCTRL o SECADM.

### Sintaxis

► GRANT



## GRANT (autorizaciones de bases de datos)

### Descripción

#### ACCESSCTRL

Otorga la autorización de control de acceso. La autorización ACCESSCTRL permite a quien la posee lo siguiente:

- Otorgar y revocar las autorizaciones de bases de datos siguientes: BINDADD, CONNECT, CREATETAB, CREATE\_EXTERNAL\_ROUTINE, CREATE\_NOT\_FENCED\_ROUTINE, EXPLAIN, IMPLICIT\_SCHEMA, LOAD, QUIESE\_CONNECT, SQLADM, WLMADM
- Otorgar y revocar todos los privilegios de nivel de objeto

La autorización ACCESSCTRL no puede otorgarse a PUBLIC (SQLSTATE 42508).

#### 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 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\_EXTERNAL\_ROUTINE

Otorga la autorización para registrar rutinas externas. Deberá tenerse cuidado de que las rutinas que se registran de esta forma no generen efectos secundarios negativos. (Para obtener más información, consulte la descripción de la cláusula THREADSAFE en las sentencias de rutina CREATE o ALTER.)

Cuando una rutina externa se ha registrado, esta sigue existiendo, aunque posteriormente se invoque CREATE\_EXTERNAL\_ROUTINE.

#### CREATE\_NOT\_FENCED\_ROUTINE

Otorga la autorización para registrar rutinas que se ejecutan en el proceso del gestor de bases de datos. Deberá tenerse cuidado de que las rutinas que se registran de esta forma no generen efectos secundarios negativos. (Para obtener más información, vea la descripción de la cláusula FENCED en las sentencias de rutina CREATE o ALTER).

Cuando una rutina se ha registrado como no delimitada, sigue ejecutándose de esta forma, aunque posteriormente se invoque CREATE\_NOT\_FENCED\_ROUTINE.

CREATE\_EXTERNAL\_ROUTINE se otorga automáticamente a un *nombre-autorización* al que se haya otorgado la autorización CREATE\_NOT\_FENCED\_ROUTINE.

#### CREATE\_SECURE\_OBJECT

Otorga autorización para crear activadores seguros y funciones seguras. También otorga autorización para alterar el atributo seguro de esos objetos.

## GRANT (autorizaciones de bases de datos)

### DATAACCESS

Otorga la autorización para acceder a los datos. La autorización DATAACCESS permite a quien la posee lo siguiente:

- Seleccionar, insertar, actualizar, suprimir y cargar datos
- Ejecutar cualquier paquete
- Ejecutar cualquier rutina (excepto rutinas de auditoría)

La autorización DATAACCESS no puede otorgarse a PUBLIC (SQLSTATE 42508).

### DBADM

Otorga la autorización de administrador de bases de datos. Un administrador de bases de datos posee prácticamente todos los privilegios sobre casi todos los objetos de la base de datos. Las únicas excepciones son aquellos privilegios que forman parte de las autorizaciones de control de acceso, acceso a los datos y administrador de seguridad. DBADM no se puede otorgar a PUBLIC.

### EXPLAIN

Otorga autorización para explicar sentencias. La autorización EXPLAIN permite a quien la posee explicar, preparar y describir sentencias de SQL dinámico y estáticas sin necesidad de acceder a los datos.

### IMPLICIT\_SCHEMA

Otorga la autorización para crear implícitamente un esquema.

### 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. DATAACCESS y DBADM también tienen esta autorización por omisión. Sin embargo, si un usuario sólo tiene autorización LOAD (no DATAACCESS), también se requerirá que el usuario tenga privilegios de nivel de tabla. 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

### QUIESCE\_CONNECT

Otorga la autorización para acceder a la base de datos mientras está inmovilizada.

### SECADM

Otorga la autorización de administrador de seguridad. La autorización permite realizar las operaciones siguientes a quien la posee:

- Crear y descartar objetos de seguridad como, por ejemplo, políticas de auditoría, roles, etiquetas de seguridad, componentes de etiqueta de seguridad, políticas de seguridad y contextos fiables
- Otorgar y revocar autorizaciones, exenciones, privilegios, roles y etiquetas de seguridad.
- Otorgar y revocar el privilegio SETSESSIONUSER
- Ejecutar TRANSFER OWNERSHIP en objetos propiedad de otros

## GRANT (autorizaciones de bases de datos)

La autorización SECADM no puede otorgarse a PUBLIC (SQLSTATE 42508).

### SQLADM

Otorga autorización para gestionar la ejecución de sentencias de SQL. La autorización SQLADM permite a quien la posee lo siguiente:

- Crear, descartar, vaciar y definir supervisores de sucesos
- Explicar, preparar y describir sentencias de SQL dinámico y estáticas sin necesidad de acceder a los datos
- Vaciar la antememoria de perfil de optimización
- Vaciar la antememoria de paquetes
- Ejecutar el programa de utilidad runstats
- Crear, alterar, descartar y definir listas de uso

### WLMADM

Otorga autorización para gestionar cargas de trabajo. La autorización WLMADM permite a quien la posee lo siguiente:

- Crear, descartar y alterar las clases de servicio, conjuntos de acciones de trabajo, conjuntos de clases de trabajo o cargas de trabajo.

**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.

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

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 un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.

## GRANT (autorizaciones de bases de datos)

- Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.

### Notas

- Las autorizaciones ACCESSCTRL, CREATE\_SECURE\_OBJECT, DATAACCESS, DBADM o SECADM no pueden otorgarse al grupo especial PUBLIC. Por lo tanto, otorgar las autorizaciones ACCESSCTRL, CREATE\_SECURE\_OBJECT, DBADM, DATAACCESS o SECADM a un rol *nombre-rol* generará errores si *nombre-rol* se ha otorgado a PUBLIC directa o indirectamente (SQLSTATE 42508).
  - El rol *nombre-rol* se otorga directamente a PUBLIC si se ha emitido la siguiente sentencia:  
**GRANT ROLE *nombre-rol* TO PUBLIC**
  - El rol *nombre-rol* se otorga indirectamente a PUBLIC si se han emitido las siguientes sentencias:  
**GRANT ROLE *nombre-rol* TO ROLE *nombre2-rol***  
**GRANT ROLE *nombre2-rol* TO PUBLIC**
- **Alternativas de sintaxis:** Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos.
  - CREATE\_NOT\_FENCED puede especificarse en lugar de CREATE\_NOT\_FENCED\_ROUTINE
  - Puede especificarse SYSTEM en lugar de DATABASE
- **Privilegios que se otorgan a un grupo:** un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### 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 la autorización BINDADD en la base de datos a un grupo denominado 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).

- *Ejemplo 3:* Proporcione la autorización de administrador de seguridad al usuario Walid.

```
GRANT SECADM ON DATABASE TO USER Walid
```

- *Ejemplo 4:* Un usuario con autorización SECADM otorga la autorización CREATE\_SECURE\_OBJECT al usuario Haytham.

## GRANT (autorizaciones de bases de datos)

```
GRANT CREATE_SECURE_OBJECT ON DATABASE TO USER HAYTHAM
```



## GRANT (exención)

Este formato de la sentencia GRANT otorga a un usuario, grupo o rol una exención para una norma de acceso para una política de seguridad de control de acceso basado en etiquetas (LBAC) especificada.

Cuando el usuario que mantiene la exención accede a los datos de una tabla protegida por esa política de seguridad, la norma indicada no se impondrá al decidir si se puede acceder a los datos.

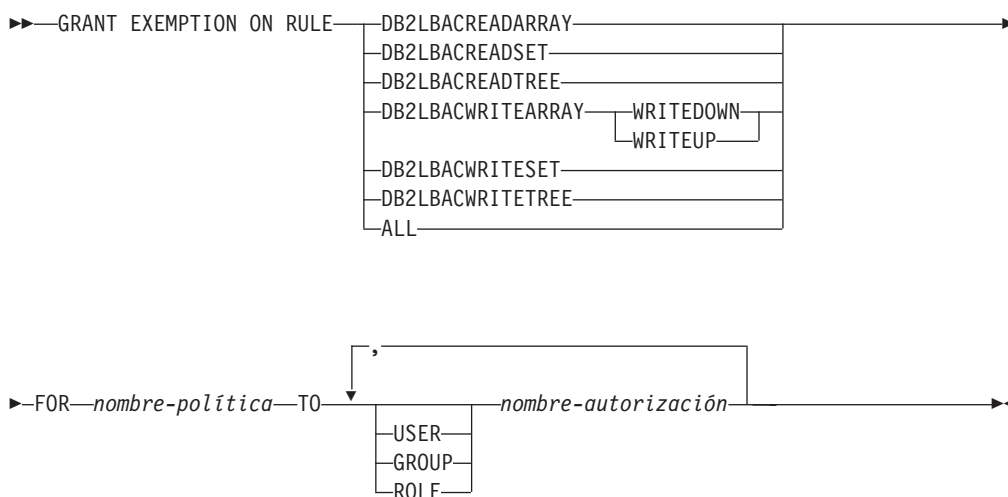
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

### Sintaxis



### Descripción

#### EXEMPTION ON RULE

Otorga una exención en una norma de acceso.

#### DB2LBACREADARRAY

Otorga una en la norma DB2LBACREADARRAY predefinida.

#### DB2LBACREADSET

Otorga una exención en la norma DB2LBACREADSET predefinida.

#### DB2LBACREADTREE

Otorga una en la norma DB2LBACREADTREE predefinida.

#### DB2LBACWRITEARRAY

Otorga una exención en la norma DB2LBACWRITEARRAY predefinida.

## GRANT (exención)

### WRITEDOWN

Especifica que la exención sólo se aplica a la grabación hacia abajo.

### WRITEUP

Especifica que la exención sólo se aplica a la grabación hacia arriba.

### DB2LBACWRITASET

Otorga una exención en la norma DB2LBACWRITASET predefinida.

### DB2LBACWRITETREE

Otorga una exención en la norma DB2LBACWRITETREE predefinida.

### ALL

Otorga una exención en todas las normas predefinidas.

### FOR *nombre-política*

Identifica la política de seguridad para la que se otorga la exención. La exención sólo será efectiva para las tablas que estén protegidas por esta política de seguridad. El nombre debe identificar a una política de seguridad ya descrita en el catálogo (SQLSTATE 42704).

**TO** Especifica a quién se le otorga la exención.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- Si la política de seguridad no está definida para que tenga en cuenta el acceso a través de grupos o roles, se ignorará cualquier exención otorgada a un grupo o rol cuando se intente el acceso.

## Notas

- Por omisión, cuando se crea una política de seguridad, sólo se tienen en cuenta las exenciones otorgadas a un usuario individual. Para que se tengan en cuenta

los grupos o los roles para la política de seguridad, deberá emitir la sentencia ALTER SECURITY POLICY y especificar USE GROUP AUTHORIZATION o USE ROLE AUTHORIZATION que sea de aplicación.

### Ejemplos

- *Ejemplo 1:* Otorgar una exención en la norma de acceso DB2LBACREADSET para la política de seguridad DATA\_ACCESS del usuario WALID.

```
GRANT EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS TO USER WALID
```

- *Ejemplo 2:* Otorga una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEDOWN para la política de seguridad DATA\_ACCESS al usuario BOBBY.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
FOR DATA_ACCESS TO USER BOBBY
```

- *Ejemplo 3:* Otorga una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEUP para la política de seguridad DATA\_ACCESS al usuario BOBBY.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
FOR DATA_ACCESS TO USER BOBBY
```

### GRANT (privilegios de variable global)

Este formato de la sentencia GRANT otorga uno o más privilegios de una variable global creada.

#### Invocación

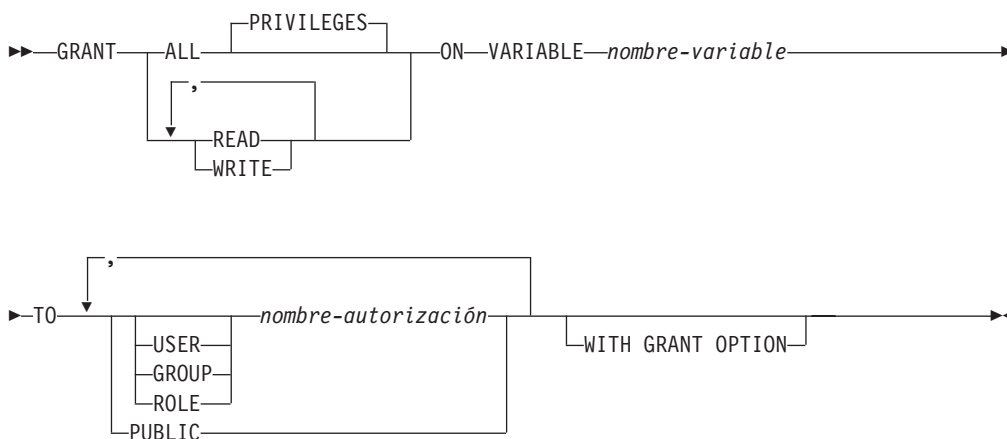
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

#### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH GRANT OPTION para cada privilegio identificado en la variable global
- Autorización ACCESSCTRL o SECADM

#### Sintaxis



#### Descripción

##### ALL PRIVILEGES

Otorga todos los privilegios sobre la variable global especificada.

##### READ

Otorga el privilegio para leer el valor de la variable global especificada.

##### WRITE

Otorga el privilegio para asignar un valor a la variable global especificada.

##### ON VARIABLE *nombre-variable*

Identifica la variable global sobre la que va a otorgarse uno o más privilegios. El *nombre-variable*, incluyendo un calificador implícito o explícito, debe identificar una variable global que existe en el servidor actual y que no es una variable de módulo (SQLSTATE 42704).

**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 a un grupo.

### ROLE

Especifica que el *nombre-autorización* identifique un rol existente en el servidor actual (SQLSTATE 42704).

### *nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles. 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 especificados a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte "Autorizaciones, privilegios y propiedad de objetos".

### WITH GRANT OPTION

Permite que el *nombre-autorización* especificado pueda otorgar los privilegios a otros usuarios. Si se omite la cláusula WITH GRANT OPTION, los *nombre-autorización* especificados no pueden otorgar los privilegios a otros usuarios a menos que dicha autorización se haya recibido de alguna otra fuente.

## Normas

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se asume GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.

## Notas

- **Privilegios que se otorgan a un grupo:** un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

## Ejemplo

Otorgue el privilegio READ y WRITE en la variable global MYSCHEMA.MYJOB\_PRINTER para el usuario ZUBIRI.

## GRANT (privilegios de variable global)

```
GRANT READ, WRITE ON VARIABLE
MYSHEMA.MYJOB_PRINTER TO ZUBIRI
```

## GRANT (privilegios de índice)

Esta forma de la sentencia GRANT otorga el privilegio CONTROL en índices.

### Invocación

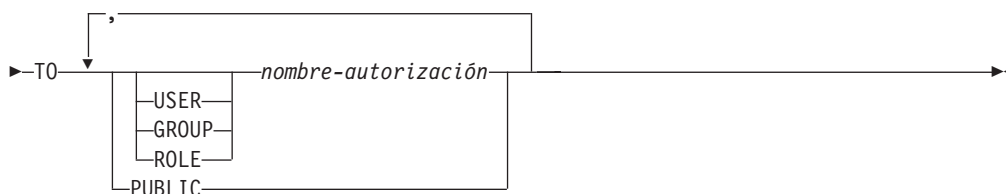
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis

► GRANT CONTROL ON INDEX *nombre-índice* ►



### Descripción

#### CONTROL

Otorga el privilegio para descartar 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 debe 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

#### *nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

## GRANT (privilegios de índice)

### **PUBLIC**

Otorga los privilegios a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

### **Normas**

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.

### **Notas**

- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### **Ejemplo**

Otorgar privilegio CONTROL en el índice DEPTIDX al usuario cuyo ID es KIESLER:

```
GRANT CONTROL ON INDEX DEPTIDX TO USER KIESLER
```



## GRANT (privilegios de módulo)

Esta forma de la sentencia GRANT otorga privilegios sobre un módulo.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

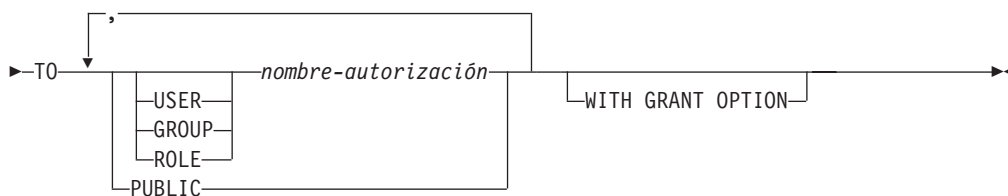
### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH GRANT OPTION para EXECUTE en el módulo.
- Autorización ACCESSCTRL o SECADM.

### Sintaxis

►► GRANT EXECUTE ON MODULE *nombre-módulo* ►►



### Descripción

#### EXECUTE

Otorga el privilegio para hacer referencia a objetos de módulo no publicados. Esto incluye el privilegio para:

- Ejecutar cualquier rutina publicada definida en el módulo.
- Leer y grabar en cualquier variable global publicada definida en el módulo.
- Hacer referencia a cualquier tipo definido por el usuario publicado que esté definido en el módulo.
- Hacer referencia a cualquier condición publicada definida en el módulo.

#### ON MODULE *nombre-módulo*

Identifica el módulo sobre el que se ha otorgado el privilegio. El nombre-módulo debe identificar un módulo que exista en el servidor actual (SQLSTATE 42704).

**TO** Indica 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.

## GRANT (privilegios de módulo)

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista uno o varios ID de autorización.

### PUBLIC

Otorga el privilegio a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

### WITH GRANT OPTION

Permite que los *nombres-autorización* especificados puedan otorgar el privilegio EXECUTE a otros usuarios. Si se omite la cláusula WITH GRANT OPTION, los *nombre-autorización* especificados no pueden otorgar los privilegios EXECUTE a otros usuarios a menos que dicha autorización se haya recibido de alguna otra fuente.

## Notas

- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

## Ejemplo

Otorgar el privilegio EXECUTE para el módulo MYMODA al usuario JONES:

```
GRANT EXECUTE
ON MODULE MYMODA
TO JONES
```

# GRANT (privilegios de paquete)

Esta forma de la sentencia GRANT otorga los privilegios en un paquete.

## Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

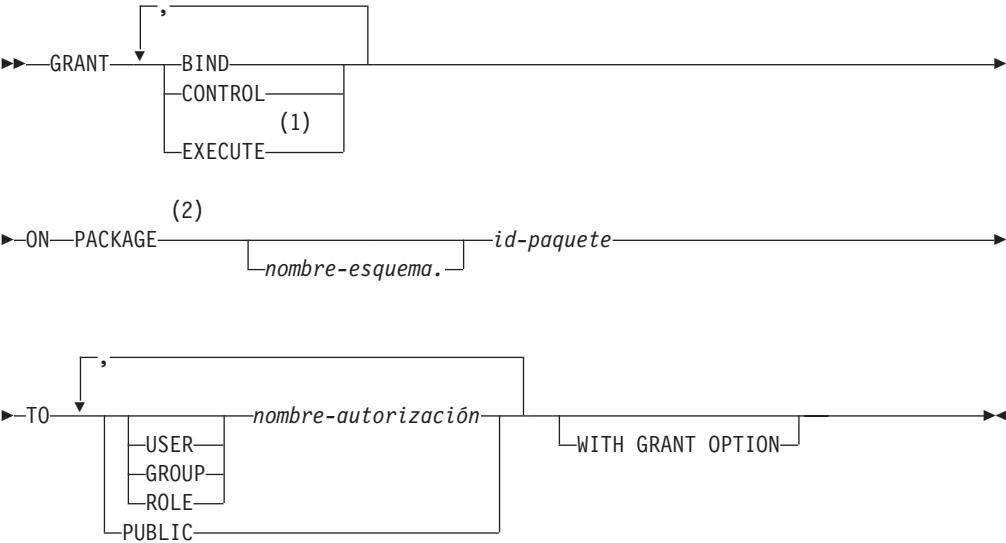
## Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CONTROL para el paquete referenciado
- WITH GRANT OPTION para cada privilegio identificado en *nombre-paquete*
- Autorización ACCESSCTRL o SECADM

La autorización ACCESSCTRL o SECADM es necesaria para otorgar el privilegio CONTROL.

## 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. El privilegio BIND permite a un usuario volver a emitir el mandato BIND para ese paquete o bien emitir el mandato REBIND. También permite a un usuario crear una nueva versión de un paquete existente.

## GRANT (privilegios de paquete)

Además del privilegio BIND, un usuario debe disponer de los privilegios necesarios para cada tabla a la que hagan referencia las sentencias DML estáticas contenidas en un programa. Esto es necesario porque la autorización para las sentencias DML estáticas se comprueba durante el enlace.

### CONTROL

Otorga el privilegio para volver a enlazar, descartar 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.

CONTROL permite otorgar los privilegios mencionados anteriormente (excepto CONTROL) a otros usuarios.

### EXECUTE

Otorga el privilegio para ejecutar el paquete.

### ON PACKAGE *nombre-esquema.id-paquete*

Especifica el nombre del paquete en el que se deben otorgar los privilegios. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. La concesión de un privilegio de paquete se aplica a todas las versiones del paquete (es decir, a todos los paquetes que comparten el mismo ID de paquete y esquema de paquete).

### 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

#### *nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

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 un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

#### WITH GRANT OPTION

Permite que el *nombre-autorización* especificado pueda otorgar (GRANT) los privilegios a otros usuarios.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios que pueden aplicarse, excepto CONTROL (SQLSTATE 01516).

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:

- Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
- Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
- Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
- Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
- Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
- Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.

### Notas

- Los privilegios de paquete se aplican a todas las versiones de un paquete (es decir, a todos los paquetes que comparten el mismo ID de paquete y esquema de paquete). No es posible restringir el acceso sólo a una única versión. Puesto que el privilegio CONTROL se otorga implícitamente al usuario que enlaza el paquete, si dos usuarios distintos enlazan dos versiones de un paquete, a ambos usuarios se otorgará implícitamente acceso al paquete del otro.
- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### 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 bien

```
GRANT EXECUTE ON PACKAGE
CORPDATA.PKGA TO USER EMPLOYEE
```

# GRANT (rol)

Este formato de la sentencia GRANT otorga roles a usuarios, grupos u otros roles.

## Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

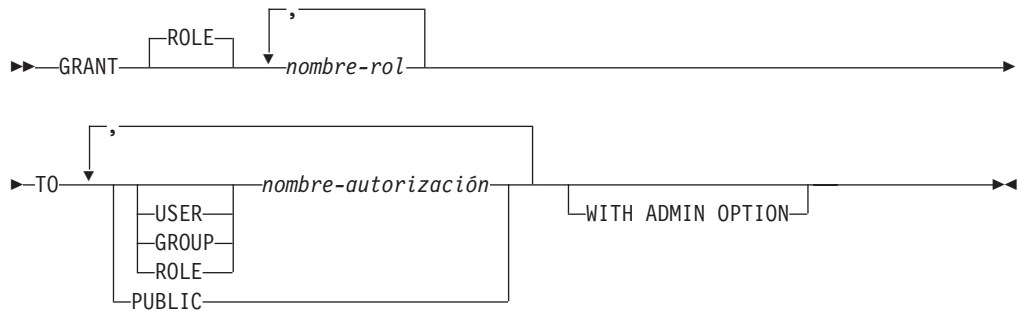
## Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH ADMIN OPTION en el rol
- Autorización SECADM

Es necesaria la autorización SECADM para otorgar el privilegio WITH ADMIN OPTION a un *nombre-autorización*.

## Sintaxis



## Descripción

**ROLE** *nombre-rol*, ...

Identifica uno o más roles a otorgar. Cada *nombre-rol* debe identificar un rol existente en el servidor actual (SQLSTATE 42704).

**TO** Especifica a quién se otorga el rol.

**USER**

Especifica que el *nombre-autorización* identifica a un usuario.

**GROUP**

Especifica que el *nombre-autorización* identifica a un grupo.

**ROLE**

Especifica que el *nombre-autorización* identifique un rol existente en el servidor actual (SQLSTATE 42704).

*nombre-autorización*, ...

Lista los ID de autorización de uno o más usuarios, grupos o roles. 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 roles especificados a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte “Autorizaciones, privilegios y propiedad de objetos”.

**WITH ADMIN OPTION**

Permite al *nombre-autorización* especificado revocar el *nombre-rol* a otros o asociar un comentario al rol. No permite que el *nombre-autorización* descarte el rol.

**Normas**

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.
- Las jerarquías de roles pueden crearse otorgando un rol a otro rol. Sin embargo, los ciclos no se permiten (SQLSTATE 428GF). Por ejemplo, si el rol R1 se otorga a otro rol R2, el rol R2 (u algún otro rol Rn que contenga R2) no podría volverse a otorgar a R1 ya que esto ocasionaría un ciclo.

**Notas**

- Cuando el rol R1 se otorga a otro rol R2, R2 contiene R1.
- La autorización DBADM no puede otorgarse a PUBLIC. Por tanto:
  - La acción de otorgar el rol R1 a PUBLIC falla (SQLSTATE 42508) si el rol R1 retiene la autorización DBADM directa o indirectamente.
    - El rol R1 retiene la autorización DBADM directamente si se ha emitido la siguiente sentencia:
 

```
GRANT DBADM ON DATABASE TO ROLE R1
```
    - El rol R1 retiene la autorización DBADM indirectamente si se han emitido las siguientes sentencias:
 

```
GRANT DBADM ON DATABASE TO ROLE R2
```

```
GRANT ROLE R2 TO ROLE R1
```
  - La acción de otorgar el rol R1, que retiene la autorización DBADM, al rol R2 falla (SQLSTATE 42508) si el rol R2 se otorga a PUBLIC directa o indirectamente.
    - El rol R2 se otorga a PUBLIC directamente si se ha emitido la siguiente sentencia:
 

```
GRANT ROLE R2 TO PUBLIC
```
    - El rol R2 se otorga a PUBLIC indirectamente si se han emitido las siguientes sentencias:

## GRANT (rol)

**GRANT ROLE R2 TO ROLE R3**

**GRANT ROLE R3 TO PUBLIC**

- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### Ejemplos

- *Ejemplo 1*: Otorgue el rol INTERN al rol DOCTOR y el rol DOCTOR al rol SPECIALIST.

**GRANT ROLE INTERN TO ROLE DOCTOR**

**GRANT ROLE DOCTOR TO ROLE SPECIALIST**

- *Ejemplo 2*: Otorgue el rol INTERN a PUBLIC.

**GRANT ROLE INTERN TO PUBLIC**

- *Ejemplo 3*: Otorgue el rol SPECIALIST al usuario BOB y al grupo TORONTO.

**GRANT ROLE SPECIALIST TO USER BOB, GROUP TORONTO**



## GRANT (privilegios de rutina)

Esta forma de la sentencia GRANT otorga privilegios para una rutina (función, método o procedimiento) que no está definida en un módulo.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

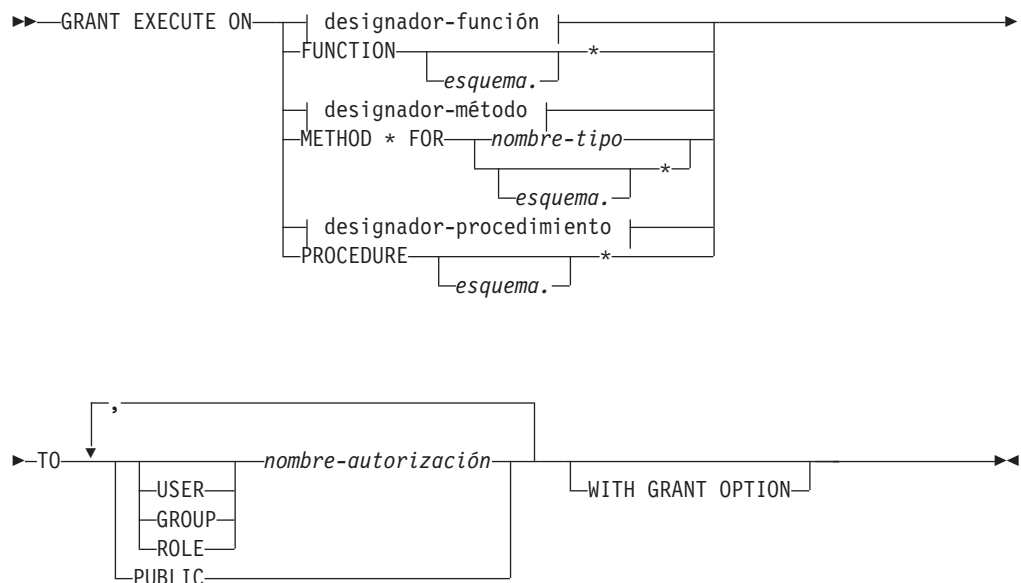
- WITH GRANT OPTION para EXECUTE en la rutina
- Autorización ACCESSCTRL o SECADM

Para otorgar todos los privilegios EXECUTE de rutina del esquema o tipo, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- WITH GRANT OPTION para EXECUTE en todas las rutinas existentes y futuras (del tipo especificado) dentro del esquema especificado
- Autorización ACCESSCTRL o SECADM

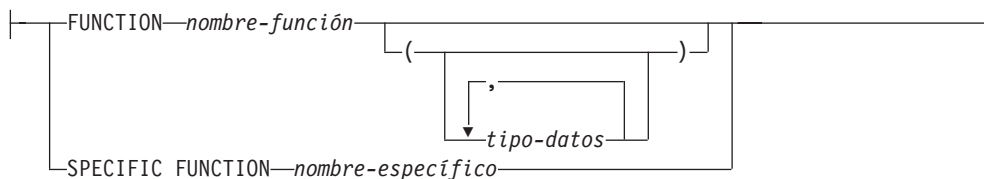
Para otorgar el privilegio EXECUTE sobre los procedimientos de auditoría y funciones de tabla, se requiere la autorización SECADM. El privilegio EXECUTE con WITH GRANT OPTION no se puede otorgar para estas rutinas (SQLSTATE 42501).

### Sintaxis

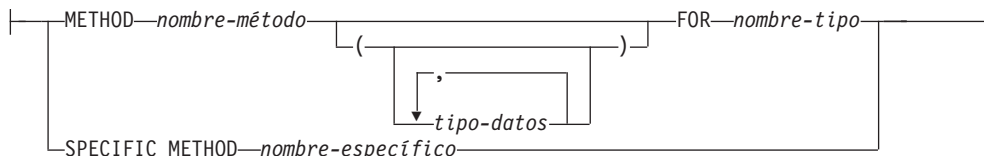


## GRANT (privilegios de rutina)

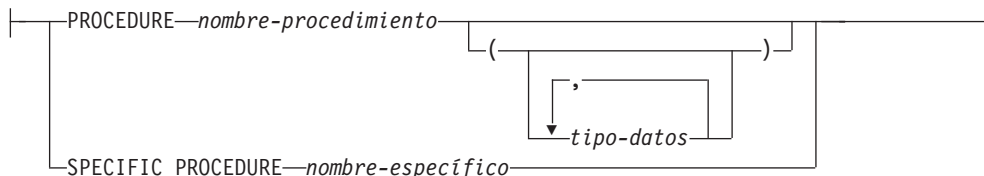
### designador-función:



### designador-método:



### designador-procedimiento:



## Descripción

### EXECUTE

Otorga el privilegio para ejecutar la función, el método o el procedimiento definido por el usuario que se identifica.

#### designador-función

Identifica de forma exclusiva la función para la que se concede el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

#### FUNCTION *esquema.\**

Identifica todas las funciones del esquema, incluida cualquier función que pueda crearse en el futuro. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

#### designador-método

Identifica de forma exclusiva el método para el que se concede el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

#### METHOD \*

Identifica todos los métodos del tipo *nombre-tipo*, incluido cualquier método que pueda crearse en el futuro.

#### FOR *nombre-tipo*

Especifica el nombre del tipo en el que se encuentra el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el valor

del registro especial CURRENT SCHEMA se utiliza como calificador de un nombre de tipo no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados. Para identificar todos los tipos del esquema, incluido cualquier tipo que pueda crearse en el futuro, puede utilizarse un asterisco (\*) en lugar del *nombre-tipo*.

### *designador-procedimiento*

Identifica de forma exclusiva el procedimiento para el que se concede el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

### **PROCEDURE** *esquema*.\*

Identifica todos los procedimientos del esquema, incluido cualquier procedimiento que pueda crearse en el futuro. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

**TO** Especifica a quién se otorga el privilegio EXECUTE.

### **USER**

Especifica que el *nombre-autorización* identifica a un usuario.

### **GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

### **ROLE**

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

### *nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

### **PUBLIC**

Otorga el privilegio EXECUTE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

### **WITH GRANT OPTION**

Permite que los *nombres-autorización* especificados puedan otorgar (GRANT) el privilegio EXECUTE a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar el privilegio EXECUTE a otros usuarios si éstos:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar el privilegio EXECUTE desde alguna otra fuente.

## **Normas**

- No es posible otorgar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).

## GRANT (privilegios de rutina)

- Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- 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 el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA y no se han otorgado privilegios, se devolverá un mensaje de aviso (SQLSTATE 01007). Si el usuario que otorga privilegios no dispone de ningún privilegio sobre el objeto de la operación de concesión de privilegios, se devolverá un error (SQLSTATE 42501).

### Notas

- Los privilegios para una rutina definida en un módulo se otorgan en el nivel de módulo mediante la sentencia GRANT (privilegios de módulo). El privilegio EXECUTE en el módulo permite el acceso a todos los objetos del módulo.
- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### Ejemplos

- *Ejemplo 1*: Otorgue el privilegio EXECUTE para la función CALC\_SALARY al usuario JONES. Se da por supuesto que, en el esquema, sólo existe una función con el nombre de función CALC\_SALARY.

```
GRANT EXECUTE ON FUNCTION CALC_SALARY TO JONES
```
- *Ejemplo 2*: Otorgue el privilegio EXECUTE para el procedimiento VACATION\_ACCR a todos los usuarios del servidor actual.

```
GRANT EXECUTE ON PROCEDURE VACATION_ACCR TO PUBLIC
```
- *Ejemplo 3*: Otorgue el privilegio EXECUTE para la función DEPT\_TOTALS al ayudante de administración y otorgue al ayudante la capacidad de otorgar el privilegio EXECUTE para esta función a otros usuarios. La función tiene el nombre específico DEPT85\_TOT. Se da por supuesto que el esquema tiene más de una función denominada DEPT\_TOTALS.

```
GRANT EXECUTE ON SPECIFIC FUNCTION DEPT85_TOT
TO ADMIN_A WITH GRANT OPTION
```
- *Ejemplo 4*: Otorgue el privilegio EXECUTE para la función NEW\_DEPT\_HIRES a HR (Recursos humanos). La función tiene dos parámetros de entrada de tipo

## GRANT (privilegios de rutina)

INTEGER y CHAR(10) respectivamente. Se da por supuesto que el esquema tiene más de una función denominada NEW\_DEPT\_HIRES.

```
GRANT EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10)) TO HR
```

- *Ejemplo 5:* Otorgue el privilegio EXECUTE para el método SET\_SALARY de tipo EMPLOYEE al usuario JONES.

```
GRANT EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE TO JONES
```

### GRANT (privilegios de esquema)

Esta forma de la sentencia GRANT otorga privilegios en un esquema.

#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

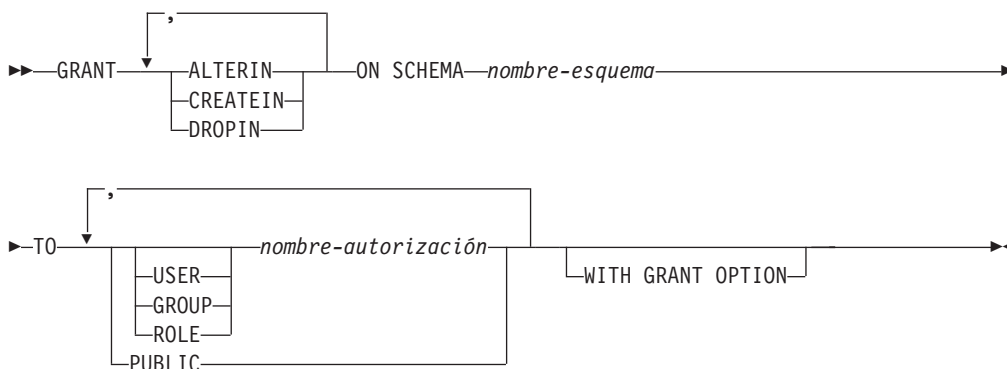
#### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH GRANT OPTION para cada privilegio identificado en *nombre-esquema*
- Autorización ACCESSCTRL o SECADM

Ningún usuario puede otorgar privilegios en los nombres de esquema que empiezan con SYSIBM, SYSCAT, SYSFUN o 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 explícitamente recibe automáticamente el privilegio CREATEIN. En un esquema creado implícitamente se otorga automáticamente el privilegio CREATEIN a PUBLIC.

##### DROPIN

Otorga el privilegio para descartar 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 deben 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

#### *nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

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 un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

### WITH GRANT OPTION

Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- 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). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

## GRANT (privilegios de esquema)

### Notas

- **Otorgación sobre SYSPUBLIC:** se pueden otorgar privilegios sobre el esquema reservado SYSPUBLIC. Si se otorga el privilegio CREATEIN, el usuario podrá crear un alias público, mientras que si se otorga el privilegio DROPIN el usuario podrá descartar cualquier alias público.
- **Privilegios que se otorgan a un grupo:** un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### 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 descartar objetos en el esquema CORPDATA.

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO IHAKES
```



## GRANT (etiqueta de seguridad)

Este formato de la sentencia GRANT otorga una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC) a un usuario, grupo o rol para el acceso de lectura, el acceso de grabación o ambos.

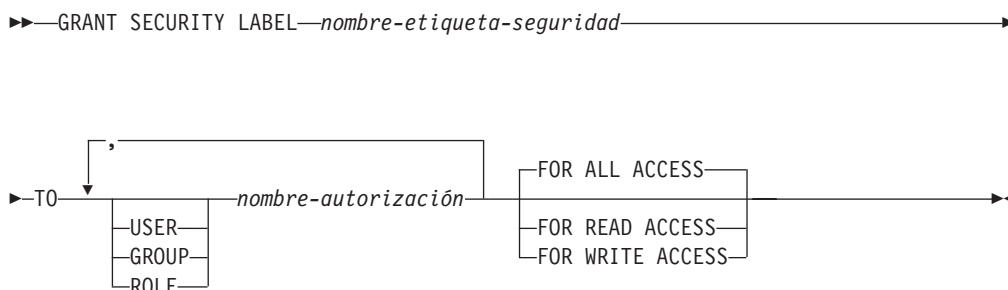
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

### Sintaxis



### Descripción

#### **SECURITY LABEL** *nombre-etiqueta-seguridad*

Otorga la etiqueta de seguridad *nombre-etiqueta-seguridad*. Se debe calificar el nombre con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704).

**TO** Especifica a quién se le otorga la etiqueta de seguridad especificada.

#### **USER**

Especifica que el *nombre-autorización* identifica a un usuario.

#### **GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

#### **ROLE**

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles.

#### **FOR ALL ACCESS**

Indica que la etiqueta de seguridad se otorgará a los accesos de lectura y grabación.

#### **FOR READ ACCESS**

Indica que la etiqueta de seguridad se otorgará solamente para el acceso de lectura.

## GRANT (etiqueta de seguridad)

### FOR WRITE ACCESS

Indica que la etiqueta de seguridad se otorgará solamente para el acceso de grabación.

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- Para cualquier política de seguridad, se puede otorgar como máximo a *nombre-autorización* una etiqueta de seguridad de esa política para el acceso de lectura y una para el acceso de grabación. Si el otorgado ya mantiene una etiqueta de seguridad para el tipo de acceso (lectura o grabación) indicado y que forma parte de la política de seguridad que califica a *nombre-etiqueta-seguridad*, se devuelve un error (SQLSTATE 428GR).
- Si la política de seguridad no está definida para que tenga en cuenta el acceso a través de grupos o roles, se ignorará cualquier etiqueta de seguridad otorgada a un grupo o rol cuando se intente el acceso.
- Si un *nombre-autorización* mantiene diferentes etiquetas de seguridad para los accesos de lectura y de grabación, las etiquetas de seguridad deben cumplir los criterios siguientes (SQLSTATE 428GQ):
  - Si algún componente de las etiquetas de seguridad es del tipo ARRAY, el valor de ese componente debe ser el mismo en ambas etiquetas de seguridad.
  - Si algún componente de las etiquetas de seguridad es del tipo SET, cada elemento del valor de ese componente en la etiqueta de seguridad de grabación debe formar parte también del valor de ese componente en la etiqueta de seguridad de lectura.
  - Si algún componente de las etiquetas de seguridad es del tipo TREE, cada elemento del valor de ese componente en la etiqueta de seguridad de grabación debe ser el mismo o un descendiente de uno de los elementos del valor de ese mismo componente en la etiqueta de seguridad de lectura.

### Notas

- Por omisión, cuando se crea una política de seguridad, sólo se tienen en cuenta las etiquetas de seguridad otorgadas a un usuario individual. Para que se tengan en cuenta los grupos o los roles para la política de seguridad, deberá emitir la sentencia ALTER SECURITY POLICY y especificar USE GROUP AUTHORIZATION o USE ROLE AUTHORIZATION que sea de aplicación.

**Ejemplo**

La sentencia siguiente otorga dos etiquetas de seguridad al usuario GUYLAINE. Se otorga la etiqueta de seguridad EMPLOYEESECLABELREAD para el acceso de lectura y la etiqueta de seguridad EMPLOYEESECLABELWRITE para el acceso de grabación. Ambas etiquetas de seguridad forman parte de la política de seguridad DATA\_ACCESS.

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
TO USER GUYLAINE FOR READ ACCESS
```

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
TO USER GUYLAINE FOR WRITE ACCESS
```

Al mismo usuario se le otorga ahora la etiqueta de seguridad BEGINNER para los accesos de lectura y grabación. Esto no provoca un error, ya que BEGINNER forma parte de la política de seguridad CLASSPOLICY, y las etiquetas de seguridad ya mantenidas forman parte de la política de seguridad DATA\_ACCESS.

```
GRANT SECURITY LABEL CLASSPOLICY.BEGINNER
TO USER GUYLAINE FOR ALL ACCESS
```

# GRANT (privilegios de secuencia)

Esta forma de la sentencia GRANT otorga privilegios para una secuencia.

## Invocación

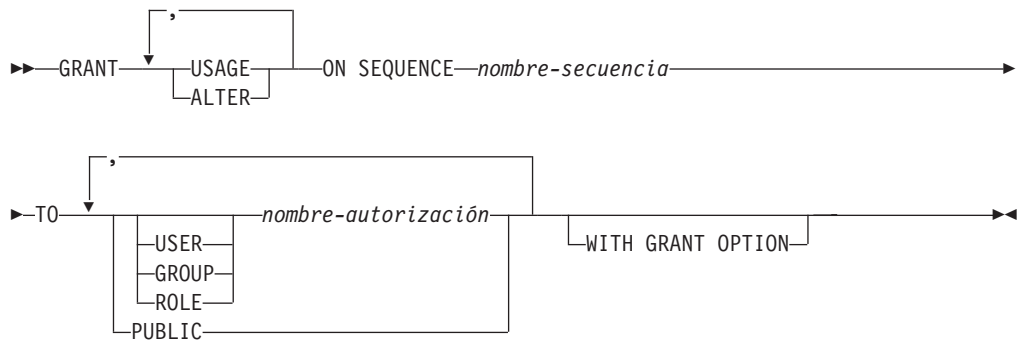
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

## Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH GRANT OPTION para cada privilegio identificado en *nombre-secuencia*
- Autorización ACCESSCTRL o SECADM

## Sintaxis



## Descripción

### USAGE

Otorga el privilegio para poder hacer referencia a una secuencia utilizando una *expresión-nextval* o *expresión-prevval*.

### ALTER

Otorga el privilegio de modificar propiedades de secuencia utilizando la sentencia ALTER SEQUENCE.

### ON SEQUENCE *nombre-secuencia*

Identifica la secuencia para la que van a otorgarse los privilegios especificados. El nombre de la secuencia, incluido un calificador de esquema implícito o explícito, debe identificar de forma exclusiva a una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704).

**TO** Especifica a quién se otorgan los privilegios especificados.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

### PUBLIC

Otorga los privilegios especificados a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

### WITH GRANT OPTION

Permite que el *nombre-autorización* especificado pueda otorgar los privilegios indicados a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar los privilegios indicados a otros si:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar los privilegios especificados desde alguna otra fuente.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- 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). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

## Notas

- **Privilegios que se otorgan a un grupo:** un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW

## GRANT (privilegios de secuencia)

- Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
- Rutina de creación de SQL
- Creación de activador

### Ejemplos

- *Ejemplo 1:* Otorgue a cualquier usuario el privilegio USAGE para una secuencia denominada ORG\_SEQ.

```
GRANT USAGE ON SEQUENCE ORG_SEQ TO PUBLIC
```

- *Ejemplo 2:* Otorgue al usuario BOBBY la capacidad de modificar una secuencia denominada GENERATE\_ID y otorgar este privilegio a otros usuarios.

```
GRANT ALTER ON SEQUENCE GENERATE_ID TO BOBBY WITH GRANT OPTION
```

## 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

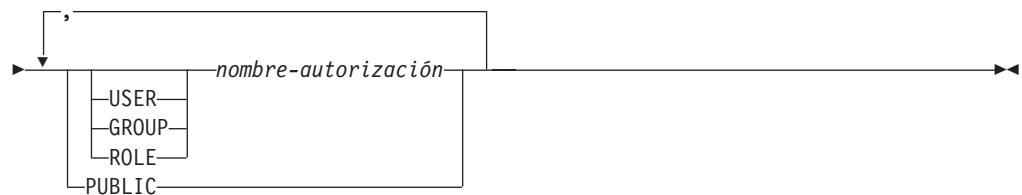
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis

►► GRANT PASSTHRU ON SERVER—*nombre-servidor*—TO



### 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

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 un conjunto de usuarios (ID de autorización) el privilegio de realizar un paso a través de *nombre-servidor*. Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

## GRANT (privilegios de servidor)

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.

### 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 (privilegio SETSESSIONUSER)

Este formato de la sentencia GRANT otorga el privilegio SETSESSIONUSER a uno o más ID de autorización. El privilegio permite a quien lo mantiene utilizar la sentencia SET SESSION AUTHORIZATION para configurar la autorización de la sesión a uno de los ID de autorización de un conjunto especificado.

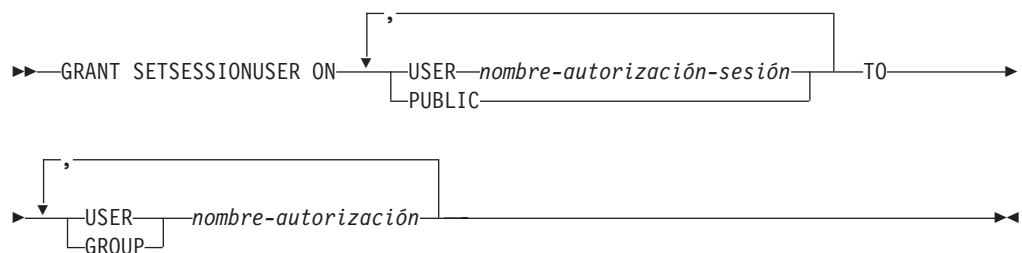
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

### Sintaxis



### Descripción

#### SETSESSIONUSER ON

Otorga el privilegio para asumir la identidad de un nuevo ID de autorización.

#### USER *nombre-autorización-sesión*

Especifica el ID de autorización que el *nombre-autorización* será capaz de asumir, mediante la sentencia SET SESSION AUTHORIZATION. El *nombre-autorización-sesión* debe identificar a un usuario, no a un grupo.

#### PUBLIC

Especifica que el otorgado podrá asumir cualquier ID válido de autorización, mediante la sentencia SET SESSION AUTHORIZATION.

**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 a un 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).

## GRANT (privilegio SETSESSIONUSER)

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER ni GROUP, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.

### Notas

- *Privilegios que se otorgan a un grupo*: un privilegio que se otorga a un grupo no se utiliza para comprobar la autorización de:
  - Las sentencias de DML estáticas de un paquete
  - Una tabla base mientras se procesa una sentencia CREATE VIEW
  - Una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada
  - Rutina de creación de SQL
  - Creación de activador

### Ejemplos

- *Ejemplo 1*: La sentencia siguiente otorga al usuario PAUL la posibilidad de configurar la autorización de la sesión al usuario WALID, y por lo tanto ejecutar sentencias como WALID.

```
GRANT SETSESSIONUSER ON USER WALID
TO USER PAUL
```

- *Ejemplo 2*: La sentencia siguiente otorga al usuario GUYLAINE la posibilidad de configurar la autorización de la sesión para el usuario BOBBY. También le otorga la posibilidad de configurar la autorización de la sesión para los usuarios RICK y KEVIN.

```
GRANT SETSESSIONUSER ON USER BOBBY, USER RICK, USER KEVIN
TO USER GUYLAINE
```

- *Ejemplo 3*: La sentencia siguiente otorga al usuario WALID y a todos los que pertenezcan a los grupos ADMINS y ACCTG la posibilidad de configurar la autorización de la sesión a cualquier usuario.

```
GRANT SETSESSIONUSER ON PUBLIC TO USER WALID, GROUP ADMINS, ACCTG
```

## GRANT (privilegios de espacio de tablas)

Esta forma de la sentencia GRANT otorga privilegios para un espacio de tablas.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

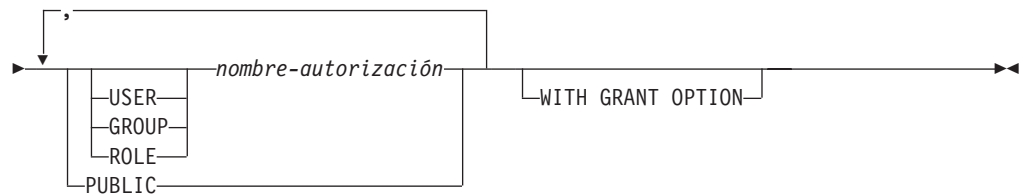
### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH GRANT OPTION para utilizar el espacio de tablas
- Autorización ACCESSCTRL, SECADM, SYSADM o SYSCTRL

### Sintaxis

►► GRANT USE OF TABLESPACE *nombre-espacio-tablas* TO



### 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

#### *nombre-autorización*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

## GRANT (privilegios de espacio de tablas)

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 un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

### **WITH GRANT OPTION**

Permite que el *nombre-autorización* especificado otorgue el privilegio USE a otros usuarios.

## **Normas**

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.

## **Ejemplo**

Otorgar 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
```

## GRANT (privilegios de tabla, vista o apodo)

Esta forma de la sentencia GRANT otorga privilegios en una tabla, vista o apodo.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

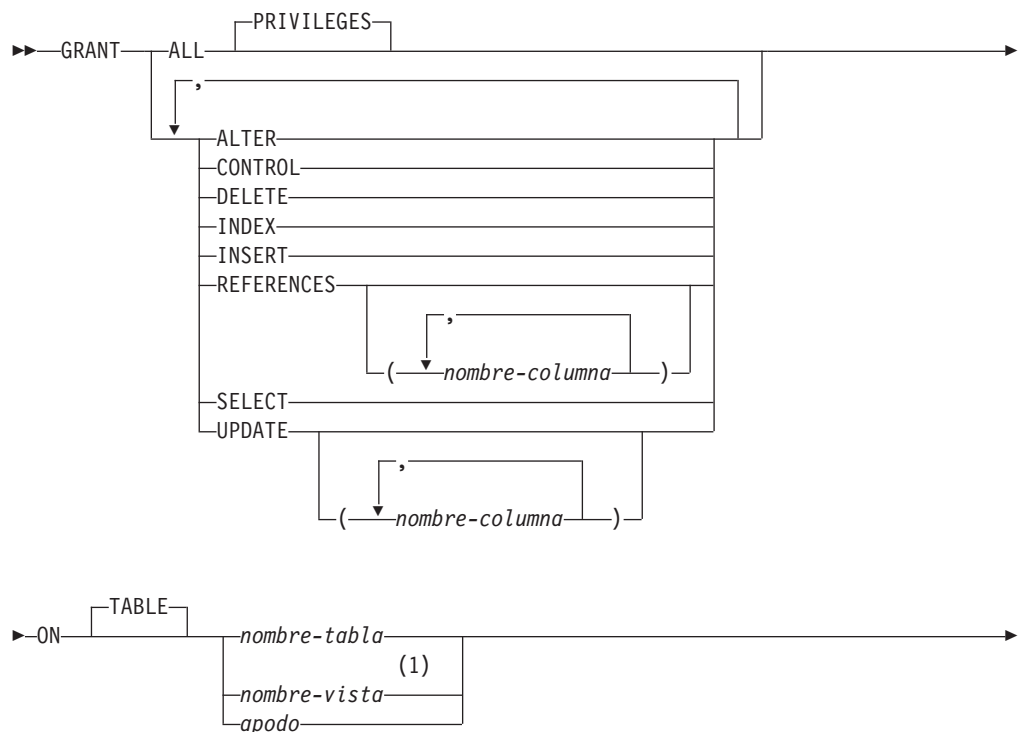
### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

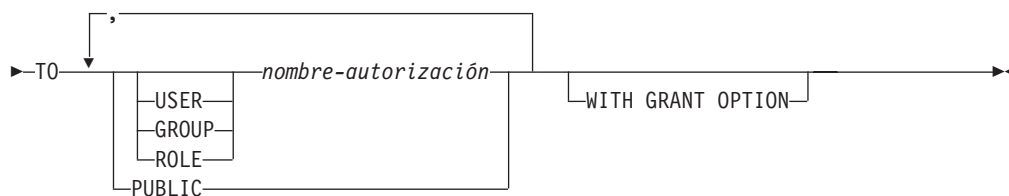
- Privilegio CONTROL sobre la tabla, vista o apodo referenciado.
- WITH GRANT OPTION para cada privilegio identificado. Si se especifica ALL, el ID de autorización debe tener algún privilegio otorgable en la tabla, vista o apodo identificado.
- Autorización ACCESSCTRL o SECADM

La autorización ACCESSCTRL o SECADM es necesaria para otorgar el privilegio CONTROL o para otorgar privilegios sobre tablas y vistas de catálogo.

### Sintaxis



## GRANT (privilegios de tabla, vista o apodo)



### Notas:

- 1 Los privilegios ALTER, INDEX y REFERENCES no son aplicables a las vistas.

## 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 sobre la tabla, vista o apodo, o la autorización ACCESSCTRL o SECADM, 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 descartar una clave primaria o una restricción de unicidad en una tabla base.
- Crear o descartar una clave foránea en una tabla base.  
También es necesario el privilegio REFERENCES en cada columna de la tabla padre.
- Crear o descartar una restricción de comprobación en una tabla base.
- Crear un activador en una tabla base.
- Añadir, restablecer o descartar una opción de columna para un apodo.
- Cambiar un nombre de columna de apodo o tipo de datos.
- Añadir o cambiar un comentario en una tabla base o en un 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 otorgar los privilegios mencionados anteriormente (excepto CONTROL) a otros.
- La posibilidad de descartar la tabla base, vista o apodo.

Esta posibilidad no puede extenderse a otros sobre la base de poseer el privilegio CONTROL. La única manera en que puede extenderse es otorgando el privilegio CONTROL en sí y sólo puede realizarse con un ID de autorización con autoridad ACCESSCTRL o SECADM.

## GRANT (privilegios de tabla, vista o apodo)

- La posibilidad de ejecutar el programa de utilidad RUNSTATS en la tabla e índices.
- La posibilidad de ejecutar el programa de utilidad REORG en la tabla.
- La posibilidad de emitir la sentencia SET INTEGRITY para una tabla base, tabla de consulta materializada o tabla de etapas.

La persona que define una tabla base, una tabla de consulta materializada, una tabla de etapas o un apodo automáticamente recibe 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 descartar el índice o la especificación de índice). Asimismo, 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 descartar 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 siguientes:

- Autorización ACCESSCTRL o SECADM
- Privilegio CONTROL sobre 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.

El privilegio puede otorgarse para un apodo, aunque no pueden definirse claves foráneas para apodos de referencia.

### REFERENCES (*nombre-columna,...*)

Otorga el privilegio para crear y descartar 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.

## GRANT (privilegios de tabla, vista o apodo)

- Ejecuta el programa de utilidad EXPORT en la tabla o vista.

### 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 siguientes:

- Autorización ACCESSCTRL o SECADM
- Privilegio CONTROL sobre la tabla o vista
- UPDATE WITH GRANT OPTION en la tabla o vista

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 deben otorgar los privilegios.

No puede otorgarse ningún privilegio para una vista no operativa o para una tabla de consulta materializada 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

#### *nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

Un privilegio que se ha otorgado a un grupo no se utiliza para la comprobación de autorizaciones:

- En las sentencias de DML estáticas de un paquete
- En una tabla base mientras se procesa una sentencia CREATE VIEW
- En una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consulta materializada

En DB2 Database para Linux, UNIX y Windows, 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;
```



## GRANT (privilegios de tabla, vista o apodo)

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 un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”. (Se han eliminado las restricciones anteriores que se aplicaban a la utilización de los privilegios que se otorgan a PUBLIC para las sentencias de SQL estático y la sentencia CREATE VIEW.)

### 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

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define *nombre-autorización* según el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se presupone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se presupone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se presupone ROLE.
- 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). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.) Si se especifica el privilegio CONTROL, los privilegios sólo se otorgarán si el ID de autorización de la sentencia tiene autorización ACCESSCTRL o SECADM (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 puede afectar a las subtablas. Por ejemplo, una actualización que especifique la

## GRANT (privilegios de tabla, vista o apodo)

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.
- *Alternativas de sintaxis*: Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Se tolera y se pasa por alto la sintaxis siguiente:
  - PUBLIC AT ALL LOCATIONS

### 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
```

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 (privilegios de tabla, vista o apodo)

```
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 sobre CALENDAR a otros usuarios a menos que tenga autorización ACCESSCTRL o SECADM.

- *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 mediante la creación de 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 de carga de trabajo)

Este formato de la sentencia GRANT otorga el privilegio USAGE en una carga de trabajo.

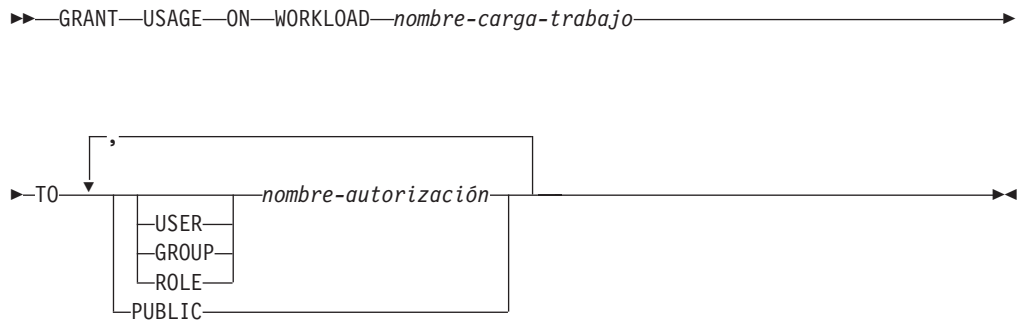
## Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

## Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL, SECADM o WLMADM.

## Sintaxis



## Descripción

### USAGE

Otorga el privilegio para utilizar una carga de trabajo. Las unidades de trabajo que somete un usuario sólo se correlacionarán con una carga de trabajo en la que el usuario tenga un privilegio USAGE. Un usuario con autorización SYSADM o DBADM tiene automáticamente un privilegio USAGE sobre cualquier carga de trabajo que exista en el servidor actual.

### ON WORKLOAD *nombre-carga-trabajo*

Identifica la carga de trabajo a la que va a otorgarse el privilegio USAGE. Este nombre consta de una sola parte. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El nombre no puede ser 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

**TO** Especifica a quién se otorga el privilegio USAGE.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica a un grupo.

### ROLE

Especifica que el *nombre-autorización* identifique un rol existente en el servidor actual (SQLSTATE 42704).

## GRANT (privilegios de carga de trabajo)

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles. 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 USAGE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte "Autorizaciones, privilegios y propiedad de objetos".

## **Normas**

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se asume GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.

## **Notas**

- La sentencia GRANT no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.
- Si la base de datos se crea con la opción RESTRICT, el privilegio USAGE de la carga de trabajo de usuario por omisión, SYSDEFAULTUSERWORKLOAD, debe otorgarse de modo explícito por medio de un usuario que tenga autorización DBADM. Si la base de datos se crea sin la opción RESTRICT, el privilegio USAGE de SYSDEFAULTUSERWORKLOAD se otorgará a PUBLIC en el momento de creación de la base de datos.

## **Ejemplo**

Otorgar al usuario LISA la capacidad de utilizar la carga de trabajo CAMPAIGN.

```
GRANT USAGE ON WORKLOAD CAMPAIGN TO USER LISA
```

---

### GRANT (privilegios de objeto XSR)

Este formato de la sentencia GRANT otorga el privilegio USAGE sobre un objeto XSR.

#### Invocación

La sentencia GRANT puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Se necesita una de las autorizaciones siguientes:

- Autorización ACCESSCTRL o SECADM
- Propietario del objeto XSR, como está registrado en la columna OWNER de la vista de catálogo SYSCAT.XSROBJECTS

#### Sintaxis

```
►► GRANT USAGE ON XSROBJECT nombre-objeto TO PUBLIC ◀◀
```

#### Descripción

**ON XSROBJECT** *nombre-objeto*

Este nombre identifica el objeto XSR sobre el que se ha otorgado el privilegio USAGE. El *nombre-objeto* (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este nombre, se devuelve un error (SQLSTATE 42704).

**TO PUBLIC**

Otorga el privilegio USAGE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

#### Ejemplo

Otorgar a todos los usuarios privilegio de uso sobre el esquema XML MYSCHEMA:

```
GRANT USAGE ON XSROBJECT MYSCHEMA TO PUBLIC
```



## IF

verdadera. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL o de una sentencia de SQL compuesto (compilado). Consulte *sentencia-procedimiento-SQL* en la sentencia de “SQL compuesto (compilado)”.

### *sentencia-función-SQL*

Especifica la sentencia que debe invocarse si la *condición-búsqueda* anterior es verdadera. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una sentencia de SQL compuesto (en línea), un activador de SQL, una función de SQL o un método de SQL. Consulte *sentencia-función-SQL* en “FOR”.

## Ejemplo

El procedimiento de SQL siguiente acepta dos parámetros IN: un número de empleado (*employee\_number*) y una tarifa de empleado (*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;
 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
```



## INCLUDE

La sentencia INCLUDE inserta declaraciones en un programa fuente.

### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable.

### Autorización

No se necesita.

### Sintaxis

```

▶▶ INCLUDE { SQLCA
 SQLDA
 nombre }

```

### Descripción

#### SQLCA

Indica que se debe incluir la descripción de un área de comunicaciones SQL (SQLCA).

#### SQLDA

Indica que se debe incluir la descripción de un área de descriptores SQL (SQLDA).

#### *nombre*

Identifica un archivo externo que contiene el texto que se debe incluir en el programa fuente que se está precompilando. Puede ser un identificador de 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.

### 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 del sistema principal especificado por *nombre*. Si es superior a 18 bytes o contiene caracteres que no están permitidos en un identificador de SQL, debe ir entre comillas simples. Las sentencias INCLUDE *nombre* pueden anidarse, aunque no cíclicamente (por ejemplo, si A y B son módulos y A contiene una sentencia INCLUDE *nombre*, no es válido que A llame a B y que, a continuación, 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 del lenguaje principal.

## INCLUDE

### 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

La sentencia INSERT inserta filas en una tabla, apodo o vista, o en las tablas, apodos o vistas subyacentes de la selección completa especificada.

La inserción de una fila en un apodo inserta la fila en el objeto de fuente de datos al que hace referencia el apodo. La inserción de una fila en una vista también inserta la fila en la tabla en la que se basa la vista, si no se ha definido ningún activador INSTEAD OF para la operación de inserción en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio INSERT para la tabla, vista o apodo de destino
- Privilegio CONTROL sobre la tabla, vista o apodo de destino
- Autorización DATAACCESS

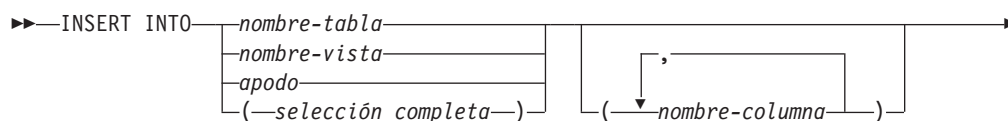
Además, para cada tabla, vista o apodo al que se haga referencia en cualquier selección completa utilizada en la sentencia INSERT, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

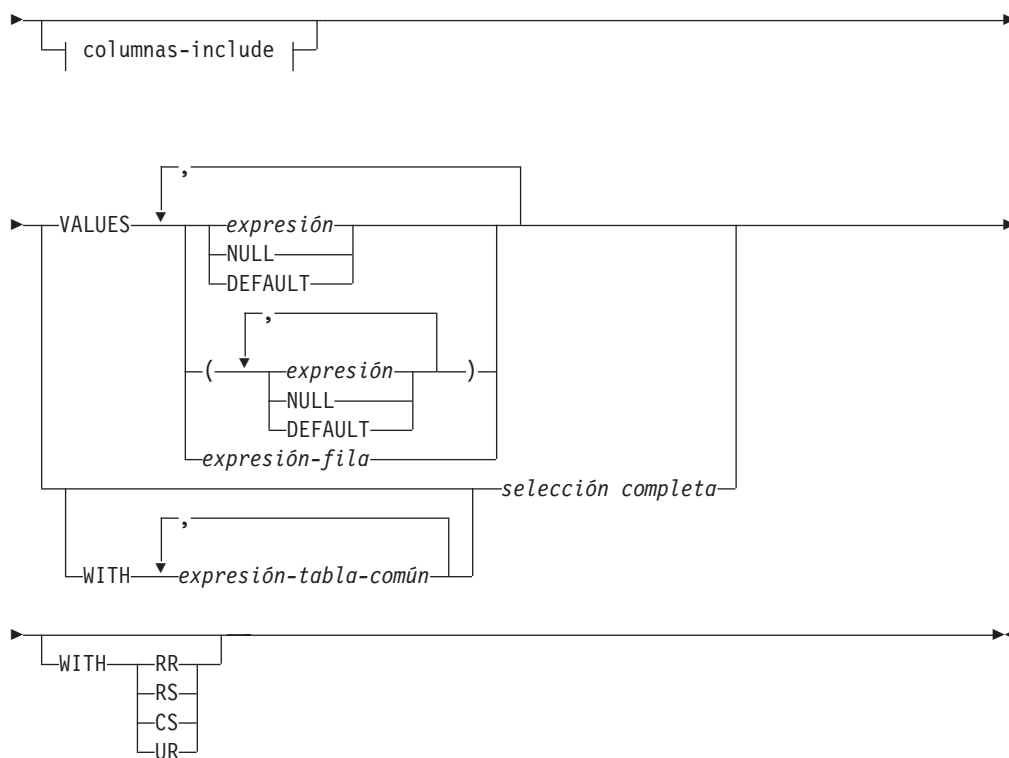
No se comprueban los privilegios GROUP para las sentencias INSERT estáticas.

Si el destino de la operación de inserción es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

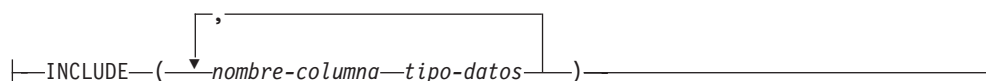
### Sintaxis



## INSERT



### columnas-include:



## Descripción

**INTO** *nombre-tabla*, *nombre-vista*, *apodo* o (*selección completa*)

Identifica el objeto de la operación de inserción. El nombre debe identificar uno de los objetos siguientes:

- Una tabla, vista o apodo que existe en el servidor de aplicaciones
- Una tabla o vista en el servidor remoto que se especifica mediante un nombre-objeto-remoto

El objeto no debe ser una tabla de catálogo, una tabla de consulta materializada mantenida por el sistema, una vista de una tabla de catálogo o una vista de sólo lectura, a menos que se haya definido un activador `INSTEAD OF` para la operación de inserción de la vista del asunto. Las filas que se insertan en un apodo se colocan en el objeto de fuente de datos al que hace referencia el apodo.

Si el objeto de la operación de actualización es una selección completa, esta debe ser insertable, según lo definido en el elemento "Vistas insertables" de Notas de la descripción de la sentencia `CREATE VIEW`.

Si el objeto de la operación de inserción es un apodo, los valores de variable de indicador ampliado `DEFAULT` y `UNASSIGNED` no deben utilizarse (SQLSTATE 22539).

Si no existe ningún activador `INSTEAD OF` para la operación de inserción en esta vista, no podrá insertarse un valor en una columna de vista que se haya obtenido de los elementos siguientes:

- Una constante, expresión o función escalar
- La misma columna de tabla base que otra columna de la vista.

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.

Una fila podrá insertarse en una vista o una selección completa que se ha definido utilizando `UNION ALL` si la fila satisface las restricciones de comprobación de exactamente una de las tablas base subyacentes. Si una fila satisface las restricciones de comprobación de más de una tabla, o no satisface la de ninguna tabla, se devolverá un error (SQLSTATE 23513).

Una fila no se puede insertar en una vista o selección completa que se haya definido con `UNION ALL` si alguna tabla base de la vista contiene un activador `BEFORE` y el activador `BEFORE` contiene una operación `UPDATE`, `DELETE` o `INSERT` o invoca una rutina que contiene ese tipo de operaciones (SQLSTATE 42987).

*(nombre-columna,...)*

Especifica las columnas para las que se proporcionan valores de inserción. Cada nombre debe identificar una columna de la tabla, vista o apodo especificado, o una columna de la selección completa. La misma columna no se puede identificar más de una vez. Si las variables de indicador ampliado no están habilitadas, no puede identificarse una columna que no pueda aceptar la inserción de valores (por ejemplo, una columna basada en una expresión).

La omisión de la lista de columnas es una especificación implícita de una lista en la que cada columna de la tabla (que no está oculta implícitamente) o vista, o cada elemento de la lista de selección de la selección completa 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.

*columns-include*

Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia `INSERT` cuando está anidada en la cláusula `FROM` de una selección completa. Las *columns-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

#### **INCLUDE**

Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia `INSERT`. Esta cláusula sólo se puede especificar si la sentencia `INSERT` está anidada en la cláusula `FROM` de la selección completa.

*nombre-columna*

Especifica una columna de la tabla de resultados intermedia de la sentencia `INSERT`. El nombre no puede coincidir con el nombre de otra columna incluye ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

*tipo-datos*

Especifica el tipo de datos de la columna incluye. El tipo de datos debe ser uno que reciba soporte de la sentencia `CREATE TABLE`.

## VALUES

Especifica una o varias filas de valores que se deben insertar.

Cada fila especificada en la cláusula VALUES debe ser asignable a la lista de columnas implícita o explícita y las columnas identificadas en la cláusula INCLUDE, a menos que se utilice una variable de fila. Cuando se especifica una lista de valores de fila en paréntesis, el primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etc. Cuando se especifica una expresión de fila, el número de campos en el tipo de fila debe coincidir con el número de nombres en la lista de columnas implícitas o explícitas.

### *expresión*

Una *expresión* puede ser cualquier expresión definida en el tema "Expresiones". Si la *expresión* es un tipo de fila, no debe aparecer entre paréntesis. Si la *expresión* es una variable, la variable del lenguaje principal puede incluir una variable de indicador o, en el caso de una estructura de sistema principal, una matriz de indicador, que esté habilitada para las variables de indicador ampliado. Si las variables de indicador ampliado están habilitadas, no deben utilizarse los valores para la variable de indicador ampliado por omisión (-5) o no asignado (-7) (SQLSTATE 22539) si cualquiera de las afirmaciones siguientes es cierta:

- La expresión es más compleja que una sola variable del lenguaje principal con conversiones explícitas.
- La columna de destino tiene un tipo de datos de tipo estructurado.

## NULL

Especifica el valor nulo y sólo debe especificarse para las columnas con posibilidad de nulos.

## DEFAULT

Especifica que se debe utilizar el valor por omisión. El resultado de especificar DEFAULT depende del modo en que se haya definido la columna, del siguiente 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 ROW CHANGE TIMESTAMP, el gestor de bases de datos genera el valor para cada fila insertada como una indicación de fecha y hora que es exclusiva para cada partición de tabla en la partición de base de datos.
- Si se utiliza la cláusula WITH DEFAULT, el valor insertado será el valor tal como se ha definido para la columna (consulte *cláusula-por-omisión* en "CREATE TABLE").
- 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).
- Cuando la inserción se realice en un apodo, la palabra clave DEFAULT se pasará por medio de la sentencia INSERT a la fuente de datos sólo si la fuente de datos da soporte a la palabra clave DEFAULT en su sintaxis de lenguaje de consulta.

*expresión-fila*

Especifica cualquier expresión de fila del tipo descrito en "Expresiones de fila" que no incluya un nombre de columna. El número de campos de la fila debe coincidir con el destino de la inserción y cada campo de la fila debe poderse asignar a la columna correspondiente.

**WITH** *expresión-tabla-común*

Define una expresión de tabla común para utilizarla con la selección completa que va a continuación.

*selección completa*

Especifica un conjunto de filas nuevas en la forma de la tabla de resultados de una selección completa. Puede haber una, más de una o ninguna. Si la tabla de resultados 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 de resultados 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.

Si la expresión que especifica el valor de una columna de resultados es una variable, la variable de lenguaje principal puede incluir una variable de indicador que esté habilitada para las variables de indicador ampliado. Si las variables de indicador ampliado están habilitadas y la expresión es más que una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita, los valores DEFAULT o UNASSIGNED de variable de indicador ampliado (SQLSTATE 22539) no deben utilizarse. Los efectos de los valores DEFAULT o UNASSIGNED se aplican a las columnas de destino correspondientes de la *selección completa*.

**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. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

**Normas**

- **Activadores:** Las sentencias INSERT pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores insertados. Si una operación de inserción en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se han realizado en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.

## INSERT

- **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 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 indicación de fecha y hora si la serie representa una fecha, hora o indicación de fecha y hora.
- **Asignación:** Los valores de inserción se asignan a las columnas de acuerdo con normas de asignación específicas.
- **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 obtener información acerca de las normas que rigen esta situación, consulte "CREATE VIEW".
- **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.
- **Valores XML:** Un valor que se inserta en una columna XML debe ser un documento XML con el formato correcto (SQLSTATE 2200M).
- **Política de seguridad:** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
  - Acceso de grabación a todas las columnas protegidas para las que se ha proporcionado explícitamente un valor de datos (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)

El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:

- No se proporciona explícitamente un valor para la columna DB2SECURITYLABEL
- Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso



de grabación para dicho valor y la política de seguridad se crea con la opción `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`

- **Uso de variable de indicador ampliado:** Si se habilita, los valores de variable de indicador negativos fuera del rango entre -1 y -7 no pueden ser una entrada (SQLSTATE 22010). Además, si se habilitan, los valores `DEFAULT` y `UNASSIGNED` de variable de indicador ampliado no deben aparecer en contextos en los que no estén soportados (SQLSTATE 22539).
- **Variables de indicador ampliado:** En una sentencia `INSERT`, un valor `UNASSIGNED` tiene como efecto establecer una columna en su valor por omisión.

Si la columna de destino es una columna definida como `GENERATED ALWAYS`, se le debe asignar la palabra clave `DEFAULT` o los valores `DEFAULT` o `UNASSIGNED` basados en una variable de indicador ampliado (SQLSTATE 428C9).

## Notas

- Tras la ejecución de una sentencia `INSERT`, el valor de la tercera variable de la parte `SQLERRD(3)` de la `SQLCA` indica el número de filas que se han pasado a la operación de inserción. En el contexto de una sentencia de procedimiento de `SQL`, el valor puede recuperarse utilizando la variable `ROW_COUNT` de la sentencia `GET DIAGNOSTICS`. `SQLERRD(5)` contiene la cuenta de todas las operaciones de inserción, actualización y supresión activadas.
- Salvo 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.
- 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.

## INSERT

El primer valor generado por DB2 es el valor de la especificación START WITH para la columna de identidad.

- Cuando se ha insertado un valor para una columna de identidad de tipo diferenciado definida por el usuario, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior en el tipo de fuente antes de realizarse el cálculo.)
- 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 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.

- Al realizar una inserción en una columna GENERATED ALWAYS ROW CHANGE TIMESTAMP, DB2 genera siempre un valor para la columna y los usuarios no deben especificar un valor en el momento de realizar la inserción (SQLSTATE 428C9). El valor generado por DB2 es exclusivo para cada fila insertada en la partición de base de datos.
- Cuando se inserta en una columna 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 de la columna IDENTITY, se debe crear un índice de unicidad en 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 se define como una columna de identidad y, por tanto, el valor insertado en esta columna es generado por DB2.

- Las normas para insertar en una columna de identidad utilizando una subselección son similares a las normas para una inserción mediante una cláusula VALUES. Sólo se puede especificar un valor para una columna de identidad si esta 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 de ejemplo 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 la columna está definida como columna de identidad GENERATED ALWAYS o como una columna ROW CHANGE TIMESTAMP, es posible especificar una cláusula VALUES con 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 o ROW CHANGE TIMESTAMP.

```
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.

- 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 descartar 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, descartar 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.

- **Variables de indicador ampliado y activadores de inserción:** El uso de variables de indicador ampliado no provoca ningún cambio en la activación de los activadores de inserción. Si todas las columnas de la lista de columnas implícitas o explícitas se han asignado a un valor DEFAULT o UNASSIGNED basado en una variable de indicador ampliado, se intenta realizar una inserción donde todas las columnas tienen sus respectivos valores por omisión y, si resulta satisfactoria, se activa el activador de inserción.
- **Variables de indicador ampliado y comprobaciones de errores diferidas:** Cuando se habilitan variables de indicador ampliado, la validación que de otro modo se realizaría en la preparación de sentencias para reconocer una inserción en una columna no actualizable se difiere hasta la ejecución de sentencias. La necesidad de notificar un error se determina únicamente durante la ejecución.
- **Inserción en tablas con columnas de inicio de fila, de fin de fila o de ID de inicio de transacción:** Cuando se inserta una fila en una tabla con estas columnas generadas (por ejemplo, una tabla temporal de período de sistema), el gestor de bases de datos asigna valores a las columnas siguientes:
  - A una columna de inicio de fila se le asigna un valor que se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exige que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla, o que se suprima una tabla temporal de período de sistema. El gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de inicio de fila en las transacciones. Si se actualizan varias filas en una sola transacción de SQL, los valores de la columna de inicio de fila son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.
  - A una columna de fin de fila se le asigna el valor máximo para el tipo de datos de la columna (9999-12-30-00.00.00.000000000000).
  - A una columna de ID de inicio de transacción se le asigna un valor de indicación de fecha y hora exclusivo o el valor nulo. El valor nulo se asigna a la columna de ID de inicio de transacción si la columna puede contener nulos. De otro modo, el valor se genera a partir de una lectura del reloj de la hora del día durante la ejecución de la primera sentencia de cambio de datos en la transacción que requiera que se asigne un valor a la columna de inicio de fila o a la columna de ID de inicio de transacción de una tabla, o que se suprima una tabla temporal de período de sistema. Si se insertan varias filas en una sola transacción de SQL, los valores de la columna de ID de inicio de transacción son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.
- **Inserción en una tabla temporal de período de sistema:** Cuando se inserta una fila en una tabla temporal de período de sistema, el gestor de bases de datos asigna valores a las columnas, como se indica para las tablas con columnas de

inicio de fila, de fin de fila o de ID de inicio de transacción. Asimismo, cuando se inserta una fila, no se añade ninguna fila a la tabla histórica asociada a la tabla temporal de período de sistema.

- **Inserción en tablas temporales de período de aplicación:** Se devuelve un error cuando se inserta una fila en una tabla temporal de período de aplicación y se cumplen las condiciones siguientes:
  - La tabla temporal de período de aplicación tiene una clave primaria o una restricción de unicidad con la cláusula BUSINESS\_TIME WITHOUT OVERLAPS definida o un índice exclusivo con la cláusula BUSINESS\_TIME WITHOUT OVERLAPS definida.
  - El período que definen las columnas de inicio y fin del período BUSINESS\_TIME se solapa con el período que definen las columnas de inicio y fin del período BUSINESS\_TIME para otra fila que coincide con otras columnas de la misma restricción de unicidad o índice exclusivo.
- **Consideraciones sobre una sentencia INSERT sin una lista de columnas:** Una sentencia INSERT sin una lista de columnas no incluye columnas ocultas implícitamente. Las columnas que se han definido como ocultas implícitamente y que no son nulas deben tener definido un valor por omisión.

## 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),

## INSERT

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);
```

- *Ejemplo 6:* Especifique una sentencia INSERT como *referencia-tabla-cambio-datos* dentro de una sentencia SELECT. Defina una columna include adicional cuyos valores se especifican en la cláusula VALUES, que luego se utiliza como una columna de clasificación para las filas insertadas.

```
SELECT INORDER.ORDERNUM
FROM NEW TABLE (INSERT INTO ORDERS(CUSTNO)INCLUDE (INSERTNUM INTEGER)
VALUES(:CNUM1, 1), (:CNUM2, 2)) InsertedOrders
ORDER BY INSERTNUM;
```

- *Ejemplo 7:* Utilice una sentencia del programa C para añadir un documento a la tabla DOCUMENTS. Obtenga valores para la columna ID del documento (DOCID) y la columna de datos del documento (XMLDOC) desde una variable del lenguaje principal que se vincula con un SQL TYPE IS XML AS BLOB\_FILE.

```
EXEC SQL INSERT INTO DOCUMENTS
(DOCID, XMLDOC) VALUES (:docid, :xmldoc)
```

- *Ejemplo 8:* para las siguientes sentencias INSERT, suponga que la tabla SALARY\_INFO está definida con tres columnas y que la última columna es una columna ROW CHANGE TIMESTAMP implícitamente oculta. En la sentencia siguiente, se hace referencia explícita a la columna implícitamente oculta en la lista de columnas y se proporciona un valor para dicha columna en la cláusula VALUES.

```
INSERT INTO SALARY_INFO (LEVEL, SALARY, UPDATE_TIME)
VALUES (2, 30000, CURRENT_TIMESTAMP)
```

La sentencia siguiente INSERT utiliza una lista de columnas implícitas. Una lista de columnas implícitas no incluye columnas implícitamente ocultas, de manera que la cláusula VALUES sólo contiene valores para las otras dos columnas.

```
INSERT INTO SALARY_INFO VALUES (2, 30000)
```

En este caso, la columna UPDATE\_TIME debe definirse para que tenga un valor por omisión y el valor por omisión se utiliza para la fila que se inserta.

## ITERATE

La sentencia ITERATE hace que el flujo de control vuelva al principio de un bucle con etiqueta.

### Invocación

Esta sentencia se puede incluir en:

- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se necesita.

### 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.

### Ejemplo

Este ejemplo utiliza un cursor para devolver información para un nuevo departamento. Si se ha invocado el manejador de condiciones *not\_found*, el flujo de control sale del bucle. Si el valor de *v\_dept* es 'D11', una sentencia ITERATE vuelve a pasar el flujo del 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;
 ELSE
 INSERT INTO DEPARTMENT (deptno, deptname, admrdept)
 VALUES (v_dept, v_deptname, v_admdept);
 END IF;
 END LOOP;
END;
```

## ITERATE

```
 ITERATE ins_loop;
 END IF;
 INSERT INTO department (deptno, deptname, admrdept)
 VALUES ('NEW', v_deptname, v_admdept);
END LOOP;
CLOSE c1;
END
```



## LEAVE

La sentencia LEAVE transfiere el control del programa hacia fuera de un bucle o de una sentencia compuesta.

### Invocación

Esta sentencia se puede incluir en:

- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis

►►—LEAVE—*etiqueta*—◄◄

### Descripción

*etiqueta*

Especifica la etiqueta de la sentencia FOR, LOOP, REPEAT, WHILE o sentencia compuesta cuya ejecución debe concluir.

### Notas

- 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 de resultados.

### Ejemplo

Este ejemplo contiene un bucle que lee datos para el cursor *c1*. Si el valor de la variable de SQL *at\_end* no es cero, la sentencia LEAVE transfiere el control 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:
```

## LEAVE

```
LOOP
 FETCH c1 INTO v_firstname, 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
```

## LOCK TABLE

La sentencia LOCK TABLE impide que procesos de aplicación simultáneos utilicen o cambien una tabla. El bloqueo se libera cuando la unidad de trabajo que emite la sentencia LOCK TABLE se confirma o se termina.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio SELECT en la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

### Sintaxis

```

▶▶ LOCK TABLE nombre-tabla | apodo IN SHARE | EXCLUSIVE MODE

```

### Descripción

*nombre-tabla* o *apodo*

Identifica la tabla o el apodo. El *nombre-tabla* debe identificar una tabla que exista en el servidor de aplicaciones, pero no debe identificar una tabla de catálogo, una tabla temporal creada o 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). Cuando se ha especificado un apodo, DB2 bloquea el objeto subyacente (es decir, una tabla o una vista) de la fuente de datos a la que el apodo hace referencia.

#### 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 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 de base de datos del grupo de particiones de base de datos

## LOCK TABLE

(la partición de base de datos que tiene el número más bajo) y, después, en las demás particiones de base de datos. Si se interrumpe la sentencia LOCK TABLE, la tabla puede estar bloqueada en algunas particiones de base de datos y en otras no. Si ocurre esto, emita otra sentencia LOCK TABLE para completar el bloqueo de todas las particiones de base de datos o emita una sentencia COMMIT o ROLLBACK para liberar los bloqueos actuales.

- Esta sentencia afecta a todas las particiones de base de datos del grupo de particiones de base de datos.
- Para tablas particionadas, el único bloqueo adquirido en la sentencia LOCK TABLE se encuentra en el nivel de la tabla; no se han adquirido bloqueos de particiones de datos.

### Ejemplo

Obtenga un bloqueo en la tabla EMP. No permita que otros programas lean o actualicen la tabla.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

## LOOP

La sentencia LOOP repite la ejecución de una sentencia o grupo de sentencias.

### Invocación

Esta sentencia se puede incluir en:

- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se requieren privilegios para invocar la sentencia LOOP. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia LOOP.

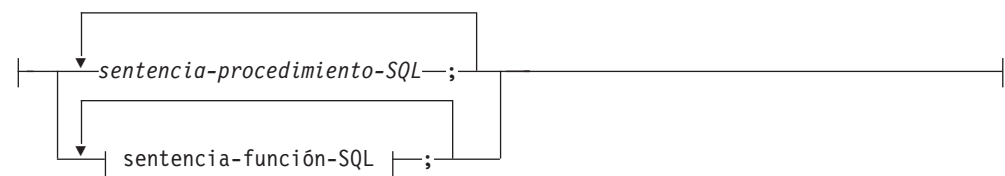
### Sintaxis

```

▶ ──┬──────────┬── LOOP ──┬── sentencia-rutina-SQL ──┬── END LOOP ──┬──▶
 │ │ │ │ │
 └─ etiqueta: ─┘ └─ etiqueta ─┘

```

#### sentencia-rutina-SQL:



### 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 de SQL que se deben invocar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL o una sentencia de SQL compuesto (compilado). Consulte *sentencia-procedimiento-SQL* en la sentencia de “SQL compuesto (compilado)”.

#### *sentencia-función-SQL*

Especifica las sentencias de SQL que se deben invocar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función SQL, un método SQL o una sentencia de SQL compuesto (en línea). Consulte *sentencia-función-SQL* en “FOR”.

## LOOP

### Ejemplo

Este procedimiento utiliza una sentencia LOOP para leer valores de la tabla employee. Cada vez que se repite el bucle, el parámetro OUT *counter* se incrementa y el valor de *v\_midinit* se comprueba para asegurarse de que el valor no es un espacio (' '). Si *v\_midinit* es un espacio, la sentencia LEAVE pasa el flujo de control 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;
 END LOOP fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END
```

## MERGE

La sentencia MERGE actualiza un destino (una tabla o vista, o las tablas o vistas subyacentes de una selección completa) utilizando datos de una fuente (resultado de una referencia de tabla).

Las filas del destino que coinciden con la fuente se pueden suprimir o actualizar, según se especifique, y las filas que no existen en el destino se pueden insertar. Si se actualiza, suprime o inserta una fila en una vista, se actualiza, suprime o inserta la fila en las tablas en las que se basa la vista.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Si se especifica una operación de inserción, privilegio INSERT sobre la tabla o vista; si se especifica una operación de suprimir, privilegio DELETE sobre la tabla o vista; y si se especifica una operación de actualizar:
  - Privilegio UPDATE sobre la tabla o vista
  - Privilegio UPDATE sobre cada columna que se va a actualizar
- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

Los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio SELECT en cada tabla o vista identificada en la *referencia-tabla*
- Privilegio CONTROL en las tablas o vistas identificadas en la *referencia-tabla*
- Autorización DATAACCESS

Si *condición-búsqueda*, *operación-inserción* o *cláusula-asignación* incluye una subconsulta, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio SELECT en cada tabla o vista identificada en la subconsulta
- Privilegio CONTROL en las tablas o vistas identificadas en la subconsulta
- Autorización DATAACCESS

Si se especifica una expresión que se refiere a una función, el conjunto de privilegios debe incluir la autoridad que sea necesaria para ejecutar la función.

### Sintaxis

```

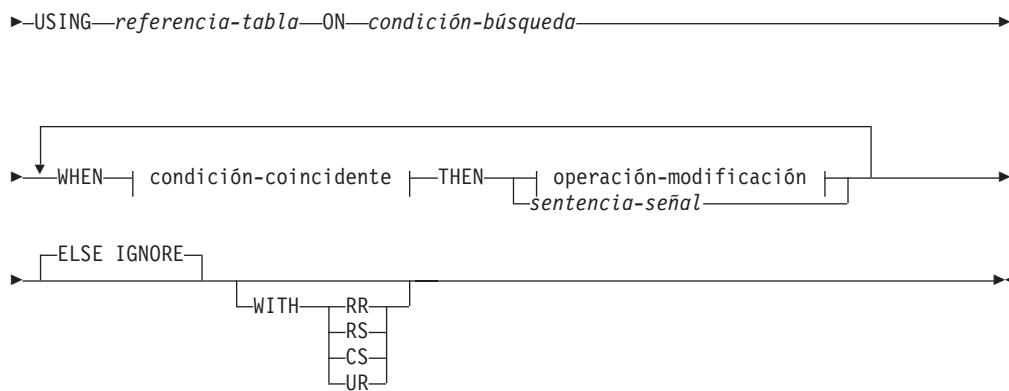
➤➤ MERGE INTO nombre-tabla

 nombre-vista
 (—selección completa—)

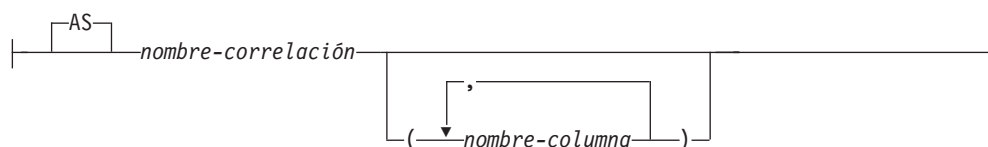
 cláusula-correlación


```

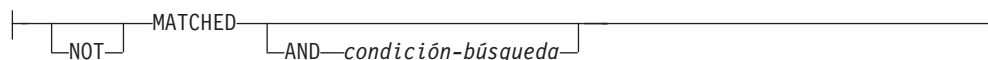
# MERGE



## cláusula-correlación:



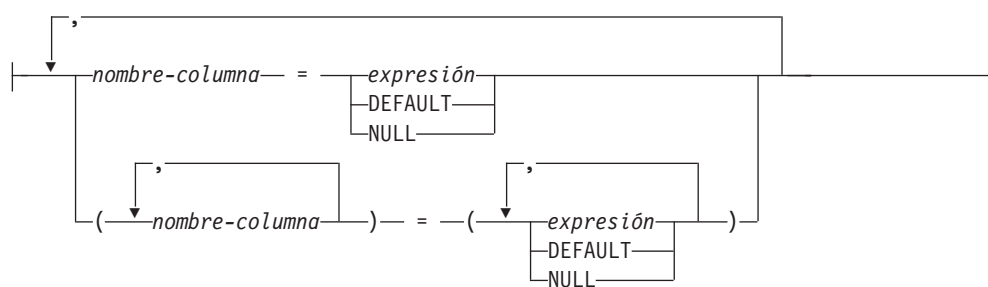
## condición-coincidente:



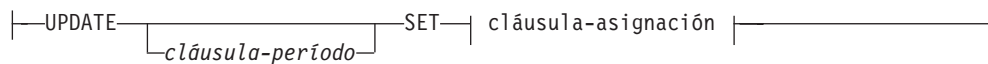
## operación-modificación:



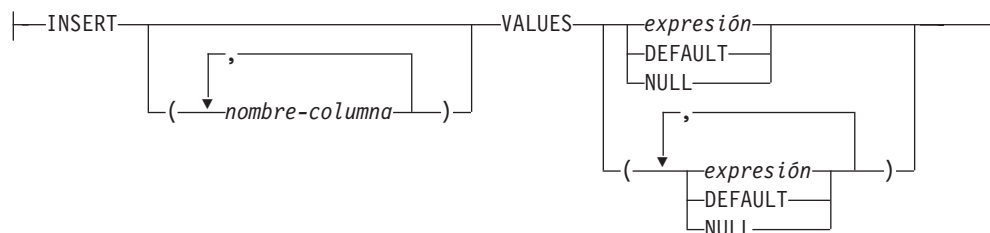
## cláusula-asignación:



## operación-update:





**operación-delete:****operación-insertar:****cláusula-período:****Descripción**

*nombre-tabla*, *nombre-vista* o (*selección completa*)

Identifica el destino de las operaciones de actualización, supresión o inserción de la fusión. El nombre debe identificar una tabla o vista que exista en el servidor actual, pero no debe identificar una tabla de catálogo, una tabla de consulta materializada mantenida por el sistema, una vista de una tabla de catálogos, una vista de sólo lectura, o una vista que directa o indirectamente contenga una cláusula WHERE que haga referencia a una subconsulta o una rutina definida como NOT DETERMINISTIC o EXTERNAL ACTION (SQLSTATE 42807).

Si el destino de la operación de fusión es una selección completa, la selección completa debe ser actualizable, suprimible o insertable, según lo definido en los elementos de Notas "Vistas actualizables", "Vistas suprimibles" o "Vistas insertables" de la descripción de la sentencia CREATE VIEW.

No se puede utilizar una *cláusula-período* en una operación-update o una operación-delete si el destino de la operación de fusión es una vista UNION ALL o una selección completa.

No puede utilizar un apodo (una referencia a una tabla federada remota) como tabla de destino.

**cláusula-correlación**

Se puede utilizar en *condición-búsqueda* o a la derecha de una *cláusula-asignación* para designar una tabla, vista o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte "referencia-tabla" en la descripción de "Subselección".

**USING referencia-tabla**

Especifica un conjunto de filas como una tabla de resultados para fusionarse en el destino. Si la tabla de resultados está vacía, se devuelve un aviso (SQLSTATE 02000).

**ON condición-búsqueda**

Especifica las filas de *referencia-tabla* que se utilizarán en la operación de

## MERGE

actualización y supresión de la fusión, y las filas que se utilizarán en la operación de inserción de la fusión. *condición-búsqueda* se aplica a cada fila de la tabla de destino y tabla de resultados de *referencia-tabla*. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* sea verdadero, se realiza la operación de actualización o supresión especificada. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* no sea verdadero, se realiza la operación de inserción especificada.

La *condición-búsqueda* tiene las restricciones siguientes (SQLSTATE 42972 a menos que se indique algo distinto):

- No puede contener ninguna subconsulta, escalar ni de cualquier otra clase
- No puede incluir ninguna operación de desreferencia ni la función Deref, donde el valor de referencia es distinto de la columna de identificadores de objeto
- No puede incluir una función SQL
- No puede incluir una expresión XMLQUERY ni XMLEXISTS
- Cualquier columna a la que se hace referencia en una expresión de la *condición-búsqueda* debe ser una columna de la tabla, vista o *referencia-tabla* de destino.
- Cualquier función a la que se haga referencia en una expresión de la *condición-uniión* de una unión externa completa debe ser determinista y no debe tener ninguna acción externa
- No puede incluir una función agregada (SQLSTATE 42903)

Si la *condición-búsqueda* es falsa o desconocida para cada fila de *referencia-tabla*, se devuelve un aviso (SQLSTATE 02000).

### WHEN *condición-coincidente*

Especifica la condición bajo la que se ejecuta la *operación-modificación* o la *sentencia-señal*. Cada *condición-coincidente* se evalúa en el orden de especificación. Las filas para las que la *condición-coincidente* se evalúa en verdadera no se consideran en las condiciones coincidentes subsiguientes.

### MATCHED

Indica la operación a realizar en las filas donde la condición de búsqueda ON es verdadera. Sólo UPDATE, DELETE o *sentencia-señal* se pueden especificar después de THEN.

#### AND *condición-búsqueda*

Especifica otra condición de búsqueda que se debe aplicar a las filas que han coincidido con la condición de búsqueda ON para la operación a realizar después de THEN.

### NOT MATCHED

Indica la operación a realizar en las filas donde la condición de búsqueda ON es falsa o desconocida. Sólo se pueden especificar INSERT o *sentencia-señal* después de THEN.

#### AND *condición-búsqueda*

Especifica otra condición de búsqueda que se debe aplicar a las filas que no han coincidido con la condición de búsqueda ON para la operación que se debe realizar después de THEN.

### THEN *operación-modificación*

Especifica la operación que se debe ejecutar cuando la *condición-coincidente* se evalúa como verdadera.

*operación-update*

Especifica la operación de actualización que se debe ejecutar para las filas donde la *condición-coincidente* se evalúa en verdadera.

**UPDATE**

Especifica la operación de actualización.

**cláusula-período**

Especifica que se aplica una cláusula de período a la operación de actualización en la sentencia MERGE. Para obtener más información sobre los efectos de la especificación de una cláusula de período en el contexto de una operación de actualización, consulte el tema que trata acerca de la sentencia UPDATE.

**SET**

Especifica la asignación de valores a nombres de columna.

*cláusula-asignación*

Especifica una lista de actualizaciones de columna.

*nombre-columna*

Identifica una columna que se debe actualizar. El *nombre-columna* debe identificar una columna de la tabla o vista especificada, pero no una columna de vista derivada de una función escalar, constante o expresión. No se debe especificar una columna más de una vez (SQLSTATE 42701).

Una columna de vista derivada de la misma columna como otra columna de la vista se puede actualizar, pero no se pueden actualizar ambas columnas en la misma sentencia MERGE (SQLSTATE 42701).

*expresión*

Indica el nuevo valor de la columna. La *expresión* no debe incluir una función agregada, excepto cuando se produce dentro de una selección completa escalar (SQLSTATE 42903).

Una *expresión* puede contener referencias a columnas del *nombre-tabla* o *nombre-vista*. 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.

Si la *expresión* es una referencia a una sola columna de la tabla fuente, el valor de columna de la tabla fuente puede que se haya especificado con un valor de variable de indicador ampliado. Los efectos de tales variables de indicador se aplican a las columnas de destino correspondientes de la *cláusula-assignment*.

Si la *expresión* es una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita, esta puede incluir una variable de indicador que esté habilitada para las variables de indicador ampliado.

Cuando las variables de indicador ampliado están habilitadas, no deben utilizarse los valores para la variable

## MERGE

de indicador ampliado por omisión (-5) o no asignado (-7) (SQLSTATE 22539) si cualquiera de las afirmaciones siguientes es cierta:

- La expresión es más compleja que una sola variable del lenguaje principal con conversiones explícitas.
- La columna de destino tiene un tipo de datos de tipo estructurado.

### DEFAULT

El valor por omisión asignado a la columna. Se puede especificar DEFAULT sólo para columnas que tengan un valor por omisión. Para obtener información acerca de los valores por omisión de tipos de datos, consulte la descripción de la cláusula DEFAULT en la sentencia "CREATE TABLE".

Se debe especificar DEFAULT para una columna que se ha definido como GENERATED ALWAYS. Se puede especificar un valor para una columna que se ha definido como GENERATED BY DEFAULT.

### NULL

Especifica un valor nulo como el nuevo valor de la columna. Especifique NULL sólo para columnas que pueden contener nulos (SQLSTATE 23502).

### *operación-delete*

Especifica la operación de suprimir que se debe ejecutar para las filas donde la *condición-coincidente* se evalúa en verdadera.

### DELETE

Especifica la operación de supresión.

### **cláusula-período**

Especifica que se aplica una cláusula de período a la operación de supresión en la sentencia MERGE. Para obtener más información sobre los efectos de la especificación de una cláusula de período en el contexto de una operación de supresión, consulte el tema que trata acerca de la sentencia DELETE.

### *operación-insertar*

Especifica la operación de insertar que se debe ejecutar para las filas donde la *condición-coincidente* evalúa en verdadera.

### INSERT

Presenta una lista de nombres de columna y expresiones de valores de fila que se deben utilizar para la operación de insertar.

El número de valores para la fila en la expresión de valor de fila debe ser igual al número de nombres en la lista de columnas a insertar. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

*(nombre-columna,...)*

Especifica las columnas para las que se proporcionan los valores de inserción. Cada nombre debe identificar una columna de la tabla o vista. No se debe identificar la misma columna más de una vez (SQLSTATE 42701). No se debe identificar una columna de vista que no puede aceptar valores de inserción. No se puede insertar un valor en una columna de vista que se derive de:

- Una constante, expresión o función escalar
- La misma columna de tabla base que otra columna de la vista.

Si el objeto de la operación es una vista con dichas columnas, se debe especificar una lista de nombres de columna y dicha lista no debe identificar estas columnas.

La omisión de la lista de columnas de una especificación implícita de una lista en la que cada columna de la tabla (que no está definida como implícitamente oculta) 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.

## VALUES

Especifica una o varias filas de valores que se deben insertar.

### *expresión*

Cualquier expresión que no incluye un nombre de columna (SQLSTATE 42703).

Si la *expresión* es una referencia a una sola columna de la tabla fuente, el valor de columna de la tabla fuente puede que se haya especificado con un valor de variable de indicador ampliado. Los efectos de tales variables de indicador se aplican a las columnas de destino correspondientes de la *operación-insertar*.

Si la *expresión* es una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita, esta puede incluir una variable de indicador (o, en el caso de una estructura de sistema principal, una matriz de indicador) que esté habilitada para las variables de indicador ampliado.

Cuando las variables de indicador ampliado están habilitadas, no deben utilizarse los valores para la variable de indicador ampliado por omisión (-5) o no asignado (-7) (SQLSTATE 22539) si cualquiera de las afirmaciones siguientes es cierta:

- La expresión es más compleja que una sola variable del lenguaje principal con conversiones explícitas.
- La columna de destino tiene un tipo de datos de tipo estructurado.

## DEFAULT

El valor por omisión asignado a la columna. Se puede especificar DEFAULT sólo para columnas que tengan un valor por omisión. Para obtener información acerca de los valores por omisión de tipos de datos, consulte la descripción de la cláusula DEFAULT en la sentencia "CREATE TABLE".

Se debe especificar DEFAULT para una columna que se ha definido como GENERATED ALWAYS. Se puede especificar un valor para una columna que se ha definido como GENERATED BY DEFAULT.

## NULL

Especifica el valor nulo como el valor de la columna. Especifique NULL sólo para columnas que pueden contener nulos (SQLSTATE 23502).

## MERGE

### *sentencia-señal*

Especifica la sentencia SIGNAL que se ejecutará para devolver un error cuando la *condición-coincidente* se evalúa en verdadera.

### **ELSE IGNORE**

Especifica que no se realizará ninguna acción para las filas donde ninguna *condición-coincidente* se evalúa en verdadera. Si todas las filas de *referencia-tabla* se pasan por alto, se devuelve un aviso (SQLSTATE 02000).

### **WITH**

Especifica el nivel de aislamiento en el que se ejecuta la sentencia MERGE.

**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

- Se puede especificar más de una *operación-modificación* (UPDATE SET, DELETE o *operación-insertar*), o *sentencia-señal* en una única sentencia MERGE.
- Sólo se puede trabajar en cada fila del destino una vez. Sólo se puede identificar una fila del destino como MATCHED con una fila en la tabla de resultados de la *referencia-tabla* (SQLSTATE 21506). Una operación de SQL anidado (RI o desencadenante excepto el desencadenante INSTEAD OF) no puede especificar la tabla de destino (o una tabla en la misma jerarquía de tablas) como un destino de una sentencia UPDATE, DELETE, INSERT o MERGE (SQLSTATE 27000).
- **Política de seguridad** si se protege la tabla de destino identificada o la tabla base de la vista de destino identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten los siguientes tipos de acceso.
  - Para la operación de actualización:
    - Acceso de grabación a todas las columnas protegidas que se están actualizando (SQLSTATE 42512)
    - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)
    - Acceso de grabación y de lectura a todas las filas que se están actualizando (SQLSTATE 42519)

El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:

- La columna DB2SECURITYLABEL no está incluida en la lista de columnas que se van a actualizar (y, por lo tanto, se actualizará implícitamente en la etiqueta de seguridad para el acceso de grabación del ID de autorización de la sesión)
- Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene

acceso de grabación para dicho valor y la política de seguridad se crea con la opción `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`

- Para la operación de supresión:
  - Acceso de grabación a todas las columnas protegidas (SQLSTATE 42512)
  - Acceso de grabación y de lectura a todas las filas que se han seleccionado para la supresión (SQLSTATE 42519)
- Para la operación de inserción:
  - Acceso de grabación a todas las columnas protegidas para las que se ha proporcionado explícitamente un valor de datos (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna `DB2SECURITYLABEL` para las políticas de seguridad que se han creado con la opción `RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL` (SQLSTATE 23523)

El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna `DB2SECURITYLABEL` (SQLSTATE 23523), lo cual puede suceder cuando:

- No se proporciona explícitamente un valor para la columna `DB2SECURITYLABEL`
  - Se proporciona explícitamente un valor para la columna `DB2SECURITYLABEL` pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción `OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL`
- **Desencadenantes *INSTEAD OF*:** Si se especifica una vista como destino de la sentencia `MERGE`, no se debe definir ningún desencadenante `INSTEAD OF` para la vista, o bien se debe definir un desencadenante `INSTEAD OF` para cada una de las operaciones de actualizar, suprimir e insertar (SQLSTATE 428FZ).
  - **Uso de variable de indicador ampliado:** Si se habilita, los valores de variable de indicador negativos fuera del rango entre -1 y -7 no pueden ser una entrada (SQLSTATE 22010). Además, si se habilitan, los valores `DEFAULT` y `UNASSIGNED` de variable de indicador ampliado no deben aparecer en contextos en los que no estén soportados (SQLSTATE 22539).
  - **Variables de indicador ampliado en la *cláusula-assignment*:** Una *expresión* que sea una referencia a una sola columna de la tabla fuente, una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita puede dar lugar a la asignación de un valor basado en una variable de indicador ampliado. La asignación del valor `UNASSIGNED` basado en una variable de indicador ampliado tiene como efecto dejar que la columna de destino se establezca en su valor por omisión, como si no se hubiera especificado en la sentencia. La asignación del valor `DEFAULT` basado en una variable de indicador ampliado asigna el valor por omisión de la columna. Para obtener información sobre los valores por omisión de los tipos de datos, consulte la descripción de la cláusula `DEFAULT` en “`CREATE TABLE`” en la página 722. Si una columna de destino no es actualizable (por ejemplo, una columna en una vista que esté definida como una expresión), se le debe asignar el valor `UNASSIGNED` basado en una variable de indicador ampliado (SQLSTATE 42808). Si la columna de destino es una columna definida como `GENERATED ALWAYS`, se le debe asignar la palabra clave `DEFAULT` o los valores `DEFAULT` o `UNASSIGNED` basados en una variable de indicador ampliado (SQLSTATE 428C9).

## MERGE

La *cláusula-assignment* no debe asignar a todas las columnas de destino un valor UNASSIGNED basado en una variable de indicador ampliado (SQLSTATE 22540).

- **Variables de indicador ampliado en la *cláusula-insert*:** Una *expresión* que sea una referencia a una sola columna de la tabla fuente, una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita puede dar lugar a la inserción de un valor basado en una variable de indicador ampliado. En una *operación-insert*, un valor UNASSIGNED tiene como efecto establecer una columna en su valor por omisión.

Si una columna de destino no es actualizable, se le debe asignar el valor UNASSIGNED basado en una variable de indicador ampliado (SQLSTATE 42808), a menos que sea una columna definida como GENERATED ALWAYS. Si la columna de destino es una columna definida como GENERATED ALWAYS, se le debe asignar la palabra clave DEFAULT o los valores DEFAULT o UNASSIGNED basados en una variable de indicador ampliado (SQLSTATE 428C9).

Para otras normas que afectan a la parte de la operación de actualizar, insertar o suprimir de la sentencia MERGE, consulte la sección "Normas" de la descripción de la sentencia correspondiente.

### Notas

- **Orden de proceso:**
  1. Determine el conjunto de filas que se deben procesar de la fuente y destino. Si se utiliza CURRENT\_TIMESTAMP en esta sentencia, sólo se realiza una lectura de reloj para toda la sentencia.
  2. Utilice la cláusula ON para clasificar estas filas en MATCHED o NOT MATCHED.
  3. Evalúe cualquier *condición-coincidente* en las cláusulas WHEN.
  4. Evalúe cualquier *expresión* en cualquier *cláusula-asignación* y *operación-insertar*.
  5. Ejecute cada *sentencia-señal*.
  6. Aplique cada *operación-modificación* a las filas aplicables en el orden especificado. Las restricciones y desencadenantes activados por cada *operación-modificación* se ejecutan para la *operación-modificación*. Los desencadenantes a nivel de sentencia se activan incluso si ninguna fila satisface la *operación-modificación*. Cada *operación-modificación* puede afectar a los desencadenantes y restricciones referenciales de cada *operación-modificación* subsiguiente.
- **Atomicidad a nivel de sentencia:** Si se produce un error durante la ejecución de la sentencia MERGE, se retrotrae toda la sentencia.
- **Número de filas actualizadas:** Cuando se completa la ejecución de una sentencia MERGE, el valor del elemento ROW\_COUNT para GET DIAGNOSTICS y SQLERRD(3) en SQLCA es el número de filas en las que actúa la sentencia MERGE, excluyendo las filas identificadas por la cláusula ELSE IGNORE. El valor de SQLERRD(3) no incluye el número de filas sobre las que se ha actuado como resultado de restricciones o activadores. El valor de SQLERRD(5) incluye el número de estas filas.
- **La fila insertada tampoco se puede actualizar:** No se hace ningún intento de actualizar una fila en el destino que aún no existía antes de ejecutarse la sentencia MERGE; es decir, no hay actualizaciones de filas que haya insertado la sentencia MERGE.
- **Variables de indicador ampliado y activadores de actualización:** Si a una columna de destino se le ha asignado un valor UNASSIGNED basado en una



variable de indicador ampliado, la columna no se considera actualizada. Esa columna se trata como si no se hubiera especificado en la lista OF *nombre-columna* de cualquier activador de actualización definido en la tabla de destino.

- **Variables de indicador ampliado y activadores de inserción:** El uso de variables de indicador ampliado no provoca ningún cambio en la activación de los activadores de inserción. Si todas las columnas de la lista de columnas implícitas o explícitas se han asignado a un valor DEFAULT o UNASSIGNED basado en una variable de indicador ampliado, se intenta realizar una inserción donde todas las columnas tienen sus respectivos valores por omisión. Si la inserción resulta satisfactoria, se activa el activador de inserción.
- **Variables de indicador ampliado y comprobaciones de errores diferidas:** Cuando se habilitan variables de indicador ampliado, la validación que de otro modo se realizaría en la preparación de sentencias para reconocer la inserción o actualización de una columna no actualizable se difiere hasta la ejecución de sentencias. La necesidad de notificar un error se determina únicamente durante la ejecución.
- **Consideraciones sobre las tablas temporales de período de sistema:** cuando se procesa MERGE para una tabla temporal de período de sistema, ello afecta a las filas del mismo modo que si se hubieran invocado operaciones de cambio de datos específicas. Consulte los temas sobre las sentencias UPDATE, DELETE e INSERT para obtener más información.
- **Consideraciones sobre tablas temporales de período de aplicación y activadores:** Cuando se suprime una fila y se especifica la cláusula FOR PORTION OF BUSINESS\_TIME, se pueden insertar implícitamente filas adicionales para reflejar cualquier porción de la fila que no haya sido suprimida. Los activadores de supresión existentes se activan para las filas suprimidas y los activadores de inserción existentes se activan para las filas que se han insertado implícitamente. Cuando se actualiza una fila y se especifica la cláusula FOR PORTION OF BUSINESS\_TIME, se pueden insertar implícitamente filas adicionales para reflejar cualquier porción de la fila que no haya sido actualizada. Los activadores de actualización existentes se activan para las filas actualizadas y los activadores de inserción existentes se activan para las filas que se han insertado implícitamente.
- **Consideraciones sobre una sentencia MERGE sin una lista de columnas en la operación de inserción:** Una sentencia MERGE sin la especificación de una lista de columna como parte de la operación de inserción no incluye columnas ocultas implícitamente. Las columnas que se han definido como ocultas implícitamente y que no son nulas deben tener definido un valor por omisión.

## Ejemplos

- *Ejemplo 1:* Para actividades cuya descripción ha cambiado, actualice la descripción en la tabla de archivo. Para actividades nuevas, inserte en la tabla de archivo. Tanto la tabla de archivo como la tabla de actividades tienen actividad como clave primaria.

```

MERGE INTO archive ar
USING (SELECT activity, description FROM activities) ac
ON (ar.activity = ac.activity)
WHEN MATCHED THEN
 UPDATE SET
 description = ac.description
WHEN NOT MATCHED THEN
 INSERT
 (activity, description)
 VALUES (ac.activity, ac.description)

```

## MERGE

- *Ejemplo 2:* Utilizando la tabla de envíos, fusione las filas en la tabla de inventario, aumentando la cantidad por número de piezas en la tabla de envíos para filas que coincidan; si no inserte el nuevo *partno* en la tabla de inventario.

```
MERGE INTO inventario AS in
USING (SELECT partno, description, count FROM shipment
 WHERE shipment.partno IS NOT NULL) AS sh
ON (in.partno = sh.partno)
WHEN MATCHED THEN
 UPDATE SET
 description = sh.description,
 quantity = in.quantity + sh.count
WHEN NOT MATCHED THEN
 INSERT
 (partno, description, quantity)
 VALUES (sh.partno, sh.description, sh.count)
```

- *Ejemplo 3:* Utilizando la tabla de transacciones, fusione las filas en una tabla de contabilidad, actualizando el saldo del conjunto de transacciones contra un ID de cuenta e insertando nuevas cuentas desde las transacciones consolidadas donde ya no existen.

```
MERGE INTO account AS a
USING (SELECT id, sum(amount) sum_amount FROM transaction
 GROUP BY id) AS t
ON a.id = t.id
WHEN MATCHED THEN
 UPDATE SET
 balance = a.balance + t.sum_amount
WHEN NOT MATCHED THEN
 INSERT
 (id, balance)
 VALUES (t.id, t.sum_amount)
```

- *Ejemplo 4:* Utilizando la tabla *transaction\_log*, fusione las filas en la tabla *employee\_file*, actualizando el teléfono y la oficina con la fila *transaction\_log* más reciente en función de la hora de la transacción, e insertando la fila *new employee\_file* más reciente donde la fila no existe aún.

```
MERGE INTO employee_file AS e
USING (SELECT empid, phone, office
 FROM (SELECT empid, phone, office,
 ROW_NUMBER() OVER (PARTITION BY empid
 ORDER BY transaction_time DESC) rn
 FROM transaction_log) AS nt
 WHERE rn = 1) AS t
ON e.empid = t.empid
WHEN MATCHED THEN
 UPDATE SET
 (phone, office) =
 (t.phone, t.office)
WHEN NOT MATCHED THEN
 INSERT
 (empid, phone, office)
 VALUES (t.empid, t.phone, t.office)
```

- *Ejemplo 5:* Utilizando los valores proporcionados dinámicamente para una fila de empleado, actualice la tabla maestra de empleado si los datos corresponden a un empleado existente, o inserte la fila si los datos son para un empleado nuevo. El ejemplo siguiente es un fragmento de código de un programa en C.

```
hvl =
"MERGE INTO employee AS t
USING TABLE(VALUES(CAST (? AS CHAR(6)), CAST (? AS VARCHAR(12)),
 CAST (? AS CHAR(1)), CAST (? AS VARCHAR(15)),
 CAST (? AS SMALLINT), CAST (? AS INTEGER)))
 s(empno, firstnme, midinit, lastname, edlevel, salary)
ON t.empno = s.empno
WHEN MATCHED THEN
```

```

UPDATE SET
 salary = s.salary
WHEN NOT MATCHED THEN
 INSERT
 (empno, firstnme, midinit, lastname, edlevel, salary)
 VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
 s.salary)";
EXEC SQL PREPARE s1 FROM :hv1;
EXEC SQL EXECUTE s1 USING '000420', 'SERGE', 'K', 'FIELDING', 18, 39580;

```

- *Ejemplo 6:* Actualice la lista de actividades organizada por el Grupo A en la tabla de archivos. Suprima todas las actividades caducadas y actualice la información de las actividades (descripción y fecha) en la tabla de archivo si estas han cambiado. Para nuevas actividades previstas, inserte en el archivo. Señalice un error si se desconoce la fecha de la actividad. Se debe especificar la fecha de las actividades en la tabla de archivo. Cada grupo tiene una tabla de actividades. Por ejemplo, `activities_groupA` contiene todas las actividades que organizan, y la tabla de archivo contiene todas las actividades previstas organizadas por grupos diferentes en una empresa. La tabla de archivo (grupo, actividad) como clave primaria, y la fecha no puede contener nulos. Todas las tablas de actividades tienen actividad como clave primaria. La columna `last_modified` en el archivo está definida con `CURRENT_TIMESTAMP` como valor por omisión.

```

MERGE INTO archive ar
 USING (SELECT activity, description, date, last_modified
 FROM activities_groupA) ac
 ON (ar.activity = ac.activity) AND ar.group = 'A'
 WHEN MATCHED AND ac.date IS NULL THEN
 SIGNAL SQLSTATE '70001'
 SET MESSAGE_TEXT =
 ac.activity CONCAT ' no se puede modificar. Razón: No se conoce la fecha'
 WHEN MATCHED AND ac.date < CURRENT_DATE THEN
 DELETE
 WHEN MATCHED AND ar.last_modified < ac.last_modified THEN
 UPDATE SET
 (description, date, last_modified) = (ac.description, ac.date, DEFAULT)
 WHEN NOT MATCHED AND ac.date IS NULL THEN
 SIGNAL SQLSTATE '70002'
 SET MESSAGE_TEXT =
 ac.activity CONCAT ' no se puede insertar. Razón: No se conoce la fecha'
 WHEN NOT MATCHED AND ac.date >= CURRENT_DATE THEN
 INSERT
 (group, activity, description, date)
 VALUES ('A', ac.activity, ac.description, ac.date)
 ELSE IGNORE

```

## OPEN

La sentencia OPEN abre un cursor para que pueda utilizarse para leer filas de la tabla de resultados.

### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente. Cuando se invoca utilizando el procesador de línea de mandatos, no se pueden especificar algunas opciones. Para obtener más información, consulte la sección sobre utilización de sentencias de SQL y XQuery de línea de mandatos.

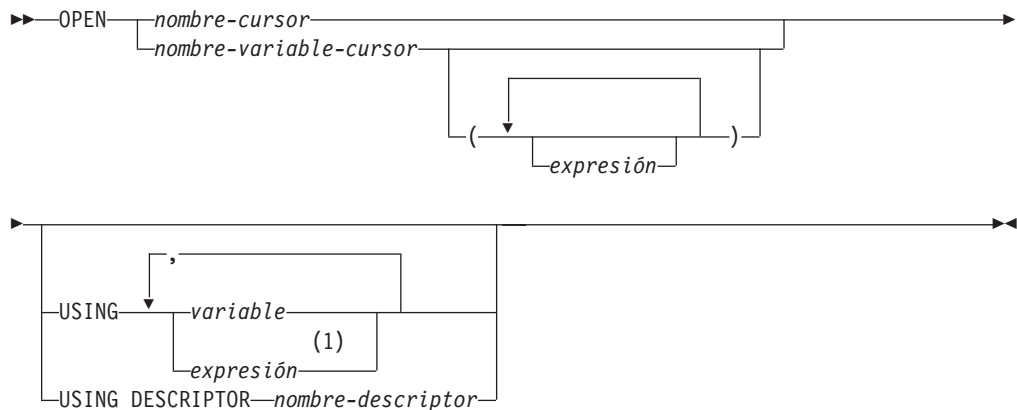
### Autorización

Si se hace referencia a una variable global, los privilegios del ID de autorización de la sentencia deben incluir unas de las autorizaciones siguientes:

- el privilegio READ sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

Los privilegios de grupo no se tienen en cuenta porque esta sentencia no puede prepararse de forma dinámica.

### Sintaxis



#### Notas:

- 1 Una expresión que no sea una variable sólo puede utilizarse en sentencias compuestas compiladas.

### Descripción

#### *nombre-cursor*

Identifica un cursor que se define en una sentencia DECLARE CURSOR que se ha expresado antes en el programa. Si el *nombre-cursor* identifica un cursor de un procedimiento de SQL declarado como WITH RETURN TO CLIENT que ya se encuentra en el estado abierto, el cursor abierto existente se convierte en un cursor de conjunto de resultados que ya no es accesible mediante el *nombre-cursor* y se abre un cursor nuevo que pasa a ser accesible mediante el

*nombre-cursor*. De lo contrario, cuando se ejecuta la sentencia OPEN, el cursor identificado por *nombre-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 de resultados del cursor se obtiene evaluándose la sentencia SELECT. La evaluación utiliza los valores actuales de cualquier registro especial, variables globales o expresiones PREVIOUS VALUE que se han especificado en la sentencia SELECT y los valores actuales de cualquier variable del lenguaje principal que se han especificado en la sentencia SELECT o en la cláusula USING de la sentencia OPEN. Las filas de la tabla de resultados pueden obtenerse durante la ejecución de la sentencia OPEN, y puede crearse una tabla temporal para contenerlas; o bien pueden obtenerse 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 de resultados. Si la tabla está vacía, el estado del cursor está, efectivamente, "después de la última fila".

#### *nombre-variable-cursor*

Asigna un nombre a una variable de cursor. El valor de la variable de cursor no puede ser nulo (SQLSTATE 34000). Una variable de cursor a la que se asigna directa o indirectamente un constructor de valor de cursor sólo puede utilizarse en una sentencia OPEN que se encuentre en el mismo ámbito que la asignación (SQLSTATE 51044). Si el constructor de valor del cursor asignado a la variable de cursor ha especificado un *nombre-sentencia*, la sentencia OPEN debe estar en el mismo ámbito en el que se declaró explícita o implícitamente ese *nombre-sentencia* (SQLSTATE 51044).

Cuando se ejecuta la sentencia OPEN, el cursor subyacente de la variable de cursor debe estar en estado cerrado. La tabla de resultados del cursor subyacente se obtiene evaluándose la sentencia SELECT o la sentencia dinámica asociada a la variable de cursor. La evaluación utiliza los valores actuales de cualquier registro especial, variables globales o expresiones PREVIOUS VALUE que se han especificado en la sentencia SELECT y los valores actuales de cualquier variable que se han especificado en la sentencia SELECT o en la cláusula USING de la sentencia OPEN. Las filas de la tabla de resultados pueden obtenerse durante la ejecución de la sentencia OPEN, y puede crearse una tabla temporal para contenerlas; o bien pueden obtenerse 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 de resultados. Si la tabla está vacía, el estado del cursor está, efectivamente, "después de la última fila".

Una sentencia OPEN que utiliza un *nombre-variable-cursor* sólo se puede utilizar en una sentencia de SQL compuesto (compilado).

#### ( *expresión*, ... )

Especifica los argumentos asociados a los parámetros con nombre de una variable de cursor parametrizada. El *constructor-valor-cursor* asignado a la variable de cursor debe incluir una lista de parámetros que tenga el mismo número de parámetros que el número de argumentos especificado (SQLSTATE 07006 o 07004). El tipo de datos y el valor de la expresión número *n* debe poder asignarse al parámetro número *n* (SQLSTATE 07006 o 22018).

**USING**

Presenta los valores que se sustituyen en los marcadores de parámetro o en las variables de la sentencia del cursor. Para obtener una explicación de los marcadores de parámetro, consulte "PREPARE".

Si se especifica un *nombre-sentencia* en la sentencia DECLARE CURSOR o en el constructor de valor de cursor asociado a la variable de cursor que incluye marcadores de parámetro, deberá utilizarse USING. Si la sentencia preparada no incluye ningún marcador de parámetro, se ignora USING.

Si se especifica una *sentencia-select* en la sentencia DECLARE CURSOR o en el constructor de valor de cursor no parametrizado asociado a la variable de cursor, puede utilizarse USING para alterar temporalmente los valores de variable.

**variable**

Identifica una variable o una estructura del sistema principal declarada en el programa según las normas para declarar variables y variables del lenguaje principal. El número de variables debe ser igual al número de marcadores de parámetro de la sentencia preparada. La variable *n* corresponde al marcador de parámetro *n* de la sentencia preparada. Las variables de localizador y las variables de referencia a archivo, cuando procede, pueden proporcionarse como fuente de los valores para los marcadores de parámetro.

**expresión**

Especifica valores para asociarlos a marcadores de parámetro utilizando expresiones. Una sentencia OPEN que especifique expresiones en la cláusula USING sólo se puede utilizar en una sentencia de SQL compuesto (compilado) (SQLSTATE 42601). El número de expresiones debe ser el mismo que el número de marcadores de parámetro de la sentencia preparada (SQLSTATE 07001). La expresión número *n* corresponde al marcador de parámetro número *n* de la sentencia preparada. El tipo de datos y el valor de la expresión número *n* debe poder asignarse al tipo asociado al marcador de parámetro número *n* (SQLSTATE 07006).

**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 parámetro con tipo, los atributos de la variable de destino son aquellos determinados por la especificación CAST. Para un marcador de parámetro sin tipo, los atributos de la variable de destino se determinan de acuerdo con el contexto del marcador de parámetro.
- 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 (incluidos los blancos finales para series que no son series largas) 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 indicada para una sentencia SELECT preparada que contenga marcadores de parámetro. Sin embargo, también puede utilizarse cuando la sentencia SELECT del cursor forma parte de la sentencia DECLARE CURSOR o del constructor de valor de cursor no parametrizado asociado a la variable de cursor. En este caso la sentencia OPEN se ejecuta como si cada variable del lenguaje principal de la sentencia SELECT fuese un marcador de parámetro, 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. No debe utilizarse una alteración temporal del valor de variable cuando se abre una variable de cursor parametrizado, porque la sentencia SELECT no incluirá ninguna otra variable.
- Las sentencias y rutinas de cambio de datos de SQL que modifican datos SQL incorporados en la definición del cursor se ejecutan por completo y el conjunto de resultados se almacena en una tabla temporal cuando se abre el cursor. Si la ejecución de la sentencia resulta satisfactoria, el campo SQLERRD(3) contiene la suma del número de filas que están cualificadas para operaciones de inserción, actualización y supresión. Si se produce un error durante la ejecución de una sentencia OPEN en la que interviene un cursor que contiene una sentencia de cambio de datos dentro de una selección completa, los resultados de dicha sentencia de cambio de datos se retrotraen.

La retrotracción explícita de una sentencia OPEN, o la retrotracción a un punto de salvaguarda antes de una sentencia OPEN, cierra el cursor. Si la definición del cursor contiene una sentencia de cambio de datos dentro de la cláusula FROM de una selección completa, los resultados de la sentencia de cambio de datos se retrotraen.

Los cambios en filas de una tabla que es el destino de una sentencia de cambio de datos anidada dentro de una sentencia SELECT o de una sentencia SELECT INTO se procesan cuando se abre el cursor y no se deshacen si se produce un error durante una operación de recuperación contra el cursor.

## Notas

- **Estado cerrado de los cursores:** Todos los cursores de un programa están en estado cerrado cuando se inicia el programa y cuando este 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.

El cursor subyacente de una variable de cursor está cerrado si la variable de cursor sale del ámbito y no hay más variables de cursor que hagan referencia al cursor subyacente.

- Para recuperar filas de la tabla de resultados 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 las tablas de resultados materializados:** En algunos casos, como cuando el cursor es de sólo lectura, las filas de resultados de un cursor se

## OPEN

obtienen durante la ejecución de las sentencias FETCH. En otros casos, en lugar de ello se utiliza el método de tabla de resultados materializados. Con el método de tabla de resultados materializados, la tabla de resultados en su totalidad se transfiere a un almacenamiento intermedio temporal durante la ejecución de la sentencia OPEN. Cuando se utiliza un almacenamiento intermedio temporal, los resultados de un programa pueden diferir, tal como se indica a continuación:

- 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 de resultados.
- Las expresiones NEXT VALUE de la sentencia SELECT se evalúan para cada fila de la tabla de resultados durante la ejecución de OPEN.

Y, a la inversa, si no se utiliza un almacenamiento intermedio temporal, las sentencias INSERT, UPDATE y DELETE que se han ejecutado mientras el cursor estaba abierto pueden afectar a la tabla de resultados si se han emitido desde la misma unidad de trabajo y las expresiones NEXT VALUE de la sentencia SELECT se evalúan a medida que se capta cada fila. Esta tabla de resultados también puede verse afectada por las operaciones que ha ejecutado la misma unidad de trabajo y el efecto de tales operaciones no siempre es previsible. Por ejemplo, si el cursor C está posicionado en una fila de su tabla de resultados definida como SELECT \* FROM T y se inserta una nueva fila en T, el efecto de dicha inserción en la tabla de resultados 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 almacenamiento de sentencias en antememoria afecta a los cursores declarados abiertos por la sentencia OPEN.
- *Apertura del mismo cursor varias veces:* Un cursor de un procedimiento de SQL declarado como WITH RETURN TO CLIENT puede abrirse incluso cuando ya haya un cursor del mismo nombre en el estado abierto. En tal caso, el cursor abierto existente se convierte en un cursor de conjunto de resultados y deja de ser accesible por su nombre de cursor. Se abre un cursor nuevo y pasa a ser accesible por el nombre de cursor. El cierre del nuevo cursor no hace que el cursor que anteriormente era accesible por ese nombre sea accesible de nuevo por el nombre de cursor. Los cursores que se convierten en cursores de conjunto de resultados de esta forma no son accesibles en el servidor y pueden procesarse solamente en el cliente.

## 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
 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 parámetro en el predicado de la sentencia-select, se proporcionan dos referencias a variables del lenguaje principal con la sentencia OPEN para pasar valores de entero 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 este ejemplo.)

```
EXEC SQL BEGIN DECLARE SECTION;
 static short hv_int;
 char hv_vchar64[65];
 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 parámetro 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;
```

*Ejemplo 4:* Cree un procedimiento que realice las operaciones siguientes:

1. Asigna un cursor a la variable de cursor de salida
2. Abre el cursor

```
CREATE PROCEDURE PROC1 (OUT P1 CURSOR)LANGUAGE SQL
BEGIN
SET P1 = CURSOR FOR SELECT DEPTNO, DEPTNAME, MGRNO FROM DEPARTMENT WHERE ADMRDEPT = 'A00'; --
OPEN P1; --
END;
```

## PIPE

La sentencia PIPE se utiliza para devolver una fila de una función de tabla compilada.

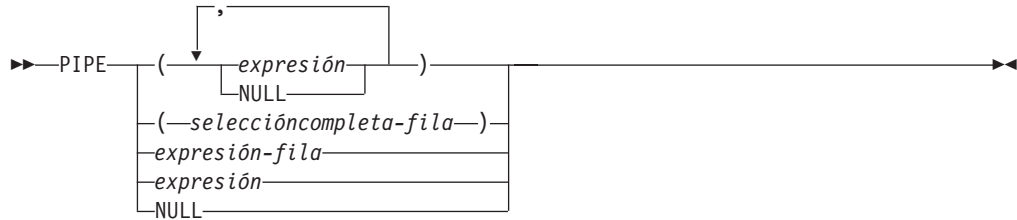
### Invocación

Esta sentencia puede incorporarse en una sentencia de SQL compuesto (compilado) de una función de tabla de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se requiere ningún privilegio para invocar la sentencia PIPE. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar cualquier expresión que se incorpore en la sentencia PIPE.

### Sintaxis



### Descripción

( *expresión*, ... )

Especifica que se devuelve un valor de fila de la función. El número de expresiones (o palabras clave NULL) de la lista debe coincidir con el tipo de datos RETURNS de la función, y el valor de cada expresión debe poder asignarse a la columna o campo correspondiente en el tipo de datos RETURNS de la función.

*selección completa-fila*

Especifica una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de columnas o campos en el tipo de datos RETURNS de la función. El valor de cada columna de la fila que la selección completa devuelve debe poder asignarse a la columna o campo correspondiente en el tipo de datos RETURNS de la función. Si el resultado de la selección completa de fila es ninguna fila, se devuelven valores nulos.

*expresión-fila*

Especifica que se devuelve el valor de fila de la función. El número de campos de la fila debe coincidir con el tipo de datos RETURNS de la función, y cada campo de la fila debe poderse asignar al campo correspondiente en el tipo de datos RETURNS de la función. Si la *expresión-fila* y el tipo de datos RETURNS son tipos de datos definidos por el usuario, los nombres de los tipos deben ser iguales (SQLSTATE 42821).

*expresión*

Especifica que se devuelve un valor escalar de la función. El tipo de datos RETURNS de la función de tabla debe tener una sola columna, y el valor de la expresión debe poder asignarse a esa columna.

**NULL**

Especifica que se devuelve un valor nulo de la función. Se devuelve un valor nulo para cada campo de columna o fila.

**Notas**

- **Procedimientos declarados localmente** La sentencia PIPE no puede utilizarse dentro de un procedimiento que se ha declarado localmente en la sentencia de SQL compuesto (compilado) de una función de tabla de SQL.
- **Términos similares:** A veces, a una función de tabla de SQL que utiliza una sentencia PIPE se hace referencia como función *segmentada*.

## 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 incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

### Autorización

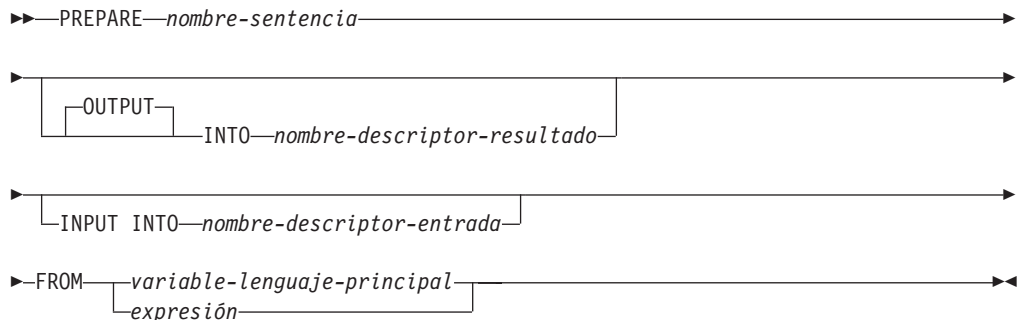
Para las sentencias en las que el control de autorizaciones se realiza al preparar la sentencia (DML), los privilegios del ID de autorización de la sentencia deben incluir los necesarios para ejecutar la sentencia de SQL especificada por la sentencia PREPARE. El ID de autorización de la sentencia podría verse afectado por la opción de enlace DYNAMICRULES.

En el caso de las sentencias en las que el control de autorizaciones se realiza durante la ejecución de la sentencia (sentencias DDL, GRANT y REVOKE), no es necesaria ninguna autorización para utilizar esta sentencia; no obstante, se comprueba la autorización cuando se ejecuta la sentencia preparada.

Para las sentencias que afectan a las tablas protegidas con una política de seguridad, las normas asociadas con la política de seguridad siempre se evaluarán durante el tiempo de ejecución de la sentencia.

Si el ID de autorización de las sentencias contiene una autorización EXPLAIN, SQLADM o DBADM, el usuario puede preparar cualquier sentencia; no obstante, la capacidad para ejecutar la sentencia se vuelve a comprobar en el momento de la ejecución de la sentencia.

### Sintaxis



### 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.

**OUTPUT INTO**

Si se utiliza OUTPUT INTO y la sentencia PREPARE se ejecuta satisfactoriamente, se colocará información acerca de los marcadores de parámetro de salida de la sentencia preparada en la SQLDA que *nombre-descriptor-resultado* especifica.

*nombre-descriptor-resultado*

Especifica el nombre de una SQLDA. (Como alternativa a esta cláusula, puede utilizarse la sentencia DESCRIBE.)

**INPUT INTO**

Si se utiliza INPUT INTO y la sentencia PREPARE se ejecuta satisfactoriamente, se colocará información acerca de los marcadores de parámetro de entrada de la sentencia preparada en la SQLDA que *nombres-descriptor-entrada* especifica. Los marcadores de parámetro de entrada se consideran siempre como con posibilidad de nulos, independientemente de su uso.

*nombre-descriptor-entrada*

Especifica el nombre de una SQLDA. (Como alternativa a esta cláusula, puede utilizarse la sentencia DESCRIBE.)

**FROM**

Especifica la serie de la sentencia. La serie de la sentencia es el valor de la variable del lenguaje principal especificada.

*variable-lenguaje-principal*

Especifica una variable del lenguaje principal que se ha descrito en el programa de acuerdo con las normas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres de longitud que puede variar cuyo tamaño sea menor que el tamaño de sentencia máximo de 2.097.152 bytes. Tenga en cuenta que un CLOB(2097152) puede contener una sentencia de tamaño máximo, pero un VARCHAR no puede.

*expresión*

Expresión que especifica la serie de la sentencia. La expresión debe devolver un tipo de serie de caracteres de longitud variable o fija que sea inferior al tamaño de sentencia máximo de 2.097.152 bytes.

**Normas**

- **Normas para las sentencias:** La sentencia debe ser una sentencia ejecutable que se pueda preparar dinámicamente. Debe ser una de las sentencias de SQL siguientes:
  - ALTER
  - CALL
  - COMMENT
  - COMMIT
  - SQL compuesto (compilado)
  - SQL compuesto (en línea)
  - CREATE
  - DECLARE GLOBAL TEMPORARY TABLE
  - DELETE
  - DROP
  - EXPLAIN
  - FLUSH EVENT MONITOR

## PREPARE

- FLUSH PACKAGE CACHE
- GRANT
- INSERT
- LOCK TABLE
- MERGE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME
- REVOKE
- ROLLBACK
- SAVEPOINT
- sentencia-select
- SET COMPILATION ENVIRONMENT
- SET CURRENT DECFLOAT ROUNDING MODE
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT FEDERATED ASYNCHRONY
- SET CURRENT IMPLICIT XMLPARSE OPTION
- SET CURRENT ISOLATION
- SET CURRENT LOCALE LC\_MESSAGES
- SET CURRENT LOCALE LC\_TIME
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT MDC ROLLOUT MODE
- SET CURRENT OPTIMIZATION PROFILE
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET CURRENT TEMPORAL BUSINESS\_TIME
- SET CURRENT TEMPORAL SYSTEM\_TIME
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET ROLE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- SET SCHEMA
- SET SERVER OPTION
- SET SESSION AUTHORIZATION
- SET SQL\_CCFLAGS
- SET USAGE LIST STATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)

- SET variable
- TRANSFER OWNERSHIP (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- TRUNCATE (solamente si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete)
- UPDATE

## Notas

- **Marcadores de parámetro:** Aunque una serie de sentencia no puede incluir referencias a variables de lenguaje principal, puede incluir *marcadores de parámetro*. Estos pueden sustituirse por los valores de las variables del lenguaje principal cuando se ejecuta la sentencia preparada. En el caso de una sentencia CALL, también puede utilizarse un marcador de parámetro para los argumentos OUT e INOUT para el procedimiento. Tras haberse ejecutado CALL, el valor que se ha devuelto para el argumento se asignará a la variable del lenguaje principal que corresponde al marcador de parámetro.

Un marcador de parámetros es un signo de interrogación (?) o un signo de dos puntos seguido de un nombre (:nombre) que se utiliza donde se pueda utilizar una variable del lenguaje principal si la serie de la sentencia fuera una sentencia de SQL estático. Para obtener una explicación sobre cómo se sustituyen los marcadores de parámetro por valores, consulte "OPEN" y "EXECUTE".

Si se asigna un nombre al marcador de parámetro, el nombre puede incluir letras, números y los símbolos @, #, \$ y \_. El nombre no se convierte a mayúsculas.

Los marcadores de parámetro con nombre tienen la misma sintaxis que las variables del lenguaje principal, aunque estos dos no son intercambiables. Una variable del lenguaje principal tiene un valor y se utiliza directamente en una sentencia de SQL estático. Un marcador de parámetro con nombre es un espacio reservado para un valor en una sentencia de SQL dinámico y el valor se suministra al ejecutarse la sentencia.

Existen dos tipos de marcadores de parámetro:

### **Marcador de parámetro 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-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 en el 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 parámetro sin tipo**

Es un marcador de parámetro que se especifica sin su tipo de datos de destino. Consta de un signo de interrogación individual. El tipo de datos de un marcador de parámetro sin tipo lo proporciona el contexto. Por ejemplo,

## PREPARE

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 parámetro con tipo pueden utilizarse en sentencias de SQL dinámico 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 parámetro sin tipo se pueden utilizar en las sentencias de SQL dinámico siempre y cuando el tipo de datos del marcador de parámetro se pueda calcular basándose en el contexto de la sentencia de SQL (SQLSTATE 42610).

En el ejemplo siguiente se obtiene un error, ya que *c1* se resolvería en el primer contexto en un tipo de datos de serie, mientras que en el segundo contexto *c1* se resolvería en un tipo de datos numérico:

```
SELECT 'Hello' || c1, 5 + c1 FROM (VALUES(?)) AS T(c1)
```

No obstante, la sentencia siguiente es correcta, ya que el marcador de parámetro asociado con la columna derivada, *c1*, se resolvería en un tipo de datos numérico en ambos contextos:

```
SELECT 7 + c1, 5 + c1 FROM (VALUES(?)) AS T(c1)
```

Consulte “Determinación de los tipos de datos de las expresiones sin tipo” para conocer las normas de establecimiento de tipo de un marcador de parámetro sin tipo.

- 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...**

**DESCRIBE**

puede ser cualquier sentencia

**DECLARE CURSOR**

debe ser SELECT

**EXECUTE**

no 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 parámetro, es más eficaz utilizar la sentencia EXECUTE IMMEDIATE en lugar de las sentencias PREPARE y EXECUTE.
- Todas las sentencias preparadas que ha creado una unidad de trabajo permanecen en estado preparado hasta que la aplicación finaliza, con las siguientes excepciones:
  - Una sentencia que se haya preparado dentro de un paquete vinculado con KEEPDYNAMIC NO y que un cursor abierto declarado con la opción WITH HOLD no utilice ya no está en estado preparado cuando la unidad de trabajo finaliza.



- Una sentencia dinámica vinculada con KEEP DYNAMIC NO y que un cursor abierto declarado con la opción WITH HOLD utilice está en un estado preparado hasta el siguiente límite de unidad de trabajo en el que se cierre el cursor.

### 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 debe 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 parámetro.

```
EXEC SQL PREPARE STMT_NAME FROM :holder;
EXEC SQL EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

Suponga que se debe 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 del departamento
MGRNO CHAR(6), -- número del director
ADMRDEPT CHAR(3) -- número del departamento de administración
```

Para insertar el número de departamento G01 que se denomina COMPLAINTS, que no tiene ningún director y que informa al departamento A00, la cláusula INSERT\_DA debe tener los valores de la Tabla 34 antes de emitirse la sentencia EXECUTE.

*Tabla 34. Valores necesarios para la estructura INSERT\_DA*

| Campo de SQLDA | Valor                  |
|----------------|------------------------|
| SQLDAID        | SQLDA                  |
| SQLDABC        | 192 (Véase la nota 1.) |
| SQLN           | 4                      |
| SQLD           | 4                      |
| SQLTYPE        | 452                    |
| SQLLEN         | 3                      |
| SQLDATA        | puntero para G01       |
| SQLIND         | (Véase la nota 2.)     |
| SQLNAME        |                        |
| SQLTYPE        | 449                    |

## PREPARE

Tabla 34. Valores necesarios para la estructura *INSERT\_DA* (continuación)

| <b>Campo de SQLDA</b> | <b>Valor</b>                                                                                                                                                                                           |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLLEN                | 29                                                                                                                                                                                                     |
| SQLDATA               | <i>puntero para COMPLAINTS</i>                                                                                                                                                                         |
| SQLIND                | <i>puntero para 0</i>                                                                                                                                                                                  |
| SQLNAME               |                                                                                                                                                                                                        |
| SQLTYPE               | 453                                                                                                                                                                                                    |
| SQLLEN                | 6                                                                                                                                                                                                      |
| SQLDATA               | (Véase la nota 3.)                                                                                                                                                                                     |
| SQLIND                | <i>puntero para -1</i>                                                                                                                                                                                 |
| SQLNAME               |                                                                                                                                                                                                        |
| SQLTYPE               | 453                                                                                                                                                                                                    |
| SQLLEN                | 3                                                                                                                                                                                                      |
| SQLDATA               | <i>puntero para A00</i>                                                                                                                                                                                |
| SQLIND                | <i>puntero para 0</i>                                                                                                                                                                                  |
| SQLNAME               |                                                                                                                                                                                                        |
| <b>Nota:</b>          |                                                                                                                                                                                                        |
| 1.                    | Este valor es para una sentencia PREPARE que se ha realizado desde una aplicación de 32 bits. Si la sentencia PREPARE se hubiera realizado en una aplicación de 64 bits, SQLDABC tendría el valor 240. |
| 2.                    | El valor de SQLIND para esta SQLVAR se pasa por alto porque SQLTYPE identifica un tipo de datos sin posibilidad de nulos.                                                                              |
| 3.                    | El valor de SQLDATA para esta SQLVAR se pasa por alto porque el valor de SQLIND indica que este es un valor nulo.                                                                                      |

## REFRESH TABLE

La sentencia REFRESH TABLE renueva los datos de una tabla de consulta materializada.

### Invocación

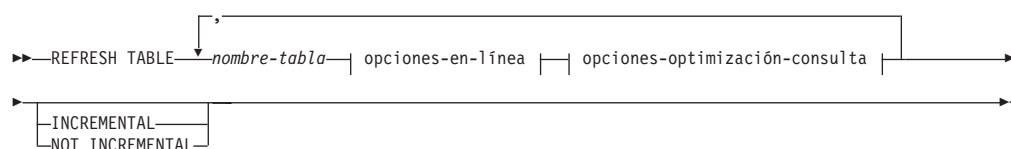
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla
- Autorización DATAACCESS

### Sintaxis



#### opciones-en-línea:



#### opciones-optimización-consulta:



### 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). Ello incluye las tablas de consulta materializada que se han definido con:

- REFRESH IMMEDIATE
- REFRESH DEFERRED

#### *opciones-en-línea*

Especifica la accesibilidad de la tabla mientras se está procesando.

## REFRESH TABLE

### **ALLOW NO ACCESS**

Especifica que ningún otro usuario puede acceder a la tabla mientras se está renovando, excepto si utiliza el nivel de aislamiento de lectura no confirmada.

### **ALLOW READ ACCESS**

Especifica que los demás usuarios tienen acceso de sólo lectura a la tabla mientras se renueva.

### **ALLOW WRITE ACCESS**

Especifica que los demás usuarios tienen acceso de lectura y grabación a la tabla mientras se renueva.

Para impedir una retrotracción de toda la sentencia a causa de un tiempo de espera excedido de bloqueo al utilizar la opción **ALLOW READ ACCESS** o **ALLOW WRITE ACCESS**, se recomienda la emisión de una sentencia **SET CURRENT LOCK TIMEOUT** (especificando la opción **WAIT**) antes de ejecutar la sentencia **REFRESH TABLE** y restaurar el registro especial a su valor anterior posteriormente. Tenga en cuenta, sin embargo, que el registro **CURRENT LOCK TIMEOUT** sólo afecta a un conjunto específico de tipos de bloqueo, no a todos los tipos.

### *opciones-optimización-consulta*

Especifica las opciones de optimización de la consulta para la renovación de las tablas de consulta materializada de **REFRESH DEFERRED**.

### **ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY**

Especifica que, si el registro especial **CURRENT REFRESH AGE** se establece en 'ANY', la renovación de *nombre-tabla* permitirá el uso de las tablas de consulta materializada **REFRESH DEFERRED** para optimizar la consulta que se utilice para renovar *nombre-tabla*. Si el *nombre-tabla* no es una tabla de consulta materializada **REFRESH DEFERRED**, se devuelve un error (SQLSTATE 428FH). Las tablas de consulta materializada **REFRESH IMMEDIATE** siempre se toman en consideración para la optimización de la consulta.

### **INCREMENTAL**

Especifica una renovación incremental de la tabla donde sólo se considera la parte delta (si existe) de sus tablas subyacentes o el contenido de una tabla de etapas asociada (si existe una y si su contenido es coherente). Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que la definición de la tabla de consulta materializada debe volver a calcularse por completo), se devolverá un error (SQLSTATE 55019).

### **NOT INCREMENTAL**

Especifica una renovación completa de la tabla en la que vuelve a realizarse el cálculo de la definición de la tabla de consulta materializada.

Si no se ha especificado **INCREMENTAL** ni **NOT INCREMENTAL**, el sistema determinará si es posible realizar el proceso incremental; si no es posible, se realizará una renovación completa. Si existe una tabla de etapas para la tabla de consulta materializada que va a renovarse, no será posible un proceso incremental porque la tabla de etapas está en estado pendiente, y se devolverá un error (SQLSTATE 428A8). La renovación completa se realizará si la tabla de etapas o si la tabla de consulta materializada está en un estado coherente; de lo contrario, para el proceso incremental se utilizará el contenido de la tabla de etapas.

## Normas

- Si se emite REFRESH TABLE en una tabla de consulta materializada que hace referencia a uno o más apodos, el ID de autorización de la sentencia deberá disponer de autorización para poder realizar selecciones desde las tablas de la fuente de datos (SQLSTATE 42501).

## Notas

- Cuando la sentencia se utiliza para renovar una tabla de consulta materializada REFRESH IMMEDIATE cuyas tablas subyacentes se han cargado, enlazado o desenlazado, puede que el sistema opte por renovar de manera incremental la tabla de consulta materializada con las partes delta de sus tablas subyacentes. Cuando la sentencia se utiliza para renovar una tabla de consulta materializada REFRESH DEFERRED con una tabla de etapas de soporte, puede que el sistema opte por renovar de manera incremental la tabla de consulta materializada con las partes delta de sus tablas subyacentes que se han capturado en la tabla de etapas. Sin embargo, existen algunas situaciones en las que no es posible esta optimización y es necesario realizar una renovación completa (es decir, volver a calcular la definición de la tabla de consulta materializada) para garantizar la integridad de los datos. El usuario puede solicitar explícitamente el mantenimiento incremental especificando la opción INCREMENTAL; si no es posible esta optimización, el sistema devuelve un error (SQLSTATE 55019).
- Si se utiliza la opción ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY, asegúrese de que el orden de renovación de las tablas de consulta materializada REFRESH DEFERRED sea correcto. Por ejemplo, imagine dos tablas de consulta materializada, MQT1 y MQT2, cuyas consultas materializadas comparten las mismas tablas subyacentes. La consulta materializada de MQT2 se puede calcular utilizando MQT1, en lugar de las tablas subyacentes. Si se emplean distintas sentencias para renovar estas dos tablas de consulta materializada, y se renueva MQT2 en primer lugar, el sistema puede elegir utilizar el contenido de MQT1, que todavía no se ha renovado, para renovar MQT2. En ese caso, MQT1 contendría los datos actuales pero MQT2 podría contener todavía datos obsoletos, aunque la renovación de ambas se realizara casi al mismo tiempo. El orden de renovación correcto, si se utilizan dos sentencias REFRESH en lugar de una, es renovar MQT1 primero.
- Si la tabla de consulta materializada tiene una tabla de etapas asociada, la tabla de etapas se podará cuando la renovación se haya ejecutado satisfactoriamente.
- Cualquier control de acceso basado en etiquetas en las tablas base o en la tabla de consulta materializada no interfiere en el proceso de renovación. La renovación tiene lugar como si el control de acceso basado en etiquetas no estuviese presente. La protección automática que se asocia a la tabla de consulta materializada en el momento en que se crea garantiza que los datos de las tablas base permanecen protegidos al pasarlos a la tabla de consulta materializada.
- Solamente para tabla de consulta materializada, SET INTEGRITY FOR *nombre\_mqt* IMMEDIATE CHECKED es lo mismo que REFRESH TABLE *nombre\_mqt*.
- **Uso de las tablas de consulta materializada por parte de REFRESH:** las tablas de consulta materializada no se utilizan para evaluar la *sentencia-select* durante el proceso de la sentencia REFRESH TABLE.
- **Nivel de aislamiento de REFRESH:** el nivel de aislamiento que se utiliza para evaluar la *sentencia-select* es el nivel de aislamiento especificado en la cláusula *nivel-aislamiento* de la *sentencia-select*. O, si no se especifica la cláusula *nivel-aislamiento*, el nivel de aislamiento de la tabla de consulta materializada registrado cuando se ha emitido CREATE TABLE o ALTER TABLE se utiliza para evaluar la *sentencia-select*.

## REFRESH TABLE

- Veamos la sentencia siguiente:

**SET INTEGRITY FOR T IMMEDIATE CHECKED**

En los escenarios siguientes, no reciben soporte ni la opción de comprobación **INCREMENTAL** para T ni una renovación incremental de T (si T es una tabla de consulta materializada (MQT) o una tabla de etapas):

- Se han añadido nuevas restricciones para T mientras se encuentra en estado pendiente de establecimiento de integridad.
  - Cuando ha tenido lugar una operación **LOAD REPLACE** para T, su tabla padre o sus tablas subyacentes.
  - Cuando se ha activado la opción **NOT LOGGED INITIALLY WITH EMPTY TABLE** tras la última comprobación de integridad realizada en T, en sus tablas padre o en sus tablas subyacentes.
  - El efecto de aplicación en cascada del proceso completo, cuando se ha comprobado la integridad de T (o una tabla subyacente, si T es una tabla de consulta materializada o una tabla de etapas) de forma no incremental.
  - Si el espacio de tablas que contiene la tabla o su tabla padre (o tabla subyacente de una tabla de consulta materializada o una tabla de etapas) se ha retrotraído hasta un momento dado y la tabla y su tabla padre (o tabla subyacente si la tabla es una tabla de consulta materializada o una tabla de etapas) residen en distintos espacios de tablas.
  - T es una MQT, y ha tenido lugar una operación **LOAD REPLACE** o **LOAD INSERT** directamente en T tras la última renovación.
- El proceso incremental se utilizará siempre que la situación lo permita, pues es más eficaz. En la mayoría de los casos, no se necesita la opción **INCREMENTAL**. Sin embargo, sí se necesita para garantizar que las comprobaciones de integridad realmente se procesan de forma incremental. Si el sistema detecta que es necesario realizar un proceso completo para garantizar la integridad de los datos, se devuelve un error (SQLSTATE 55019).
  - Si las condiciones para el proceso completo que se describen en el punto anterior no se cumplen, el sistema intentará realizar una renovación incremental (si es una tabla de consulta materializada) cuando el usuario no especifique la opción **NOT INCREMENTAL** para la sentencia **SET INTEGRITY FOR T IMMEDIATE CHECKED**.

## RELEASE (conexión)

La sentencia RELEASE (Conexión) establece una o más conexiones en estado pendiente de liberación.

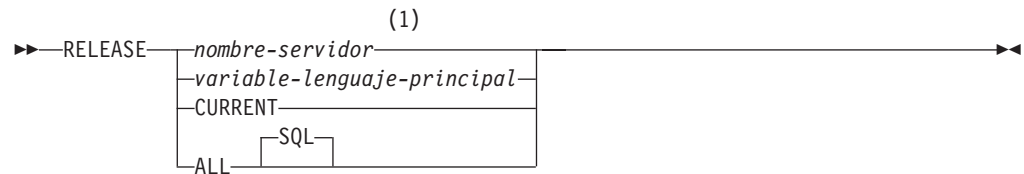
### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente.

### Autorización

No se necesita.

### Sintaxis



### Notas:

- 1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal o un identificador delimitado.

### Descripción

*nombre-servidor* o *variable-lenguaje-principal*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, 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 de indicador. El *nombre-servidor* contenido dentro de la *variable-lenguaje-principal* debe alinearse a la izquierda y no debe delimitarse mediante comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El alias-basedatos especificado o el alias-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el alias-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 o ALL SQL

Indica que todas las conexiones existentes del proceso de aplicación. Este formato de la sentencia RELEASE coloca todas las conexiones existentes del

## RELEASE (conexión)

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.

### 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 salvaguarda especificado. Después de invocar esta sentencia, ya no es posible hacer una retrotracción hasta el punto de salvaguarda.

### Invocación

Esta sentencia puede incorporarse 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-punto-salvaguarda—►►
```

### Descripción

*nombre-punto-salvaguarda*

Especifica el punto de salvaguarda que se debe liberar. Cualquier punto de salvaguarda anidado en el punto de salvaguarda nombrado también se liberará. La retrotracción a ese punto de salvaguarda, o a cualquier punto de salvaguarda anidado en él, ya no será posible. Si el punto de salvaguarda no existe en el nivel de punto de salvaguarda actual (consulte la sección “Normas” en la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B001). El *nombre-puntosalvaguarda* no puede empezar por 'SYS' (SQLSTATE 42939).

### Notas

- El nombre del punto de salvaguarda que se ha liberado se puede volver a utilizar ahora en otra sentencia SAVEPOINT, sin tener en cuenta si se ha especificado la palabra clave UNIQUE en una sentencia SAVEPOINT anterior especificando este mismo nombre de punto de salvaguarda.

### Ejemplo

Liberar un punto de salvaguarda llamado SAVEPOINT1.

```
RELEASE SAVEPOINT SAVEPOINT1
```

## RENAME

La sentencia RENAME cambia el nombre de una tabla o de un índice existente.

### Invocación

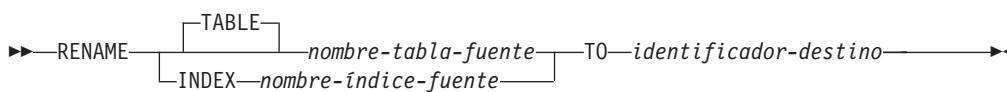
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla o índice
- Propiedad de la tabla o del índice, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.TABLES para una tabla, y en la vista de catálogo SYSCAT.INDEXES para un índice
- Privilegio ALTERIN sobre el esquema
- Autorización DBADM

### Sintaxis



### Descripción

#### TABLE nombre-tabla-fuente

Identifica la tabla existente que se debe renombrar. El nombre, incluyendo el nombre de esquema, debe identificar una tabla que ya exista en la base de datos (SQLSTATE 42704). No debe ser el nombre de una tabla de catálogos (SQLSTATE 42832), una tabla de consulta materializada, una tabla con tipo (SQLSTATE 42997), una tabla temporal creada, una tabla temporal global declarada (SQLSTATE 42995), un apodo o un objeto que no sea una tabla o un alias (SQLSTATE 42809). La palabra clave TABLE es opcional.

El nombre no debe identificar una tabla a la que se haga referencia en una definición de permiso de fila o una definición de máscara de columna (SQLSTATE 42917).

#### INDEX nombre-índice-fuente

Especifica el nombre del índice existente cuyo nombre va a cambiarse. El nombre, incluido el nombre de esquema, debe identificar un índice que ya exista en la base de datos (SQLSTATE 42704). No puede ser el nombre de un índice en una tabla temporal creada o una tabla temporal global declarada (SQLSTATE 42995). El nombre de esquema no debe ser SYSIBM, SYSCAT, SYSFUN o SYSSTAT (SQLSTATE 42832).

#### identificador-destino

Especifica el nuevo nombre de la tabla o del índice sin un nombre de esquema. El nombre de esquema del objeto de fuente se utiliza para calificar el nuevo nombre del objeto. El nombre calificado *no* debe identificar una tabla, vista, alias o índice que ya exista en la base de datos (SQLSTATE 42710).

## Normas

Cuando se cambia el nombre de una tabla, la tabla fuente debe ajustarse a las normas siguientes:

- Debe existir una referencia a ella en todas las definiciones de la tabla de consulta materializada
- Debe ser la tabla objeto de un activador existente
- 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
- No estar referenciada en un objeto XSR que se ha habilitado para su descomposición

Se devuelve un error (SQLSTATE 42986) si la tabla fuente viola una o más de esas condiciones.

Cuando se cambia el nombre de un índice:

- El índice fuente no debe ser un índice generado por el sistema para una tabla de implementación en la que se basa una tabla con tipo (SQLSTATE 42858).

## Notas

- Las entradas del catálogo se actualizan para reflejar el nuevo nombre de la tabla o índice.
- *Todas* las autorizaciones que se asocian a la tabla fuente o al nombre del índice se *transfieren* al nuevo nombre de tabla o índice (las tablas del catálogo de autorización se actualizan de acuerdo con ello).
- Los índices definidos en la tabla fuente se *transfieren* a la nueva tabla (las tablas de catálogo de índice se actualizan del modo apropiado).
- RENAME TABLE invalida cualquier paquete que dependa de la tabla fuente. RENAME INDEX invalida cualquier paquete que dependa del índice fuente.
- Si se utiliza un alias para el *nombre-tabla-fuente*, éste deberá resolverse de modo que dé como resultado un nombre de tabla. La tabla se renombra dentro del esquema de la tabla. El alias no se cambia por la sentencia RENAME y continúa haciendo referencia al anterior nombre de tabla.
- El nombre de una tabla con una clave primaria o con restricciones de unicidad podrá cambiarse si ninguna clave foránea hace referencia a la clave primaria o a las restricciones de unicidad.

## Ejemplos

- *Ejemplo 1:* cambiar el nombre de la tabla EMP por EMPLOYEE.

```
RENAME TABLE EMP TO EMPLOYEE
RENAME TABLE ABC.EMP TO EMPLOYEE
```

- *Ejemplo 2:* cambiar el nombre del índice NEW-IND por IND.

```
RENAME INDEX NEW-IND TO IND
RENAME INDEX ABC.NEW-IND TO IND
```

---

# RENAME STOGROUP

La sentencia RENAME STOGROUP cambia el nombre de un grupo de almacenamiento existente.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCtrl o SYSADM.

### Sintaxis

```
►—RENAME—STOGROUP—nombre-grupo-almacenamiento-fuente—————►
►—TO—nombre-grupo-almacenamiento-destino—————◄◄
```

### Descripción

*nombre-grupo-almacenamiento-fuente*

Identifica el grupo de almacenamiento cuyo nombre se debe cambiar; *nombre-grupo-almacenamiento-fuente* debe identificar un grupo de almacenamiento que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte.

*nombre-grupo-almacenamiento-destino*

Da nombre al grupo de almacenamiento. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo-almacenamiento-destino* no debe identificar un grupo de almacenamiento que ya exista en el catálogo (SQLSTATE 42710). El *nombre-grupo-almacenamiento-destino* no debe empezar con los caracteres 'SYS' (SQLSTATE 42939).

### Normas

- No se puede ejecutar la sentencia RENAME STOGROUP mientras se está añadiendo un servidor de particiones de base de datos (SQLSTATE 55071).

## RENAME TABLESPACE

La sentencia RENAME TABLESPACE cambia el nombre de un espacio de tablas existente.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCtrl o SYSADM.

### Sintaxis

```
►—RENAME—TABLESPACE—nombre-espacio-tablas-fuente—————►
►—TO—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 renombrar. Se trata de 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. Se trata de 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 renombrar (SQLSTATE 42832).
- Los espacios de tablas cuyo estado sea "recuperación pendiente" o "recuperación en proceso" no se pueden renombrar (SQLSTATE 55039)

### Notas

- Cuando se renombra 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.

## RENAME TABLESPACE

### Ejemplo

Cambie el nombre del espacio de tablas USERSPACE1 a DATA2000:

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

## REPEAT

La sentencia REPEAT ejecuta una sentencia o grupo de sentencias hasta que se cumpla una condición de búsqueda.

### Invocación

Esta sentencia se puede incluir en:

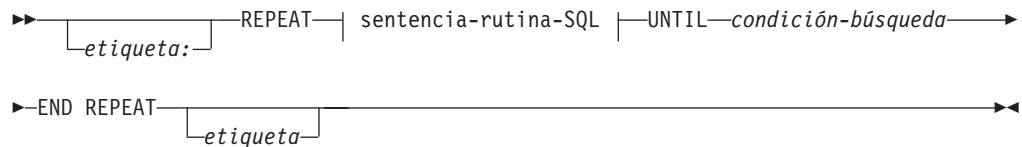
- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

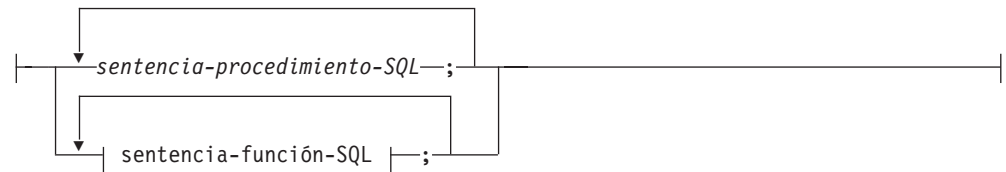
### Autorización

No se requieren privilegios para invocar la sentencia REPEAT. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia REPEAT.

### Sintaxis



### sentencia-rutina-SQL:



### 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 las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL o de una sentencia de SQL compuesto (compilado). Consulte *sentencia-procedimiento-SQL* en la sentencia de "SQL compuesto (compilado)".

## REPEAT

### *sentencia-función-SQL*

Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de un activador de SQL, una función de SQL o de un método de SQL. Consulte *sentencia-función-SQL* en "FOR".

### *condición-búsqueda*

La *condición-búsqueda* se evalúa tras cada ejecución del bucle REPEAT. Si la condición es verdadera, se sale del bucle. Si la condición es desconocida o falsa, el bucle continúa.

## Ejemplo

Una sentencia REPEAT busca filas en una tabla hasta que se invoca el manejador 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
 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
```



## RESIGNAL

La sentencia RESIGNAL se utilizan en un manejador de condiciones para retransmitir la condición que ha activado el manejador o para generar una condición alternativa a fin de poderse procesar a un nivel más alto. Produce una excepción, un aviso o una condición de no encontrado que se debe devolver, junto con el texto de mensaje opcional.

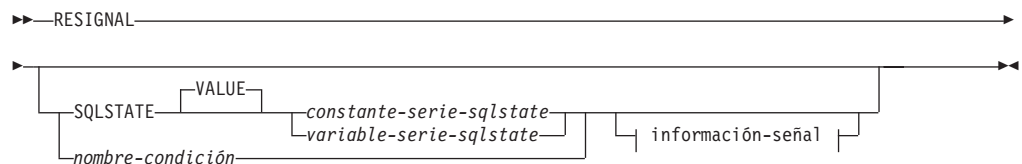
### Invocación

Esta sentencia sólo puede incorporarse en un manejador de condiciones de una sentencia de SQL compuesto (compilado). La sentencia de SQL compuesto (compilado) puede incorporarse en una definición de procedimiento, función o activador de SQL.

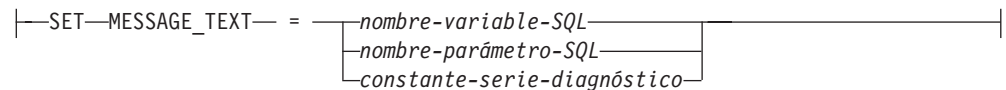
### Autorización

Si se hace referencia a una condición de módulo, los privilegios del ID de autorización de la sentencia deben incluir el privilegio EXECUTE en el módulo.

### Sintaxis



### 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 de caracteres con exactamente 5 caracteres que siguen las normas para 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 normas, se produce un error (SQLSTATE 428B3).

#### SQLSTATE VALUE

Especifica el SQLSTATE que se devolverá. Puede utilizarse cualquier valor válido de SQLSTATE. El valor especificado debe respetar las reglas para los SQLSTATE:

- Cada carácter debe formar parte de un conjunto de dígitos (de '0' a '9') o de un conjunto de letras en mayúsculas (de 'A' a 'Z') no acentuadas.

## RESIGNAL

- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

Si SQLSTATE no se ajusta a estas normas, se devuelve un error.

### *constante-serie-sqlstate*

El valor de *constante-serie-sqlstate* debe ser una constante de serie de caracteres que contenga exactamente 5 caracteres.

### *variable-serie-sqlstate*

La variable de SQL o parámetro de SQL que se especifica debe corresponder al tipo de datos CHAR(5) y no debe ser un valor nulo.

### *nombre-condición*

Especifica el nombre de una condición que se devolverá. El *nombre-condición* debe declararse en la sentencia-compuesta o identificar una condición que existe en el servidor actual.

### **SET MESSAGE\_TEXT =**

Especifica una serie de caracteres que describe el error o aviso. La serie de caracteres se devuelve en el campo sqlerrmc de la SQLCA. Si la serie tiene más de 70 bytes de longitud, se truncará sin avisar de ello.

### *nombre-variable-SQL*

Identifica una variable de SQL que se debe declarar dentro de la sentencia compuesta que contiene el texto del mensaje.

### *nombre-parámetro-SQL*

Identifica un parámetro de SQL, que se ha definido para la rutina, que contiene el texto del mensaje. El parámetro de SQL debe estar definido 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 se emite una sentencia RESIGNAL sin especificar una cláusula SQLSTATE o un *nombre-condición*, se devuelve la misma condición que ha invocado el manejador. El SQLSTATE, el SQLCODE y la SQLCA que se asocian a la condición no cambian.
- Si una sentencia RESIGNAL se emite usando un *nombre-condición* que no tiene un valor de SQLSTATE asociado y la condición no se gestiona, se devuelve SQLSTATE 45000 y el SQLCODE se establece en -438. Tenga en cuenta que una condición de este tipo no se manejará por un manejador de condiciones para SQLSTATE 45000 que se encuentre en el ámbito de la rutina que emite la sentencia RESIGNAL.
- Si una sentencia RESIGNAL se emite utilizando un valor SQLSTATE o un *nombre-condición* con un valor SQLSTATE asociado, el valor SQLCODE devuelto se basa en el valor SQLSTATE de la siguiente manera:
  - Si la clase de SQLSTATE especificada es '01' ó '02', se devuelve un aviso o una condición de no encontrado y el SQLCODE se establece en +438.
  - De otro modo, se devuelve una condición de excepción y el SQLCODE se establece en -438.
- Una sentencia RESIGNAL tiene los campos del SQLCA establecidos de la forma siguiente:
  - Los campos sqlerrd se establecen en cero.
  - Los campos sqlwarn se establecen en blancos.
  - El campo sqlerrmc se establece en los 70 primeros bytes de MESSAGE\_TEXT.

- El campo `sqlerrml` se establece en la longitud de `sqlerrmc` o bien en cero si no se ha especificado ninguna cláusula `SET MESSAGE_TEXT`.
- El campo `sqlerrp` se establece en `ROUTINE`
- Consulte la sección "Notas" en "Sentencia SIGNAL" para obtener más información acerca de los valores de `SQLSTATE`.

## Ejemplo

Este ejemplo detecta los errores de división por cero. La sentencia `IF` utiliza una sentencia `SIGNAL` para invocar el manejador de condiciones *overflow*. El manejador 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
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
```

## RETURN

La sentencia RETURN se utiliza para concluir la ejecución de una 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.

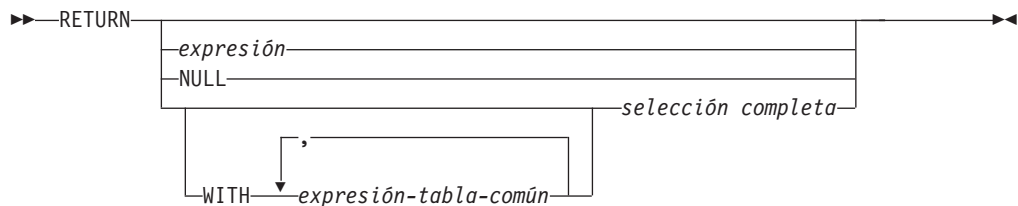
### Invocación

Esta sentencia se puede incluir en una función de SQL, un método de SQL o un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se requieren privilegios para invocar la sentencia RETURN. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar cualquier expresión o selección completa incorporada en la sentencia RETURN.

### Sintaxis



### Descripción

#### *expresión*

Especifica un valor devuelto procedente de la rutina:

- Si la rutina es una función o un método distintos de una función de tabla compilada, debe especificarse una *expresión*, NULL o *selección-completa* (SQLSTATE 42631) y el tipo de datos del resultado debe poder asignarse al tipo RETURNS de la rutina (SQLSTATE 42866).
- Si la rutina es una función de tabla en línea, no puede especificarse una expresión escalar (que no sea una selección completa escalar) (SQLSTATE 428F1). Si la rutina es una función de tabla compilada, no puede especificarse una expresión.
- Si la rutina es un procedimiento, el tipo de datos de *expresión* debe ser INTEGER (SQLSTATE 428F2). 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 puede especificarse NULL para un RETURN de una función de tabla, función de fila o procedimiento.

#### WITH *expresión-tabla-común*

Define una expresión de tabla común para utilizarla con la *selección completa* que viene a continuación.

#### *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 ser asignables a los tipos de columnas declaradas del resultado de la función, utilizando las normas para la asignación a columnas (SQLSTATE 42866).

La *selección-completa* no puede especificarse para un RETURN de un procedimiento o una función de tabla compilada.

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 en línea, puede devolver cero o más filas con una o más columnas. Si la selección completa tiene cero filas de resultados, la sentencia RETURN no devuelve ninguna fila para la tabla de resultados.

## Normas

- La ejecución de un método o una función SQL debe finalizar con una sentencia RETURN (SQLSTATE 42632).
- En una función de tabla de SQL que utiliza una sentencia de SQL compuesto (compilado), no puede especificarse una *expresión*, NULL o *selección-completa*. Las filas se devuelven desde la función utilizando la sentencia PIPE, y se requiere la sentencia RETURN como la última sentencia que ha de ejecutarse cuando la función salga (SQLSTATE 2F005).
- En una función de fila o tabla de SQL que utiliza una sentencia de SQL compuesto (en línea), la única sentencia RETURN permitida es la que se encuentra al final de la sentencia compuesta. (SQLSTATE 429BD).

## Notas

- Cuando se devuelve un valor desde un procedimiento, el llamador puede acceder al valor:
  - utilizando la sentencia GET DIAGNOSTICS para recuperar el valor de DB2\_RETURN\_STATUS cuando se ha llamado al procedimiento de SQL desde otro procedimiento de 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 desde el campo sqlerrd[0] de la SQLCA, tras procesarse la cláusula CALL de un procedimiento de SQL. Este campo sólo es válido si el SQLCODE es cero o un valor positivo (de lo contrario, debe darse por supuesto un valor de -1).

## Ejemplo

Utilice una sentencia RETURN para salir de un procedimiento de 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
```

### REVOKE (autorizaciones de bases de datos)

Esta forma de la sentencia REVOKE revoca las autorizaciones que se aplican a toda la base de datos.

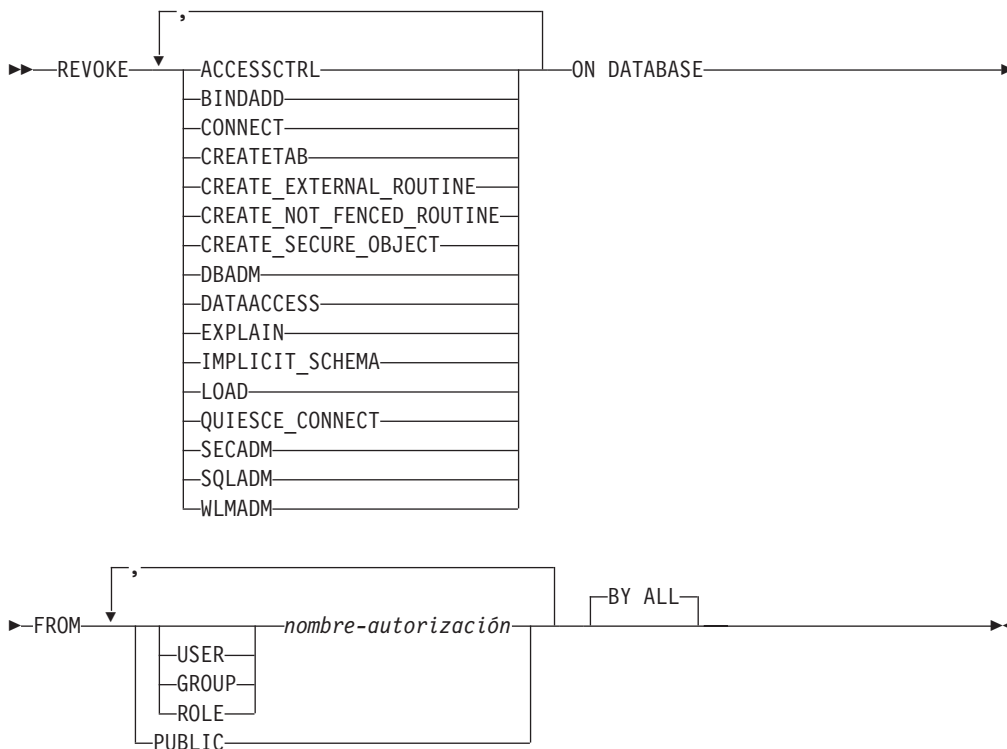
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Para revocar la autorización ACCESSCTRL, CREATE\_SECURE\_OBJECT, DATAACCESS, DBADM o SECADM, se necesita la autorización SECADM. Para revocar otras autorizaciones, se requiere la autorización ACCESSCTRL o SECADM.

#### Sintaxis



#### Descripción

##### ACCESSCTRL

Revoca la autorización para otorgar y revocar la mayoría de las autorizaciones de base de datos y privilegios de objetos.

##### 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\_EXTERNAL\_ROUTINE

Revoca la autorización para registrar rutinas externas. Cuando una rutina externa se ha registrado, esta sigue existiendo, aunque posteriormente se revoque la autorización CREATE\_EXTERNAL\_ROUTINE del ID de autorización que ha registrado la rutina.

La autorización CREATE\_EXTERNAL\_ROUTINE no puede revocarse de un *nombre-autorización* que tenga autorización DBADM o CREATE\_NOT\_FENCED\_ROUTINE sin que también se revoque la autorización DBADM o CREATE\_NOT\_FENCED\_ROUTINE (SQLSTATE 42504).

### CREATE\_NOT\_FENCED\_ROUTINE

Revoca la autorización para registrar rutinas que se ejecutan en el proceso del gestor de bases de datos. Cuando una rutina se ha registrado como no delimitada, esta sigue ejecutándose de esta forma, aunque posteriormente se revoque la autorización CREATE\_NOT\_FENCED\_ROUTINE del ID de autorización que ha registrado la rutina.

La autorización CREATE\_NOT\_FENCED\_ROUTINE no puede revocarse de un *nombre-autorización* que tenga autorización DBADM sin que también se revoque la autorización DBADM (SQLSTATE 42504).

### CREATE\_SECURE\_OBJECT

Revoca la autorización para crear activadores seguros y funciones seguras. También revoca la autorización para alterar el atributo seguro de esos objetos.

### DATAACCESS

Revoca la autorización para acceder a los datos.

### 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)

### PRECAUCIÓN:

La revocación de la autorización DBADM no revoca automáticamente los privilegios que ostentaba *nombre-autorización* sobre los objetos de la base de datos.

### EXPLAIN

Revoca la autorización para explicar, preparar y describir sentencias estáticas y dinámicas sin necesidad de acceder a los datos.

### IMPLICIT\_SCHEMA

Revoca la autorización para crear implícitamente un esquema. No afecta a la capacidad de crear objetos en esquemas existentes o procesar una sentencia CREATE SCHEMA.

La autorización IMPLICIT\_SCHEMA no puede revocarse de un *nombre-autorización* que contenga la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

### LOAD

Revoca la autorización para cargar (LOAD) en esta base de datos.

### QUIESCE\_CONNECT

Revoca la autorización para acceder a la base de datos mientras está inmovilizada.

### SECADM

Revoca la autorización para administrar la seguridad de la base de datos.

### SQLADM

Revoca la autorización para supervisar y ajustar las sentencias de SQL.

### WLMADM

Revoca la autorización para gestionar los objetos del gestor de carga de trabajo.

### 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.

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Revoca las autorizaciones de PUBLIC.

### BY ALL

Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.



### Normas

**Administrador de seguridad obligatorio:** la base de datos debe tener al menos un ID de autorización de tipo USER con la autorización SECADM. La autorización SECADM no puede revocarse desde cada ID de autorización de usuario (SQLSTATE 42523).

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.DBAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### Notas

- La revocación de un privilegio específico no revoca necesariamente la capacidad de realizar una acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o un rol tienen otros privilegios o si el usuario tiene un grado de autorización más alto como, por ejemplo, DBADM.
- **Alternativas de sintaxis:** Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos.
  - CREATE\_NOT\_FENCED puede especificarse en lugar de CREATE\_NOT\_FENCED\_ROUTINE
  - Puede especificarse SYSTEM en lugar de DATABASE
  - Puede especificarse NOT INCLUDING DEPENDENT PRIVILEGES como sintaxis alternativa

### Ejemplos

- *Ejemplo 1:* Dado que USER6 sólo es un usuario y no un grupo, revoque el privilegio para crear las tablas del usuario USER6.
- *Ejemplo 2:* Revoque la autorización BINDADD en la base de datos de un grupo denominado D024. Existen dos filas en la vista de catálogo SYSCAT.DBAUTH para el usuario al que se otorga la autorización; una con un GRANTEETYPE que es U y otra con un GRANTEETYPE que es 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).

- *Ejemplo 3:* Revoque la autorización del administrador de seguridad del usuario Walid.

```
REVOKE SECADM ON DATABASE FROM USER Walid
```

- *Ejemplo 4:* Un usuario con autorización SECADM revoca la autorización CREATE\_SECURE\_OBJECT del usuario Haytham.

```
REVOKE CREATE_SECURE_OBJECT ON DATABASE FROM USER HAYTHAM
```

## REVOKE (exención)

Este formato de la sentencia REVOKE revoca una exención a una norma de acceso de control de acceso basado en etiquetas (LBAC).

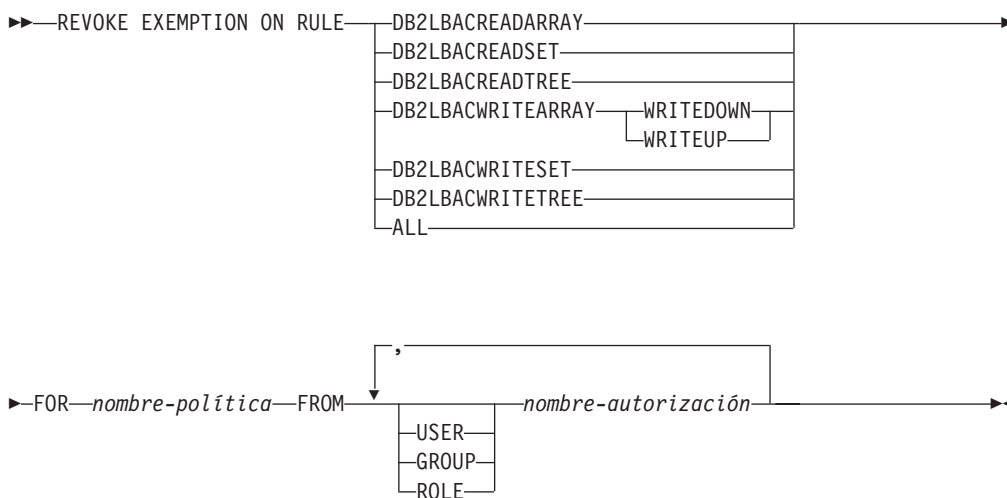
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

### Sintaxis



### Descripción

#### EXEMPTION ON RULE

Revoca la exención en una norma de acceso.

#### DB2LBACREADARRAY

Revoca una exención en la norma DB2LBACREADARRAY predefinida.

#### DB2LBACREADSET

Revoca una exención en la norma DB2LBACREADSET predefinida.

#### DB2LBACREADTREE

Revoca una exención en la norma DB2LBACREADTREE predefinida.

#### DB2LBACWRITEARRAY

Revoca una exención en la norma DB2LBACWRITEARRAY predefinida.

#### WRITEDOWN

Especifica que la exención sólo se aplica a la grabación hacia abajo.

#### WRITEUP

Especifica que la exención sólo se aplica a la grabación hacia arriba.

**DB2LBACWRITESET**

Revoca una exención en la norma DB2LBACWRITESET predefinida.

**DB2LBACWRITETREE**

Revoca una exención en la norma DB2LBACWRITETREE predefinida.

**ALL**

Revoca las exenciones en todas las normas predefinidas.

**FOR** *nombre-política*

Especifica el nombre de la política de seguridad en la que se revocarán las exenciones.

**FROM**

Especifica a quién se le revoca la exención.

**USER**

Especifica que el *nombre-autorización* identifica a un usuario.

**GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

**ROLE**

Especifica que el *nombre-autorización* identifica un nombre de rol.

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles.

**Normas**

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SECURITYPOLICYEXEMPTIONS en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

**Ejemplos**

- *Ejemplo 1:* Revocar la exención en la norma de acceso DB2LBACREADSET para la política de seguridad DATA\_ACCESS del usuario WALID.

```
REVOKE EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS
FROM USER WALID
```

- *Ejemplo 2:* Revoca una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEDOWN para la política de seguridad DATA\_ACCESS del usuario BOBBY.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
FOR DATA_ACCESS FROM USER BOBBY
```

- *Ejemplo 3:* Revoca una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEUP para la política de seguridad DATA\_ACCESS del usuario BOBBY.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
FOR DATA_ACCESS FROM USER BOBBY
```

### REVOKE (privilegios de variable global)

Este formato de la sentencia REVOKE revoca uno o más privilegios de una variable global creada.

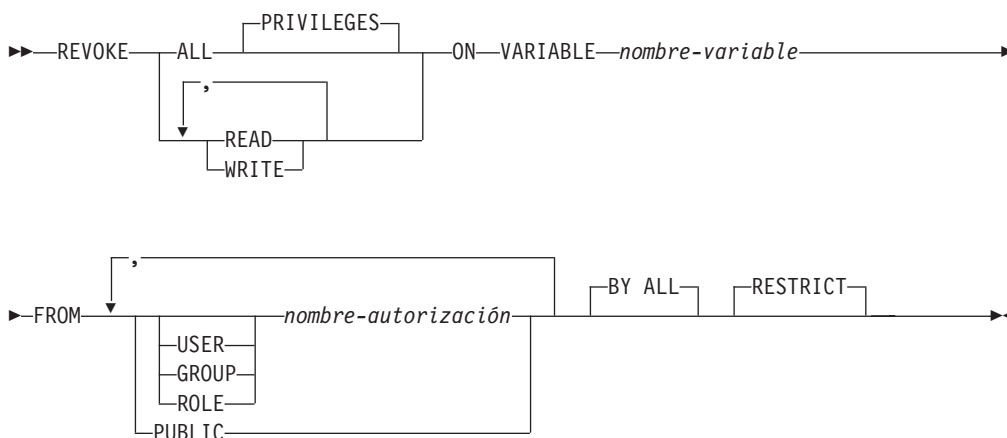
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

#### Sintaxis



#### Descripción

##### ALL PRIVILEGES

Revoca todos los privilegios que retiene un *nombre-autorización* para la variable global especificada. Si no se especifica ALL, debe especificarse READ o WRITE. No se debe especificar READ o WRITE más de una vez.

##### READ

Revoca el privilegio para leer el valor de la variable global especificada.

##### WRITE

Revoca el privilegio para asignar un valor a la variable global especificada.

##### ON VARIABLE *nombre-variable*

Identifica la variable global sobre la que van a revocarse uno o más privilegios. El *nombre-variable* debe identificar una variable global que existe en el servidor actual y que no es una variable de módulo (SQLSTATE 42704).

##### FROM

Especifica a quién se revocarán los privilegios.

##### USER

Especifica que el *nombre-autorización* identifica a un usuario.

## REVOKE (privilegios de variable global)

### GROUP

Especifica que el *nombre-autorización* identifica a un grupo.

### ROLE

Especifica que el *nombre-autorización* identifique un rol existente en el servidor actual (SQLSTATE 42704).

### *nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Revoca los privilegios especificados de PUBLIC.

### BY ALL

Revoca todos los privilegios especificados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

### RESTRICT

Especifica que la sentencia dará error si se revoca algún objeto que dependa del privilegio. Éste es el comportamiento por omisión.

## Normas

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.VARIABLEAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092).
- Si alguna función SQL, método SQL, procedimiento, vista, desencadenante u otra variable global contiene una variable global y depende de que se revoque el privilegio, la operación de revocación fallará (SQLSTATE 42893).

## Notas

- Si el privilegio READ se revoca en una variable global, los paquetes con una dependencia para grabar el valor de la variable global (por ejemplo, por medio de la sentencia SET) no resultan afectados, ya que la grabación en una variable global está controlada por medio del privilegio WRITE en dicha variable global.
- Si el privilegio WRITE se revoca en una variable global, los paquetes con una dependencia para leer el valor de la variable global no resultan afectados, ya que la lectura de una variable global está controlada por medio del privilegio READ en dicha variable global.
- La revocación de un privilegio no afecta necesariamente la posibilidad de realizar la acción. Es posible que un usuario no pueda proceder en el caso de que se retenga el privilegio requerido por medio de la pertenencia a un rol o grupo diferente o por medio de PUBLIC.

## Ejemplo

Revoque el privilegio WRITE en la variable global MYSCHEMA.MYJOB\_PRINTER del usuario ZUBIRI.

## REVOKE (privilegios de variable global)

```
REVOKE WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER FROM ZUBIRI
```

## REVOKE (privilegios de índice)

Esta forma de la sentencia REVOKE revoca el privilegio CONTROL en un índice.

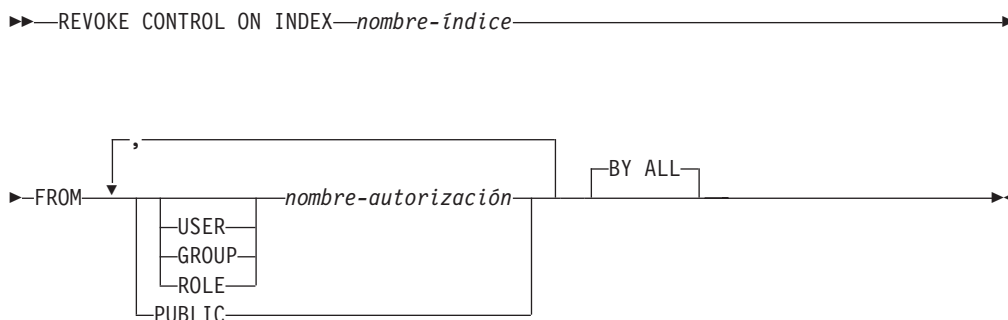
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis



### Descripción

#### CONTROL

Revoca el privilegio para descartar 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 debe revocar el privilegio CONTROL.

#### FROM

Indica de quién se deben revocar los privilegios.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

#### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

#### nombre-autorización,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

#### PUBLIC

Revoca los privilegios de PUBLIC.

## REVOKE (privilegios de índice)

### BY ALL

Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Éste es el comportamiento por omisión.

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.INDEXAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o un rol tienen otros privilegios o si el usuario tiene autorizaciones como, por ejemplo, ALTERIN para el esquema de un índice.

### Ejemplos

- *Ejemplo 1:* Dado que USER4 es sólo un usuario y no un grupo, revoque el privilegio para descartar un índice DEPTIDX del usuario USER4.  

```
REVOKE CONTROL ON INDEX DEPTIDX FROM KIESLER
```
- *Ejemplo 2:* Revoque el privilegio para descartar un índice LUNCHITEMS del usuario CHEF y del grupo WAITERS.  

```
REVOKE CONTROL ON INDEX LUNCHITEMS
FROM USER CHEF, GROUP WAITERS
```



## REVOKE (privilegios de módulo)

Este formato de la sentencia REVOKE revoca el privilegio sobre un módulo.

### Invocación

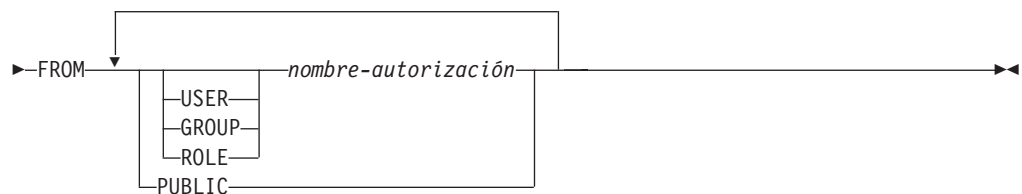
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis

► REVOKE EXECUTE ON MODULE *nombre-módulo* →



### Descripción

#### EXECUTE

Revoca el privilegio para hacer referencia a objetos de módulo publicados. Esto incluye revocar el privilegio para:

- Ejecutar cualquier rutina publicada definida en el módulo.
- Leer y grabar en cualquier variable global publicada definida en el módulo.
- Hacer referencia a cualquier tipo definido por el usuario publicado que esté definido en el módulo.
- Hacer referencia a cualquier condición publicada definida en el módulo.

#### ON MODULE *nombre-módulo*

Identifica el módulo del que se ha revocado el privilegio. El *nombre-módulo* debe identificar un módulo que exista en el servidor actual (SQLSTATE 42704).

#### FROM

Indica 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.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

## REVOKE (privilegios de módulo)

*nombre-autorización*

Lista uno o varios ID de autorización. No se debe especificar el mismo *nombre-autorización* más de una vez

### **PUBLIC**

Otorga el privilegio a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

### **Ejemplo**

El ejemplo siguiente muestra cómo revocar el privilegio EXECUTE de un módulo denominado *myModa* del usuario *jones*

```
REVOKE EXECUTE ON MODULE MYMODA FROM JONES
```

## REVOKE (privilegios de paquete)

Esta forma de la sentencia REVOKE revoca los privilegios CONTROL, BIND y EXECUTE en un paquete.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

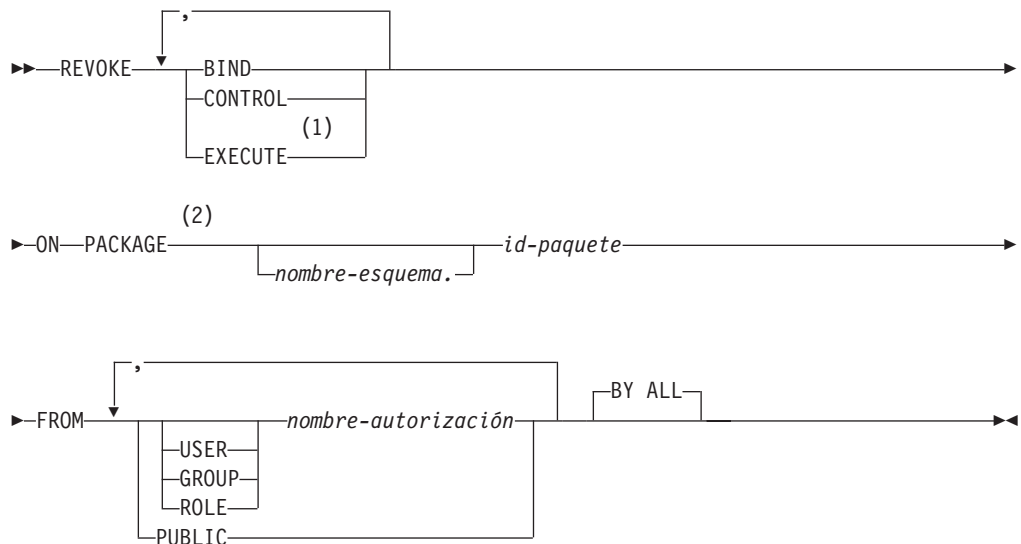
### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CONTROL para el paquete referenciado
- Autorización ACCESSCTRL o SECADM

La autorización ACCESSCTRL o SECADM es necesaria para revocar el privilegio CONTROL.

### 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 para el paquete al que se hace referencia o bien para añadir una nueva versión del paquete al que se hace referencia.

## REVOKE (privilegios de paquete)

El privilegio BIND no puede revocarse de un *nombre-autorización* que disponga de privilegio CONTROL para el paquete sin que se revoque también el privilegio CONTROL.

### CONTROL

Revoca el privilegio para descartar 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-esquema.id-paquete*

Especifica el nombre del paquete para el que van a revocarse privilegios. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. La revocación de un privilegio de paquete se aplica a todas las versiones del paquete.

### FROM

Indica de quién se deben revocar los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

*nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Revoca los privilegios de PUBLIC.

### BY ALL

Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.PACKAGEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o un rol tienen otros privilegios o si el usuario tiene privilegios como, por ejemplo, ALTERIN para 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 (rol)

Este formato de la sentencia REVOKE revoca roles de usuarios, grupos u otros roles.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

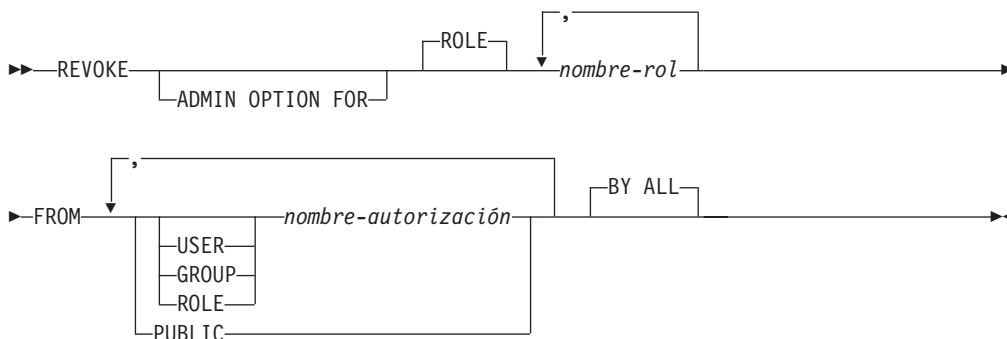
### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- WITH ADMIN OPTION en el rol
- Autorización SECADM

Se necesita la autorización SECADM para revocar ADMIN OPTION FOR *nombre-rol* de un *nombre-autorización* o para revocar un *nombre-rol* de un *nombre-autorización* que tiene WITH ADMIN OPTION en dicho rol.

### Sintaxis



### Descripción

#### ADMIN OPTION FOR

Revoca WITH ADMIN OPTION en *nombre-rol*. WITH ADMIN OPTION en *nombre-rol* debe retenerse por medio de *nombre-autorización* o por medio de PUBLIC, si se especifica PUBLIC (SQLSTATE 42504). Si se especifica la cláusula ADMIN OPTION FOR, sólo se revoca WITH ADMIN OPTION en ROLE *nombre-rol*, no el propio rol.

#### ROLE *nombre-rol*

Especifica el rol que va a revocarse. El *nombre-rol* debe identificar un rol existente en el servidor actual (SQLSTATE 42704) que se haya otorgado a *nombre-autorización* o a PUBLIC, si se ha especificado PUBLIC (SQLSTATE 42504).

#### FROM

Especifica a quién se revoca el rol.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

**GROUP**

Especifica que el *nombre-autorización* identifica a un grupo.

**ROLE**

Especifica que el *nombre-autorización* identifique un rol existente en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

**PUBLIC**

Revoca los roles especificadas de PUBLIC.

**BY ALL**

Revoca el *nombre-rol* de cada *nombre-autorización* especificado al que se otorgó explícitamente dicho rol, sin tener en cuenta quién lo otorgó. Éste es el comportamiento por omisión.

**Normas**

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.ROLEAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092).
- El *nombre-rol* no debe identificar un rol o un rol que contenga *nombre-rol*, si el rol tiene privilegio EXECUTE sobre una rutina o privilegio USAGE sobre una secuencia y un objeto SQL que no sea un paquete depende de la rutina o secuencia (SQLSTATE 42893). El propietario del objeto SQL es *nombre-autorización* o cualquier usuario que sea miembro de *nombre-autorización*, donde *nombre-autorización* es un rol.

**Notas**

- Si se revoca un rol de un *nombre-autorización* o de PUBLIC, todos los privilegios que retiene el rol ya no están a disposición del *nombre-autorización* o de PUBLIC a través de dicho rol.
- La revocación de un rol no revoca necesariamente la posibilidad de realizar una acción concreta por medio de un privilegio otorgado a dicho rol. Un usuario puede proseguir si PUBLIC, un grupo al que pertenezca el usuario u otro rol otorgado al usuario tienen otros privilegios o si el usuario tiene un nivel de autorización más alto como, por ejemplo, DBADM.

**Ejemplos**

- *Ejemplo 1:* Revoque el rol INTERN del rol DOCTOR y el rol DOCTOR del rol SPECIALIST.

```
REVOKE ROLE INTERN FROM ROLE DOCTOR
```

```
REVOKE ROLE DOCTOR FROM ROLE SPECIALIST
```

- *Ejemplo 2:* Revoque el rol INTERN de PUBLIC.

```
REVOKE ROLE INTERN FROM PUBLIC
```

## REVOKE (rol)

- *Ejemplo 3:* Revoque el rol SPECIALIST del usuario BOB y del grupo TORONTO.

```
REVOKE ROLE SPECIALIST
FROM USER BOB, GROUP TORONTO BY ALL
```



## REVOKE (privilegios de rutina)

Esta forma de la sentencia REVOKE revoca privilegios para una rutina (función, método o procedimiento) que no está definida en un módulo.

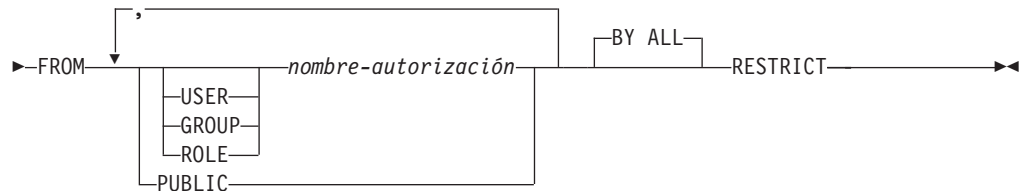
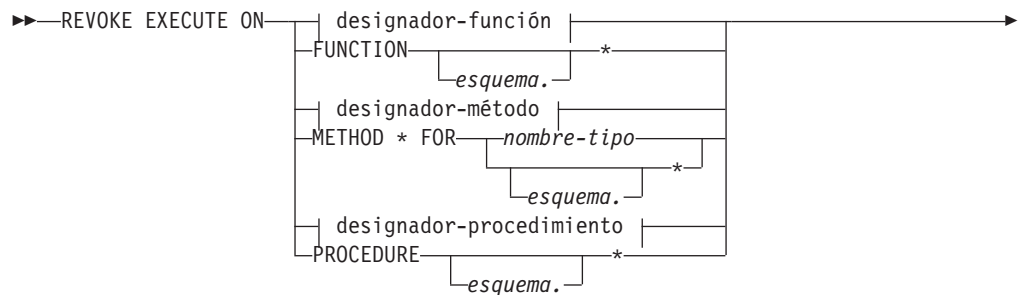
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

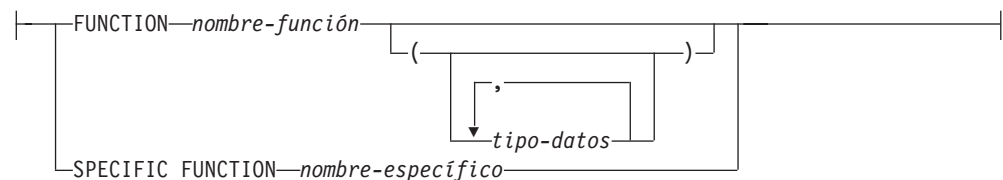
### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis

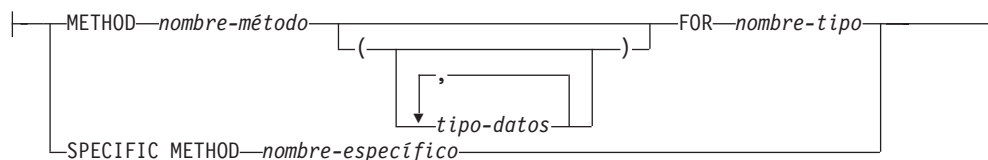


### designador-función:

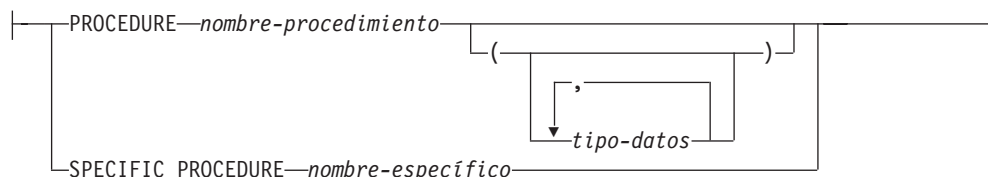


## REVOKE (privilegios de rutina)

### designador-método:



### designador-procedimiento:



## Descripción

### EXECUTE

Revoca el privilegio para ejecutar la función, el método o el procedimiento definido por el usuario que se identifica.

#### *designador-función*

Identifica de forma exclusiva la función a la que se revoca el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

### FUNCTION *esquema.\**

Identifica el otorgamiento explícito para todas las funciones existentes y futuras del esquema. La revocación del privilegio *esquema.\** no revoca ninguno de los privilegios que se han otorgado para una función específica. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

#### *designador-método*

Identifica de forma exclusiva el método al que se revoca el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

### METHOD \*

Identifica el otorgamiento explícito para todos los métodos existentes y futuros para el tipo *nombre-tipo*. La revocación del privilegio \* no revoca ninguno de los privilegios que se han otorgado para un método específico.

#### FOR *nombre-tipo*

Especifica el nombre del tipo en el que se encuentra el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el valor del registro especial CURRENT SCHEMA se utiliza como calificador de un nombre de tipo no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados. Para identificar el otorgamiento explícito para todos los métodos existentes y futuros para todos los tipos existentes y futuros del esquema, puede utilizarse un asterisco (\*) en lugar del *nombre-tipo*. La revocación del privilegio mediante

la utilización de un asterisco para el método y el *nombre-tipo* no revoca ninguno de los privilegios que se han otorgado para un método específico o para todos los métodos para un tipo específico.

### *designador-procedimiento*

Identifica de forma exclusiva el procedimiento al que se revoca el privilegio. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22.

### **PROCEDURE** *esquema.\**

Identifica el otorgamiento explícito para todos los procedimientos existentes y futuros del esquema. La revocación del privilegio *esquema.\** no revoca ninguno de los privilegios que se han otorgado para un procedimiento específico. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

### **FROM**

Especifica a quién se le revoca el privilegio EXECUTE.

### **USER**

Especifica que el *nombre-autorización* identifica a un usuario.

### **GROUP**

Especifica que el *nombre-autorización* identifica un nombre de grupo.

### **ROLE**

Especifica que el *nombre-autorización* identifica un nombre de rol.

### *nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### **PUBLIC**

Revoca el privilegio EXECUTE de PUBLIC.

### **BY ALL**

Revoca el privilegio EXECUTE de todos los usuarios especificados a los que se ha otorgado explícitamente el privilegio, independientemente de qué usuario lo ha otorgado. Éste es el comportamiento por omisión.

### **RESTRICT**

Especifica que el privilegio EXECUTE no puede revocarse si se cumplen las dos condiciones siguientes (SQLSTATE 42893):

- La rutina especificada se utiliza en una vista, activador, restricción, extensión de índice, función de SQL, método de SQL, grupo de transformación o bien se hace referencia a ésta como SOURCE de una función derivada.
- La pérdida del privilegio EXECUTE puede dar lugar a que el propietario de la vista, del activador, de la restricción, de la extensión de índice, de la función de SQL, del método de SQL, del grupo de transformación o de la función derivada ya no pueda ejecutar la rutina especificada.

## **Normas**

- No es posible revocar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:

## REVOKE (privilegios de rutina)

- Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.ROUTINEAUTH en la que el otorgado es *nombre-autorización*:
  - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
  - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
  - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
  - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### Notas

- Si un paquete depende de una rutina (rol, método o procedimiento) y se revoca el privilegio EXECUTE sobre dicha rutina de PUBLIC, un usuario o un rol, el paquete se convierte en no operativo si la rutina es una función o método y se convierte en no válido si la rutina es un procedimiento, a menos que el propietario del paquete todavía tenga el privilegio EXECUTE sobre la rutina. El propietario del paquete todavía tiene el privilegio EXECUTE si:
  - Se ha otorgado explícitamente el privilegio EXECUTE al propietario del paquete
  - El propietario del paquete es miembro de un rol que tiene el privilegio EXECUTE
  - Se ha otorgado el privilegio EXECUTE a PUBLIC

Dado que los privilegios de grupo no se consideran para los paquetes estáticos, el paquete se convierte en no operativo (en el caso de una función o de un método) o en no válido (en el caso de un procedimiento) incluso si un grupo al que pertenece el propietario del paquete tiene el privilegio EXECUTE.

### Ejemplos

- *Ejemplo 1:* Revoque el privilegio EXECUTE para la función CALC\_SALARY del usuario JONES. Se da por supuesto que, en el esquema, sólo existe una función con el nombre de función CALC\_SALARY.  
**REVOKE EXECUTE ON FUNCTION CALC\_SALARY FROM JONES RESTRICT**
- *Ejemplo 2:* Revoque el privilegio EXECUTE para el procedimiento VACATION\_ACCR de todos los usuarios del servidor actual.  
**REVOKE EXECUTE ON PROCEDURE VACATION\_ACCR FROM PUBLIC RESTRICT**
- *Ejemplo 3:* Revoque el privilegio EXECUTE para la función NEW\_DEPT\_HIRES de HR (Recursos humanos). La función tiene dos parámetros de entrada de tipo INTEGER y CHAR(10) respectivamente. Se da por supuesto que el esquema tiene más de una función denominada NEW\_DEPT\_HIRES.  
**REVOKE EXECUTE ON FUNCTION NEW\_DEPT\_HIRES (INTEGER, CHAR(10))  
FROM HR RESTRICT**
- *Ejemplo 4:* Revoque el privilegio EXECUTE para el método SET\_SALARY para el tipo EMPLOYEE del usuario Jones.  
**REVOKE EXECUTE ON METHOD SET\_SALARY FOR EMPLOYEE FROM JONES RESTRICT**

## REVOKE (privilegios de esquema)

Esta forma de la sentencia REVOKE revoca los privilegios en un esquema.

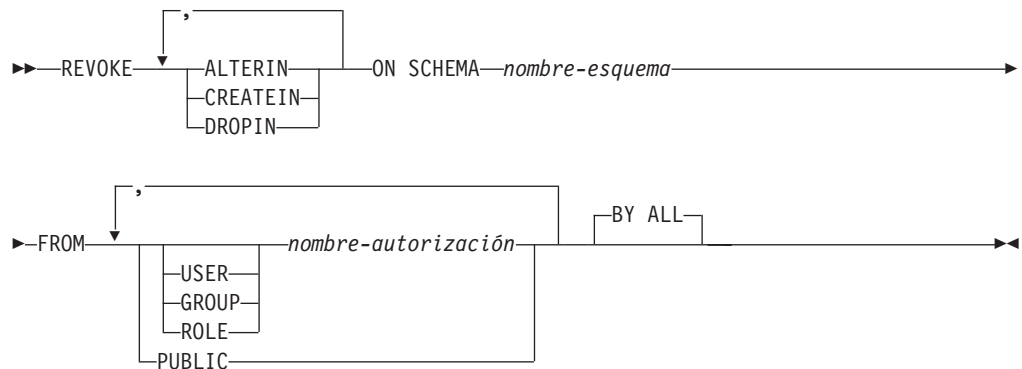
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### 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 descartar objetos en el esquema.

#### ON SCHEMA nombre-esquema

Especifica el nombre del esquema en el que se deben revocar los privilegios.

#### FROM

Indica de quién se deben revocar los privilegios.

#### USER

Especifica que el nombre-autorización identifica a un usuario.

#### GROUP

Especifica que el nombre-autorización identifica un nombre de grupo.

#### ROLE

Especifica que el nombre-autorización identifica un nombre de rol.

nombre-autorización,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

## REVOKE (privilegios de esquema)

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### **PUBLIC**

Revoca los privilegios de PUBLIC.

### **BY ALL**

Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

## **Normas**

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SCHEMAAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

## **Notas**

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o un rol tienen otros privilegios o si el usuario tiene un grado de autorización más alto 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 descartar objetos en el esquema LUNCH del usuario CHEF y del grupo WAITERS.  

```
REVOKE DROPIN ON SCHEMA LUNCH
FROM USER CHEF, GROUP WAITERS
```

## REVOKE (etiqueta de seguridad)

Este tipo de sentencia REVOKE revoca una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC).

### Invocación

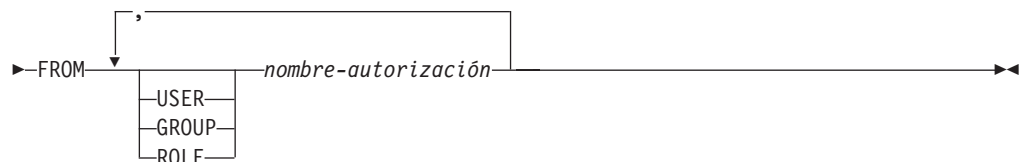
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

### Sintaxis

►—REVOKE SECURITY LABEL—*nombre-etiqueta-seguridad*—►



### Descripción

**SECURITY LABEL** *nombre-etiqueta-seguridad*

Revoca la etiqueta de seguridad *nombre-etiqueta-seguridad*. El nombre debe calificarse con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704) y que esté retenida por *nombre-autorización* (SQLSTATE 42504).

#### FROM

Especifica a quién se le revoca la etiqueta de seguridad especificada.

#### USER

Especifica que el *nombre-autorización* identifica a un usuario.

#### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

#### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol. El nombre de rol debe existir en el servidor actual (SQLSTATE 42704).

*nombre-autorización,...*

Lista los ID de autorización de uno o más usuarios, grupos o roles.

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:

## REVOKE (etiqueta de seguridad)

- Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SECURITYLABELACCESS en la que el otorgado es *nombre-autorización*:
  - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
  - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
  - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
  - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### Ejemplo

Revocar la etiqueta de seguridad EMPLOYEESECLABEL, que forma parte de la política de seguridad DATA\_ACCESS, del usuario WALID.

```
REVOKE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
FROM USER WALID
```



## REVOKE (privilegios de secuencia)

Esta forma de la sentencia REVOKE revoca los privilegios en una secuencia.

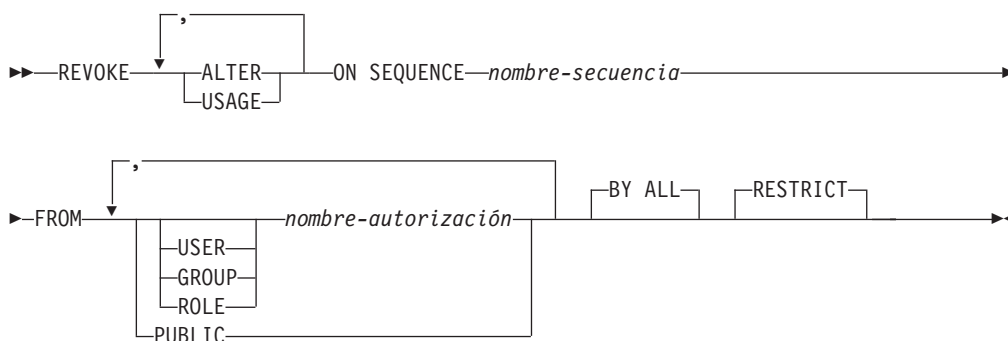
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica. No obstante, si se aplica la opción de vinculación DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

### Sintaxis



### Descripción

#### ALTER

Revoca el privilegio de cambiar las propiedades de una secuencia o reiniciar la generación de números de secuencia mediante la sentencia ALTER SEQUENCE.

#### USAGE

Revoca el privilegio de poder hacer referencia a una secuencia mediante una *expresión-nextval* o *expresión-prevval*.

#### ON SEQUENCE nombre-secuencia

Identifica la secuencia para la que van a revocarse los privilegios especificados. El nombre de la secuencia, incluido un calificador de esquema implícito o explícito, debe identificar de forma exclusiva a una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704).

#### FROM

Especifica a quién se revocarán los privilegios.

##### USER

Especifica que el *nombre-autorización* identifica a un usuario.

##### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

## REVOKE (privilegios de secuencia)

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Revoca los privilegios especificados de PUBLIC.

### BY ALL

Revoca todos los privilegios especificados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

### RESTRICT

Esta palabra clave opcional indica que la sentencia dará error si se revoca algún objeto que dependa del privilegio.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SEQUENCEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

## Notas

- Al revocar un privilegio sobre una secuencia del ID de autorización bajo el cual se ha vinculado un paquete, el paquete se convierte en no válido si el ID de autorización no sigue teniendo el privilegio sobre la secuencia a través de diferentes medios como, por ejemplo, ser miembro de un rol que tiene el privilegio.
- La revocación de un privilegio específico no elimina necesariamente la capacidad de realizar una acción. Un usuario puede continuar si PUBLIC o un grupo al que pertenezca el usuario tienen otros privilegios, o si el usuario tiene un nivel de autorización más alto como, por ejemplo, DBADM.

## Ejemplos

- *Ejemplo 1:* Revoque al usuario ENGLÉS el privilegio USAGE para una secuencia denominada GENERATE\_ID. Hay una fila en la vista de catálogo SYSCAT.SEQUENCEAUTH para esta secuencia y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.  
**REVOKE USAGE ON SEQUENCE GENERATE\_ID FROM ENGLÉS**
- *Ejemplo 2:* Revoque los privilegios de modificación en la secuencia GENERATE\_ID que se han otorgado previamente a todos los usuarios locales. (Lo otorgado a usuarios específicos no se ve afectado.)  
**REVOKE ALTER ON SEQUENCE GENERATE\_ID FROM PUBLIC**
- *Ejemplo 3:* Revoque todos los privilegios en la secuencia GENERATE\_ID de los usuarios PELLOW y MLI y del grupo PLANNERS.

## REVOKE (privilegios de secuencia)

```
REVOKE ALTER, USAGE ON SEQUENCE GENERATE_ID
FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

### 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.

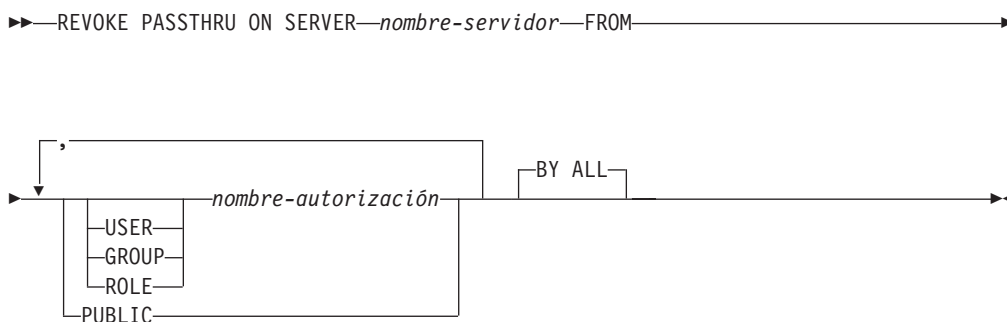
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL o SECADM.

#### 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.

##### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

##### *nombre-autorización,...*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

##### PUBLIC

Revoca de PUBLIC el privilegio de realizar un paso a través de *nombre-servidor*.

### BY ALL

Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Éste es el comportamiento por omisión.

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.PASSTHROUGH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

### 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 (privilegio SETSESSIONUSER)

Este formato de la sentencia REVOKE revoca uno o más privilegios SETSESSIONUSER de uno o más ID de autorización.

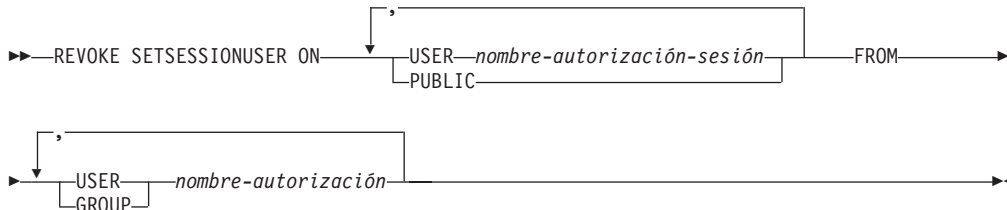
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

#### Sintaxis



#### Descripción

##### SETSESSIONUSER ON

Revoca el privilegio de asumir la identidad de un nuevo ID de autorización.

##### USER nombre-autorización-sesión

Especifica el ID de autorización que el *nombre-autorización* es capaz de asumir, mediante la sentencia SET SESSION AUTHORIZATION. El *nombre-autorización-sesión* debe identificar a un usuario que el *nombre-autorización* pueda suponer, no un grupo (SQLSTATE 42504).

##### PUBLIC

Especifica que se revocarán todos los privilegios para configurar la autorización de la sesión.

##### 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.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### Ejemplos

- *Ejemplo 1:* El usuario PAUL tiene el privilegio de configurar la autorización de la sesión a WALID, y por ello ejecutar las sentencias de SQL como usuario WALID. La sentencia siguiente revoca ese privilegio.

```
REVOKE SETSESSIONUSER ON USER WALID
FROM USER PAUL
```

- *Ejemplo 2:* El usuario GUYLAINE tiene el privilegio de configurar la autorización de la sesión a BOBBY, RICK o KEVIN, y por ello ejecutar las sentencias de SQL como BOBBY, RICK o KEVIN. La sentencia siguiente revoca el privilegio de utilizar dos de esos ID de autorización. Después de que se ejecute esta sentencia, GUYLAINE sólo podrá configurar la autorización de la sesión a KEVIN.

```
REVOKE SETSESSIONUSER ON USER BOBBY, USER RICK
FROM USER GUYLAINE
```

- *Ejemplo 3:* El grupo ACCTG y el usuario WALID pueden configurar la autorización de la sesión a cualquier ID de autorización. La sentencia siguiente revoca ese privilegio de ACCTG y WALID.

```
REVOKE SETSESSIONUSER ON PUBLIC
FROM USER WALID, GROUP ACCTG
```

### REVOKE (privilegios de espacio de tablas)

Esta forma de la sentencia REVOKE revoca el privilegio USE para un espacio de tablas.

#### Invocación

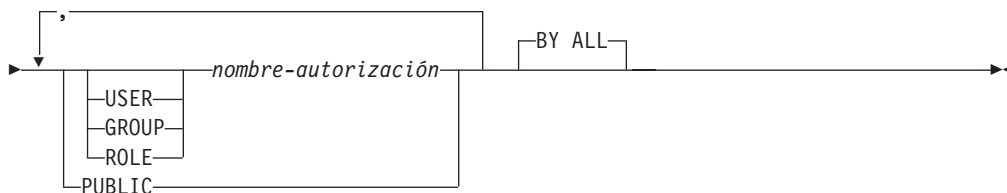
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL, SECADM, SYSCTRL o SYSADM.

#### Sintaxis

►► REVOKE USE OF TABLESPACE—*nombre-espacio-tablas*—FROM —————►



#### 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.

##### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

##### *nombre-autorización*

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).



## REVOKE (privilegios de espacio de tablas)

### PUBLIC

Revoca el privilegio USE a PUBLIC.

### BY ALL

Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Éste es el comportamiento por omisión.

### Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.TBSPACEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve 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.

### Ejemplo

Revocar el privilegio del usuario BOBBY para crear tablas en el espacio de tablas PLANS.

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```

### REVOKE (privilegios de tabla, vista o apodo)

Esta forma de la sentencia REVOKE revoca privilegios en una tabla, vista o apodo.

#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

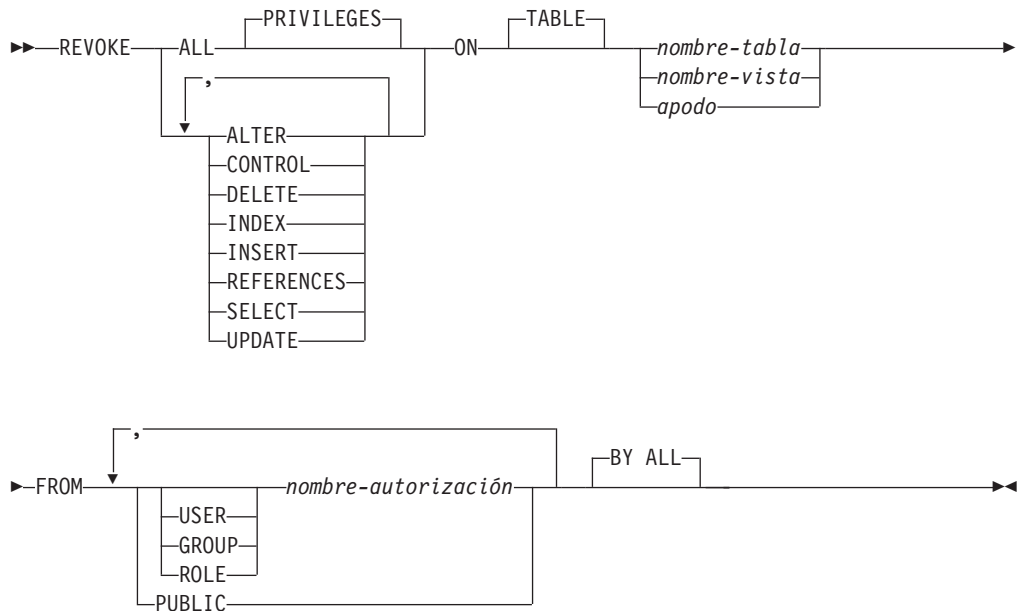
#### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla, vista o apodo referenciado.
- Autorización ACCESSCTRL o SECADM

La autorización ACCESSCTRL o SECADM es necesaria para revocar el privilegio CONTROL o para revocar privilegios sobre tablas y vistas de catálogo.

#### Sintaxis



#### Descripción

##### ALL o ALL PRIVILEGES

Revoca todos los privilegios (excepto CONTROL) del nombre-autorización para las tablas, vistas o apodos especificados.

Si no se utiliza ALL, debe utilizarse una o varias de las palabras clave listadas en la pila de opciones (desde ALTER hasta UPDATE). 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.

## REVOKE (privilegios de tabla, vista o apodo)

### ALTER

Revoca el privilegio para añadir columnas a la definición de la tabla base; crear o descartar una clave primaria o restricción de unicidad en la tabla; crear o descartar una clave foránea en la tabla; añadir/cambiar un comentario en la tabla, vista o apodo; crear o descartar una restricción de comprobación; crear un activador; añadir, restablecer o descartar una opción de columna para un apodo; o cambiar nombres de columna de apodo o tipos de datos.

### CONTROL

Revoca la capacidad para descartar 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, de la vista que puede actualizarse o del apodo.

### 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 descartar el índice o la especificación de índice). Asimismo, el creador mantiene este privilegio incluso si se revoca el privilegio INDEX.

### INSERT

Revoca los privilegios para insertar filas en la tabla, en la vista que puede actualizarse o en el apodo y para ejecutar el programa de utilidad IMPORT.

### REFERENCES

Revoca el privilegio para crear o descartar una clave foránea que haga referencia a la tabla como padre. Cualquier privilegio REFERENCES de nivel de columna también se revoca.

### 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 acerca de las vistas no operativas, consulte "CREATE VIEW".)

### UPDATE

Revoca el privilegio para actualizar filas en la tabla, en la vista que puede actualizarse o en el apodo. 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 deben revocar los privilegios. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

### FROM

Indica de quién se deben revocar los privilegios.

### USER

Especifica que el *nombre-autorización* identifica a un usuario.

### GROUP

Especifica que el *nombre-autorización* identifica un nombre de grupo.

## REVOKE (privilegios de tabla, vista o apodo)

### ROLE

Especifica que el *nombre-autorización* identifica un nombre de rol.

*nombre-autorización*,...

Lista los ID de autorización de uno o varios usuarios, grupos o roles.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

### PUBLIC

Revoca los privilegios de PUBLIC.

### BY ALL

Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

## Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado de las vistas de catálogo SYSCAT.TABAUTH y SYSCAT.COLAUTH en las que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

## Notas

- Si se revoca un privilegio del *nombre-autorización* que es el propietario de la vista (como se ha registrado en la columna OWNER en SYSCAT.VIEWS), dicho privilegio también se revoca de las vistas dependientes.
- Si el propietario de la vista pierde un privilegio SELECT para algún objeto del que depende la definición de vista (o si se descarta un objeto del que depende la definición de vista o se convierte en no operativo en el caso de otra vista), la vista quedará no operativa.

Sin embargo, si un usuario que tiene autorización ACCESSCTRL o SECADM revoca explícitamente todos los privilegios del propietario sobre la vista, el registro de OWNER no aparecerá en SYSCAT.TABAUTH, pero a la vista no le sucederá nada, seguirá estando operativa.

- Los privilegios en vistas no operativas no pueden revocarse.
- Un paquete puede convertirse en no válido cuando el ID de autorización en el que se ha vinculado el paquete pierde un privilegio sobre un objeto del que depende el paquete. El privilegio puede perderse de una de las formas siguientes:
  - El privilegio se revoca desde el ID de autorización
  - El privilegio se revoca desde un rol del cual es miembro el ID de autorización
  - El privilegio se revoca desde PUBLIC

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

## REVOKE (privilegios de tabla, vista o apodo)

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 desde 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 al ejecutarse.

Otro ejemplo es un paquete propiedad de USER1, que es miembro del rol R1. El paquete contiene SELECT de la tabla T1 y se revoca el privilegio SELECT para la tabla T1 desde el rol R1. El paquete se marcará como no válido, partiendo de que USER1 no tiene el privilegio SELECT para la tabla T1 por otros medios.

- Los paquetes, activadores 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, los paquetes, activadores y 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 un privilegio SELECT en cada subtabla o subvista de Z. Estos paquetes pueden convertirse en no válidos y las vistas o activadores pueden pasar a ser no operativos cuando el ID de autorización bajo el que se han vinculado los paquetes o el propietario de los activadores o vistas pierde el privilegio SELECT. El privilegio SELECT puede perderse de una de las formas siguientes:
  - El privilegio SELECT se revoca desde el ID de autorización
  - El privilegio SELECT se revoca desde un rol del cual es miembro el ID de autorización
  - El privilegio SELECT se revoca desde PUBLIC
- 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 realizando una tarea si PUBLIC, un grupo o un rol tienen otros privilegios o si el usuario tiene privilegios como, por ejemplo, ALTERIN para el esquema de una tabla o vista.
- Si el propietario de la tabla de consulta materializada pierde un privilegio SELECT en una tabla de la que depende la definición de la tabla de consulta materializada (o si se descarta una tabla de la que depende la definición de la tabla de consulta materializada), la tabla de consulta materializada se descartará. Sin embargo, si un usuario con la autorización SECADM o ACCESSCTRL revoca explícitamente todos los privilegios del propietario sobre la tabla de consulta materializada, el registro de SYSTABAUTH de OWNER se suprimirá, pero no le sucederá nada a la tabla de consulta materializada; seguirá estando 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 a la que se hace referencia directa o indirectamente en una función de SQL o cuerpo de método podría no ejecutarse satisfactoriamente si la función de SQL o el cuerpo de método no puede descartarse porque algún otro objeto depende de éste (SQLSTATE 42893).
- La revocación del privilegio SELECT hace que se descarte un cuerpo de método o de función SQL cuando:
  - El propietario del cuerpo de método o de función SQL pierde el privilegio SELECT para algún objeto del que depende la definición de cuerpo de

## REVOKE (privilegios de tabla, vista o apodo)

método o de función SQL; tenga en cuenta que el privilegio se puede perder porque se ha revocado desde PUBLIC o desde un rol del que es miembro el propietario.

- Se descarta un objeto del que depende la definición del cuerpo de método o de función SQL

Sin embargo, la revocación falla si otro objeto depende de la función o método (SQLSTATE 42893).

- **Revocación de WITH GRANT OPTION:** la única manera de revocar WITH GRANT OPTION es revocar el propio privilegio y, a continuación, volverlo a otorgar sin especificar WITH GRANT OPTION.
- **Revocación de los privilegios de columna:** la única manera de revocar los privilegios de columna es revocar el privilegio de toda la tabla y, a continuación, volverlo a otorgar para cada columna.

### Ejemplos

- *Ejemplo 1:* Revoque el privilegio SELECT en la tabla EMPLOYEE del usuario ENGLER. 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 ENGLER
```

- *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
```

- *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
```

## REVOKE (privilegios de tabla, vista o apodo)

o bien

```
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 de carga de trabajo)

Este formato de la sentencia REVOKE revoca el privilegio USAGE en una carga de trabajo.

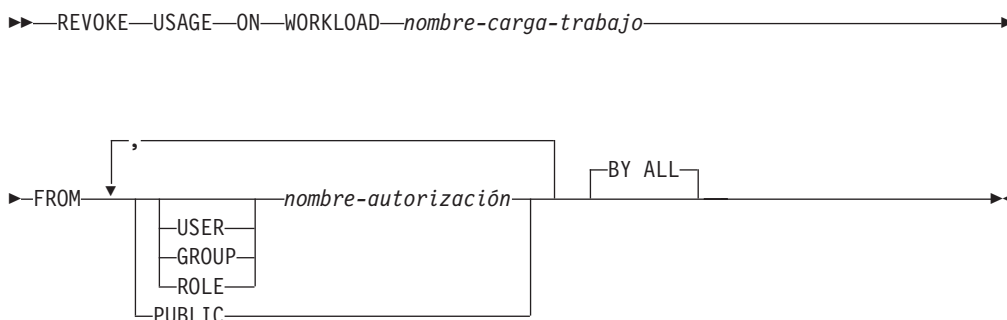
#### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización ACCESSCTRL, SECADM o WLMADM.

#### Sintaxis



#### Descripción

##### USAGE

Revoca el privilegio para utilizar una carga de trabajo.

##### ON WORKLOAD nombre-carga-trabajo

Identifica la carga de trabajo en la que va a revocarse el privilegio USAGE. Este nombre consta de una sola parte. El nombre-carga-trabajo debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El nombre no puede ser 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

##### FROM

Especifica a quién se le revoca el privilegio USAGE.

##### USER

Especifica que el nombre-autorización identifica a un usuario.

##### GROUP

Especifica que el nombre-autorización identifica a un grupo.

##### ROLE

Especifica que el nombre-autorización identifique un rol existente en el servidor actual (SQLSTATE 42704).

##### nombre-autorización,...

Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).



## REVOKE (privilegios de carga de trabajo)

### **PUBLIC**

Revoca el privilegio USAGE de PUBLIC.

### **BY ALL**

Revoca el privilegio USAGE de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Éste es el comportamiento por omisión.

### **Normas**

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.WORKLOADAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092).

### **Notas**

- La sentencia REVOKE no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.

### **Ejemplo**

Revoque el privilegio para utilizar la carga de trabajo CAMPAIGN del usuario LISA.

```
REVOKE USAGE ON WORKLOAD CAMPAIGN FROM USER LISA
```

---

### REVOKE (privilegios de objeto XSR)

Este formato de la sentencia REVOKE otorga el privilegio USAGE sobre un objeto XSR.

#### Invocación

La sentencia REVOKE puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

#### Autorización

Se necesita una de las autorizaciones siguientes:

- Autorización ACCESSCTRL o SECADM

#### Sintaxis

```
►►—REVOKE USAGE ON—XSROBJECT—nombre-objetoxsr—FROM—PUBLIC—BY ALL—►►
```

#### Descripción

**ON XSROBJECT** *nombre-objeto*xsr

Este nombre identifica el objeto XSR sobre el que se ha revocado el privilegio USAGE. El *nombre-objeto*xsr (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este nombre en el esquema especificado, se producirá un error (SQLSTATE 42704).

**FROM PUBLIC**

Revoca el privilegio USAGE de PUBLIC.

**BY ALL**

Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Éste es el comportamiento por omisión.

#### Ejemplo

Revoca los privilegios de uso sobre el esquema XML MYSCHEMA de PUBLIC:

```
REVOKE USAGE ON XSROBJECT MYSCHEMA FROM PUBLIC
```

## 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 salvaguarda.

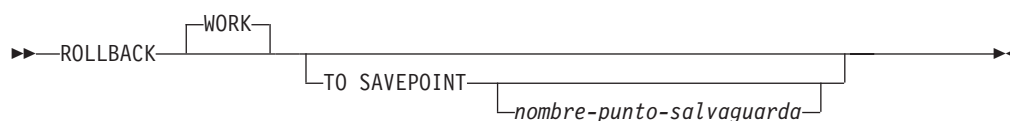
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### 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 la transacción y los cambios que realicen serán independientes de la emisión de la sentencia ROLLBACK:

- SET CONNECTION
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET PASSTHRU (aunque la sentencia SET PASSTHRU no está bajo el control de la transacción, la sesión PASSTHRU que la sentencia inicia está bajo el control de la transacción.)
- SET SERVER OPTION
- Una sentencia SET que establece un registro especial actualizable.

La generación de valores de secuencia y de identidad no están bajo el control de la transacción. Los valores generados y consumidos por la *expresión-nextval* o por la inserción de filas en una tabla que contiene una columna de identidad son independientes de la emisión de la sentencia ROLLBACK. Asimismo, la emisión de la sentencia ROLLBACK no afecta al valor devuelto por la *expresión-prevval*, ni la función IDENTITY\_VAL\_LOCAL.

La modificación de los valores de variables globales no está bajo el control de la transacción. Las sentencias ROLLBACK no afectan a los valores asignados a las variables globales.

#### TO SAVEPOINT

Especifica que debe realizarse una retrotracción parcial (ROLLBACK TO SAVEPOINT). Si no hay ningún punto de salvaguarda activo en el nivel del

## ROLLBACK

punto de salvaguarda actual (consulte la sección "Normas" de la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B502). Tras una retrotracción satisfactoria, el punto de salvaguarda continúa existiendo, pero se liberan los puntos de salvaguarda anidados y ya no existen. Se considera que los puntos de salvaguarda anidados, si existen, se han retrotraído y después liberado como parte de la retrotracción hasta el punto de salvaguarda actual. Si no se facilita un *nombre-puntosalvaguarda*, la retrotracción se produce hasta el punto de salvaguarda definido más recientemente dentro del nivel de punto de salvaguarda actual.

Si se omite esta cláusula, la sentencia ROLLBACK retrotrae toda la transacción. Además, se liberan todos los puntos de salvaguarda definidos dentro de la transacción.

### *nombre-punto-salvaguarda*

Especifica el punto de salvaguarda que se debe utilizar en la operación de retrotracción. El *nombre-puntosalvaguarda* especificado no puede empezar por 'SYS' (SQLSTATE 42939). Tras una operación de retrotracción satisfactoria, el punto de salvaguarda nombrado continúa existiendo. Si el nombre de punto de salvaguarda 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 salvaguarda.

## 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.
- El almacenamiento en antememoria de sentencias se ve afectado por la operación de retrotracción.
- El efecto que una sentencia ROLLBACK TO SAVEPOINT tiene sobre los cursores depende de las sentencias contenidas en el punto de salvaguarda.
  - Si el punto de salvaguarda 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 salvaguarda, el cursor permanece abierto y se coloca delante de la primera fila lógica de la tabla de resultados. (FETCH debe realizarse antes de emitirse una sentencia UPDATE o DELETE con posición.)
    - En otro caso, el cursor no queda afectado por ROLLBACK TO SAVEPOINT (permanece abierto y posicionado).
- Las sentencias de SQL dinámico preparadas en un paquete vinculado con la opción KEEP DYNAMIC YES se conservan en el contexto SQL después de una sentencia ROLLBACK. La sentencia se puede volver a preparar de forma implícita, como resultado de operaciones DDL que se retrotraen dentro de la unidad de trabajo.
- Las sentencias de SQL dinámico preparadas en un paquete vinculado con KEEP DYNAMIC NO se eliminan del contexto SQL tras una operación de retrotracción. La sentencia debe volver a prepararse antes de que se pueda ejecutar en una transacción nueva.

- Las sentencias de SQL dinámico siguientes pueden estar activas durante la operación ROLLBACK:
  - ROLLBACK, sentencia
  - Sentencias CALL en las que se ha ejecutado la sentencia ROLLBACK
- Una operación ROLLBACK TO SAVEPOINT descartará las tablas temporales creadas que se hayan creado en el punto de salvaguarda. Si una tabla temporal creada se modifica en el punto de salvaguarda y esa tabla se ha definido como no anotada cronológicamente, todas las filas de la tabla se suprimen.
- Una operación ROLLBACK TO SAVEPOINT descartará las tablas temporales declaradas que se hayan declarado en el punto de salvaguarda. Si una tabla temporal declarada se modifica en el punto de salvaguarda y esa tabla se ha definido como no anotada cronológicamente, todas las filas de la tabla se suprimen.
- 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
```



- Un nuevo nivel de punto de salvaguarda se inicia cuando se produce uno de los sucesos siguientes:
  - Se inicia una nueva unidad de trabajo (UOW).
  - Se llama a un nuevo procedimiento definido con la cláusula NEW SAVEPOINT LEVEL.
  - Se inicia una sentencia de SQL compuesto atómica.
- Un nivel de punto de salvaguarda termina cuando finaliza o se elimina el suceso que ha provocado su creación. Cuando finaliza un nivel de punto de salvaguarda, se liberan todos los puntos de salvaguarda que contiene. Cualquier cursor abierto, acciones DDL o modificaciones de datos se heredan del nivel de punto de salvaguarda padre (es decir, el nivel de punto de salvaguarda en el cual se ha creado el que acaba de terminar) y están sujetos a cualquier sentencia relacionada con el punto de salvaguarda emitida en el nivel de punto de salvaguarda padre.
- Las normas siguientes se aplican a las acciones de un nivel de punto de salvaguarda:
  - Sólo se puede hacer referencia a los puntos de salvaguarda en el nivel de punto de salvaguarda en el que se establecen. No se puede liberar ni destruir un punto de salvaguarda establecido fuera del nivel actual de punto de salvaguarda, ni tampoco realizar una retrotracción hasta él.
  - Todos los puntos de salvaguarda activos establecidos en el nivel de punto de salvaguarda actual se liberan automáticamente cuando finaliza el nivel de punto de salvaguarda.
  - La exclusividad de los nombres de punto de salvaguarda sólo es obligatoria en el nivel del punto de salvaguarda actual. Los nombres de los puntos de salvaguarda que están activos en otros niveles de punto de salvaguarda pueden reutilizarse en el nivel de punto de salvaguarda actual sin que afecte a los puntos de salvaguarda de los otros niveles de punto de salvaguarda.

## Notas

- Cuando se ha emitido una sentencia SAVEPOINT, las operaciones para realizar inserciones, actualizaciones o supresiones en los apodos no están permitidas.
- La omisión de la cláusula UNIQUE especifica que el otro punto de salvaguarda puede reutilizar el *nombre-puntosalvaguarda* en el nivel de punto de salvaguarda. Si un punto de salvaguarda del mismo nombre ya existe en el nivel de punto de salvaguarda, el punto de salvaguarda existente se destruye y se crea un nuevo punto de salvaguarda con el mismo nombre en el punto actual del proceso. El nuevo punto de salvaguarda se considera como el último punto de salvaguarda establecido por la aplicación. Tenga en cuenta que la destrucción de un punto de salvaguarda mediante la reutilización de su nombre por otro punto de salvaguarda simplemente destruye ese punto de salvaguarda y no libera ningún punto de salvaguarda establecido después del punto de salvaguarda destruido. Estos puntos de salvaguarda subsiguientes sólo pueden liberarse mediante la sentencia RELEASE SAVEPOINT, que libera el punto de salvaguarda nombrado y todos los puntos de salvaguarda establecidos después del punto de salvaguarda nombrado.
- Si se especifica la cláusula UNIQUE, el *nombre-puntosalvaguarda* sólo puede reutilizarse después de que se haya liberado un punto de salvaguarda existente con el mismo nombre.
- En un punto de salvaguarda, si un programa de utilidad, una sentencia de SQL o un mandato DB2 realiza confirmaciones intermitentes durante el proceso, el punto de salvaguarda se liberará implícitamente.

## SAVEPOINT

- Si la sentencia SET INTEGRITY se retrotrae dentro del punto de salvaguarda, los nombres de sentencias preparadas dinámicamente siguen siendo válidos, aunque la sentencia pueda prepararse de nuevo implícitamente.
- Si las inserciones se colocan en almacenamiento intermedio (es decir, la aplicación se ha compilado previamente con la opción INSERT BUF), el almacenamiento intermedio se vaciará cuando se emitan las sentencias SAVEPOINT, ROLLBACK o RELEASE TO SAVEPOINT.

### Ejemplo

Realizar una operación de retrotracción para los puntos de salvaguarda anidados. Primero, cree una tabla denominada DEPARTMENT. Inserte una fila antes de iniciar SAVEPOINT1; inserte otra fila e inicie SAVEPOINT2; después, inserte una tercera fila e inicie SAVEPOINT3.

```
CREATE TABLE DEPARTMENT (
 DEPTNO CHAR(6),
 DEPTNAME VARCHAR(20),
 MGRNO INTEGER)

INSERT INTO DEPARTMENT VALUES ('A20', 'MARKETING', 301)

SAVEPOINT SAVEPOINT1 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('B30', 'FINANCE', 520)

SAVEPOINT SAVEPOINT2 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('C40', 'IT SUPPORT', 430)

SAVEPOINT SAVEPOINT3 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('R50', 'RESEARCH', 150)
```

En este momento, la tabla DEPARTMENT existe con las filas A20, B30, C40 y R50. Si ahora emite:

```
ROLLBACK TO SAVEPOINT SAVEPOINT3
```

la fila R50 ya no estará en la tabla DEPARTMENT. Si después emite:

```
ROLLBACK TO SAVEPOINT SAVEPOINT1
```

la tabla DEPARTMENT todavía existirá, pero las filas insertadas desde que se ha establecido SAVEPOINT1 (B30 y C40) ya no estarán en la tabla.



---

**SELECT**

La sentencia `SELECT` es una forma de consulta.

La sentencia `SELECT` puede incorporarse en un programa de aplicación o emitirse interactivamente.

## SELECT INTO

La sentencia SELECT INTO produce una tabla de resultados 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. Es una sentencia ejecutable que no puede prepararse dinámicamente.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio SELECT para la tabla, vista o apodo
- Privilegio CONTROL para la tabla, vista o apodo
- Autorización DATAACCESS

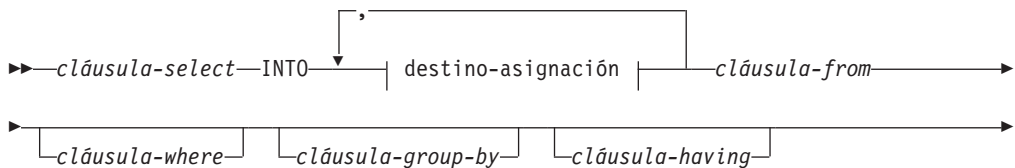
Para cada variable global utilizada como destino de asignación, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes:

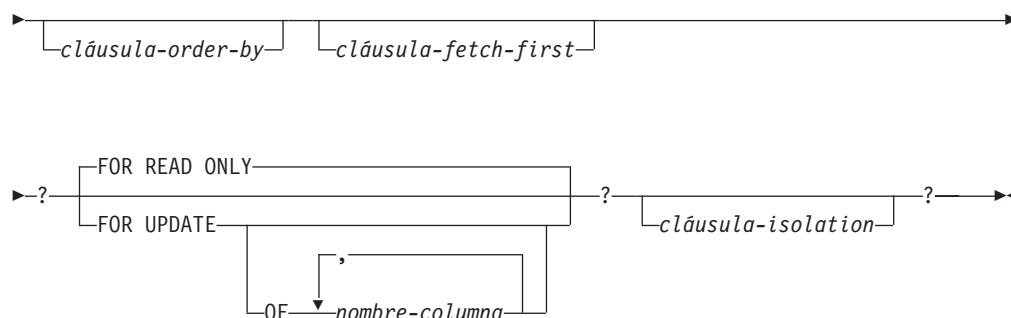
- el privilegio WRITE sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

Los privilegios GROUP no se comprueban para las sentencias SELECT INTO estáticas.

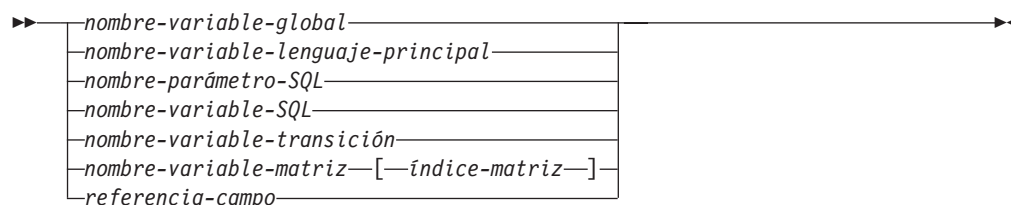
Si el destino de la sentencia SELECT INTO es un apodo, los privilegios para el objeto en la fuente de datos no se toman en consideración hasta que se ejecuta la sentencia en la fuente de datos. En ese momento, el ID de autorización utilizado para conectarse a la fuente de datos debe disponer de los privilegios necesarios a fin de realizar la operación con el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

### Sintaxis





### destino-asignación



### Descripción

Para obtener una descripción de la *cláusula-select*, la *cláusula-from*, la *cláusula-where*, la *cláusula-group-by*, la *cláusula-having*, la *cláusula-order-by*, la *cláusula-fetch-first* y la *cláusula-isolation*, vea “Consultas” en la publicación *Consulta de SQL - Volumen 1*.

#### INTO destino-asignación

Identifica uno o varios destinos para la asignación de los valores de salida.

El primer valor de la fila del resultado se asigna al primer destino de la lista, el segundo valor al segundo destino, etcétera. Cada asignación que se realiza para un *destino-asignación* se realiza secuencialmente y siguiendo la lista. Si se produce un error en cualquier asignación, no se asigna ningún valor a ningún *destino-asignación*.

Cuando el tipo de datos de cada *destino-asignación* no es un tipo de fila, se asigna el valor 'W' al campo SQLWARN3 del SQLCA si el número de *destinos-asignación* es menor que el número de valores de la columna de resultados.

Si el tipo de datos de un *destino-asignación* es un tipo de fila, debe haber exactamente un *destino-asignación* especificado (SQLSTATE 428HR), el número de columnas debe coincidir con el número de campos en el tipo de fila y los tipos de datos de las columnas de la fila captada deben poder asignarse a los campos correspondientes del tipo de fila (SQLSTATE 42821).

Si el tipo de datos de un *destino-asignación* es un elemento de matriz, debe haber exactamente un *destino-asignación* especificado.

#### *nombre-variable-global*

Identifica la variable global que es el sujeto de la asignación.

#### *nombre-variable-lenguaje-principal*

Identifica la variable del lenguaje principal que es el sujeto de la asignación. Para los valores de salida LOB, el destino puede ser una

## SELECT INTO

variable del lenguaje principal normal (si es lo suficientemente grande), una variable de localizador LOB o una variable de referencia a archivos LOB.

### *nombre-parámetro-SQL*

Identifica el parámetro que es el sujeto de la asignación.

### *nombre-variable-SQL*

Identifica la variable de SQL que es el sujeto de la asignación. Las variables de SQL se deben declarar antes de utilizarlas.

### *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 activador, calificado opcionalmente por un nombre de correlación que identifica el nuevo valor.

### *nombre-variable-matriz*

Identifica una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz.

### *[índice-matriz]*

Expresión que especifica qué elemento de la matriz será el destino de la asignación. Para una matriz común, la expresión de *índice-matriz* se debe poder asignar a INTEGER (SQLSTATE 428H1) y no puede ser un valor nulo. Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz (SQLSTATE 2202E). Para una matriz asociativa, la expresión de *índice-matriz* se debe poder asignar al tipo de datos de índice de la matriz asociativa (SQLSTATE 428H1) y no puede ser un valor nulo.

### *referencia-campo*

Identifica el campo de un valor de tipo de fila que es el destino de la asignación. La *referencia-campo* debe especificarse como un *nombre-campo* calificado donde el calificador identifica el valor de fila donde está definido el campo.

## **FOR READ ONLY o FOR UPDATE**

Indica el uso que se desea para la fila seleccionada. El valor por omisión es FOR READ ONLY.

### **FOR READ ONLY**

Especifica que la fila seleccionada no se bloqueará para su actualización.

### **FOR UPDATE**

Especifica que la fila seleccionada de la tabla subyacente se bloqueará para facilitar la actualización de la fila más adelante durante la transacción, situación similar a la del bloqueo que se realiza en la sentencia select de un cursor que incluye la cláusula FOR UPDATE.

FOR UPDATE no debe especificarse si la tabla de resultados de la sentencia SELECT INTO es de sólo lectura (SQLSTATE 42829).

Si se muestran valores de *nombre-columna*, estas columnas deben poder actualizarse (SQLSTATE 42829).

Tenga en cuenta que la visualización de las columnas sólo tiene lugar a efectos informativos y que no se impide que las posteriores sentencias de actualización buscadas puedan modificar otras columnas.

## **Normas**

- Las variables globales no se pueden asignar dentro de activadores que no se han definido mediante una sentencia de SQL compuesto (compilado), funciones que

no se han definido mediante una sentencia de SQL compuesto (compilado), métodos o sentencias de SQL compuesto (en línea) (SQLSTATE 428GX).

## Notas

- **Alternativas de sintaxis:** para mantener la coherencia con consultas SQL:
  - Puede especificarse FOR FETCH ONLY en lugar de FOR READ ONLY

## Ejemplos

- *Ejemplo 1:* Este ejemplo C coloca el salario máximo de la tabla EMP en la variable del lenguaje principal MAXSALARY.

```
EXEC SQL SELECT MAX(SALARY)
 INTO :MAXSALARY
 FROM EMP;
```

- *Ejemplo 2:* Este ejemplo C coloca la fila del empleado 528671 (de la tabla EMP) en las variables del lenguaje principal.

```
EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
 FROM EMP
 WHERE EMPNO = '528671';
```

- *Ejemplo 3:* Este ejemplo SQLJ coloca la fila del empleado 528671 (de la tabla EMP) en las variables del lenguaje principal. Esa fila se actualizará más tarde con una actualización de búsqueda y se debe bloquear cuando se ejecuta la consulta.

```
#sql { SELECT * INTO :FIRSTNAME, :LASTNAME, :EMPNO, :SALARY
 FROM EMP
 WHERE EMPNO = '528671'
 FOR UPDATE };
```

- *Ejemplo 4:* este ejemplo C coloca el salario máximo de la tabla EMP en la variable global GV\_MAXSALARY.

```
EXEC SQL SELECT MAX(SALARY)
 INTO GV_MAXSALARY
 FROM EMP;
```

---

# SET COMPILATION ENVIRONMENT

La sentencia SET COMPILATION ENVIRONMENT cambia el entorno de compilación actual en la conexión para que coincida con los valores incluidos en el entorno de compilación proporcionado por un supervisor de sucesos. Esta sentencia cambia los valores de uno o más registros especiales; estos cambios, a su vez, afectarán a la compilación de cualquier sentencia de SQL dinámico subsiguiente.

La sentencia no está bajo el control de la transacción.

### Invocación

La sentencia puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis

►► SET COMPILATION ENVIRONMENT  *variable-lenguaje-principal* ◀◀

### Descripción

*variable-lenguaje-principal*

Una variable de tipo BLOB que contiene un entorno de compilación proporcionado por un supervisor de sucesos. No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815). Si el formato del entorno de compilación es incorrecto, se devolverá un error y los valores de la conexión permanecerán sin modificar (SQLSTATE 51040).

### Notas

- Para restablecer el entorno de compilación en los valores por omisión originales, termine y luego reinicie la conexión. Puede conseguir el mismo efecto emitiendo esta sentencia dentro de una rutina de SQL, de modo que ningún cambio realizado en los registros especiales se refleje en la conexión al volver de esa rutina.
- Utilice la función de tabla COMPILATION\_ENV a fin de observar los elementos individuales que contiene el entorno de compilación.

### Ejemplo

Establecer el entorno de compilación de la sesión actual en los valores incluidos en un entorno de compilación capturado anteriormente por un supervisor de sucesos de punto muerto. Un supervisor de sucesos de punto muerto que se ha creado especificando la opción WITH DETAILS HISTORY capturará el entorno de compilación de las sentencias de SQL dinámico. Este entorno capturado es el que se acepta como entrada en la sentencia.

```
SET COMPILATION ENVIRONMENT = :hv1
```

## SET CONNECTION

La sentencia SET CONNECTION cambia el estado de una conexión de inactivo a actual, convirtiendo la ubicación especificada en el servidor actual.

La sentencia no está bajo el control de la transacción.

### Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente.

### Autorización

No se necesita.

### Sintaxis

```

▶▶ SET CONNECTION [nombre-servidor
 [variable-lenguaje-principal]
▶▶

```

### Descripción

*nombre-servidor* o *variable-lenguaje-principal*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, 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 de indicador. El *nombre-servidor* contenido dentro de la *variable-lenguaje-principal* debe alinearse a la izquierda y no debe delimitarse mediante comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El *nombre-servidor* o la *variable-lenguaje-principal* 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).
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma forma que con "CONNECT (Tipo 1)", donde encontrará información adicional.

#### Conexión no satisfactoria

Si la sentencia SET CONNECTION falla:

## SET CONNECTION

- 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 tema sobre las “opciones que rigen la semántica de la unidad de trabajo distribuida”) no impide utilizar SET CONNECTION, pero la sentencia no es necesaria, porque en su lugar se pueden utilizar sentencias CONNECT de 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 con el estado de bloqueos, cursores y sentencias preparadas.

### Ejemplo

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 DECFLOAT ROUNDING MODE

La sentencia SET CURRENT DECFLOAT ROUNDING MODE verifica que la modalidad de redondeo especificada sea el valor que está establecido actualmente para el registro especial CURRENT DECFLOAT ROUNDING MODE.

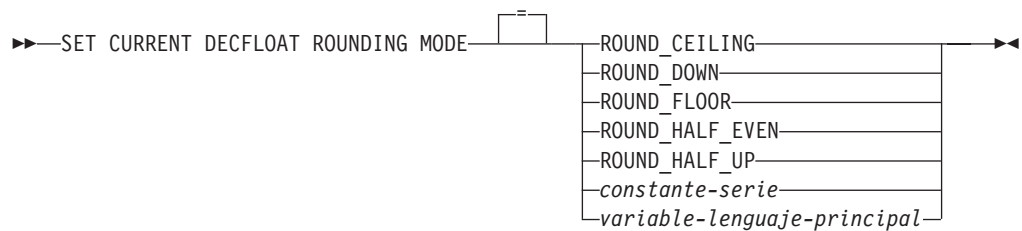
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### ROUND\_CEILING

Redondea el valor hacia el infinito positivo. Si todos los dígitos descartados son cero o si el signo es negativo, el resultado no cambia (a excepción de la eliminación de los dígitos descartados). De lo contrario, el coeficiente de resultado se incrementa en 1.

#### ROUND\_DOWN

Redondea el valor hacia 0 (truncamiento). Se ignoran los dígitos descartados.

#### ROUND\_FLOOR

Redondea el valor hacia el infinito negativo. Si todos los dígitos descartados son cero o si el signo es positivo, el resultado no cambia (a excepción de la eliminación de los dígitos descartados). De lo contrario, el signo es negativo y el coeficiente de resultado se incrementa en 1.

#### ROUND\_HALF\_EVEN

Redondea el valor al valor más próximo. Si los valores son equidistantes, redondea el valor de forma que el dígito final sea par. Si los dígitos descartados representan más de la mitad del valor de un número en la siguiente posición izquierda, el coeficiente de resultado se incrementa en 1. Si representan menos de la mitad, el coeficiente de resultado no se ajusta (es decir, se ignoran los dígitos descartados). De lo contrario, el coeficiente de resultado no se modifica si el dígito situado más a la derecha es par o se incrementa en 1 si el dígito situado más a la derecha es impar (para convertirlo en dígito par).

#### ROUND\_HALF\_UP

Redondea el valor al valor más próximo. Si los valores son equidistantes, redondea el valor por exceso. Si los dígitos descartados representan la mitad o

## SET CURRENT DECFLOAT ROUNDING MODE

más de la mitad del valor de un número en la siguiente posición izquierda, el coeficiente de resultado se incrementa en 1. De lo contrario, los dígitos descargados se ignoran.

### *constante-serie*

Constante de serie de caracteres con una longitud máxima de 15 bytes, después de que se hayan eliminado los espacios en blanco de cola. El valor debe ser una serie alineada a la izquierda que especifique una de las cinco palabras clave de modalidad de redondeo (no sensibles a las mayúsculas y minúsculas).

### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. El valor de la variable del lenguaje principal debe ser una serie alineada a la izquierda que especifique una de las cinco palabras clave de modalidad de redondeo (no sensibles a las mayúsculas y minúsculas). La longitud real del contenido de *variable-lenguaje-principal* no debe tener más de 15 bytes, después de que se hayan eliminado los espacios en blanco de cola. El valor se debe rellenar a la derecha con espacios de blanco cuando se utilice una variable del lenguaje principal de caracteres de longitud fija. La variable del lenguaje principal no se puede establecer en el valor nulo.

## Normas

- El valor de modalidad de redondeo especificado debe ser el mismo valor que el del registro especial CURRENT DECFLOAT ROUNDING MODE (SQLSTATE 42815).

## Notas

- Esta sentencia no cambia el valor del registro especial CURRENT DECFLOAT ROUNDING MODE en un servidor DB2 para Linux, UNIX y Windows. Sin embargo, cuando la sentencia se procesa en un servidor DB2 para z/OS o un servidor DB2 para i, se puede utilizar para cambiar el valor del registro especial CURRENT DECFLOAT ROUNDING MODE en ese servidor.

## Ejemplo

La sentencia siguiente verifica si el valor de modalidad de redondeo especificado para el cliente coincide con el valor de modalidad de redondeo que está establecido actualmente en el servidor.

```
SET CURRENT DECFLOAT ROUNDING MODE = ROUND_CEILING
```

## 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 la transacción.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### 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 de SQL, de hasta 128 bytes 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 normas 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.

### Ejemplo

Establezca el 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

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```



## SET CURRENT DEGREE

### Notas

- El grado de paralelismo intrapartición para las sentencias de SQL estático puede controlarse utilizando la opción DEGREE del mandato PREP o BIND.
- El grado de ejecución real del paralelismo intrapartición será el más bajo de los siguientes:
  - Parámetro de configuración de grado máximo de consulta (**max\_querydegree**)
  - El grado de ejecución de la aplicación
  - El grado de compilación de la sentencia de SQL
- El parámetro de configuración del gestor de bases de datos **intra\_parallel** debe haberse activado para poder utilizar el paralelismo intrapartición. Si está desactivado, el valor de este registro se pasará por alto y la sentencia no utilizará el paralelismo intrapartición para llevar a cabo la optimización (SQLSTATE 01623).
- El valor del registro especial CURRENT DEGREE y el valor **intra\_parallel** pueden alterarse temporalmente en una carga de trabajo estableciendo el atributo de carga de trabajo MAXIMUM DEGREE.
- Algunas sentencias de SQL no pueden utilizar el paralelismo intrapartición.

### Ejemplos

- *Ejemplo 1:* La siguiente sentencia establece el valor CURRENT DEGREE para que se inhiba el paralelismo intrapartición.  
**SET CURRENT DEGREE = '1'**
- *Ejemplo 2:* La siguiente sentencia establece el valor CURRENT DEGREE para que se 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 el control de la transacción.

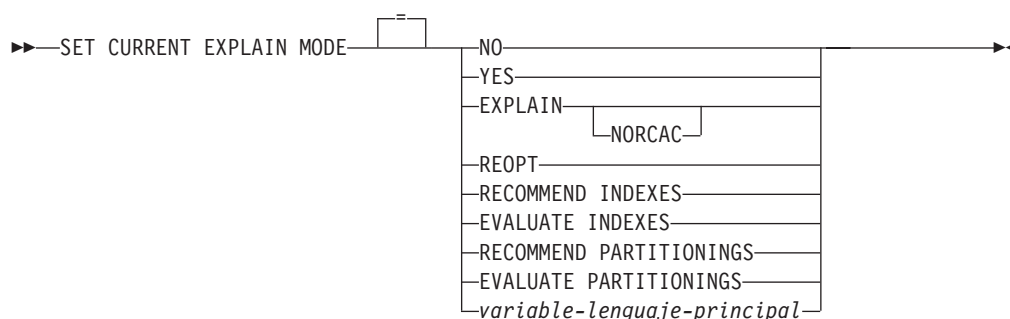
### Invocación

Esta sentencia puede incorporarse 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



### 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 dinámico elegibles. Todas las sentencias de SQL dinámico se compilan y ejecutan normalmente.

#### EXPLAIN

Habilita el recurso Explain y provoca que se capte la información de Explain para cualquier sentencia de SQL dinámico elegible que esté preparada. Sin embargo, no se ejecutan las sentencias dinámicas.

#### EXPLAIN NORCAC

Habilita el recurso Explain y provoca que se capture la información de Explain para cualquier sentencia de SQL dinámico elegible que esté preparada, como si el control de acceso a filas o columnas (RCAC) no se hubiera activado. No se ejecutan las sentencias dinámicas. Cuando se establece esta modalidad explain, el recurso explain explicaría el plan como si el RCAC no estuviera presente.

#### REOPT

Habilita el recurso Explain y hace que la información de Explain se capture para una sentencia de SQL estático o dinámico durante la reoptimización de la sentencia en tiempo de ejecución; es decir, cuando están disponibles los valores de datos reales de las variables del lenguaje principal, los registros especiales, las variables globales o los marcadores de parámetro.

## SET CURRENT EXPLAIN MODE

### RECOMMEND INDEXES

Habilite el compilador SQL para los índices recomendados. Todas las consultas que se ejecutan en esta modalidad de Explain llenarán la tabla `ADVISE_INDEX` con los índices recomendados. Asimismo, la 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 de SQL para evaluar los índices virtuales recomendados para las consultas dinámicas. Las consultas que se ejecutan en esta modalidad de Explain serán compiladas y optimizadas utilizando estadísticas fabricadas basadas en los índices virtuales. No se ejecutan las sentencias. Los índices que deben evaluarse se leen de la tabla `ADVISE_INDEX` si la columna `USE_INDEX` contiene 'Y'. Los índices no exclusivos existentes también se pueden omitir si se establece la columna `USE_INDEX` en 'I' y la columna `EXISTS` en 'Y'. Si se indica una combinación de `USE_INDEX='I'` y `EXISTS='N'`, se continúa realizando normalmente la evaluación de índice para la consulta, pero se omitirá el índice en cuestión.

### RECOMMEND PARTITIONINGS

Especifica que el compilador debe recomendar la mejor partición de base de datos para cada tabla a la que accede una consulta específica. Las mejores particiones de base de datos se graban en una tabla `ADVISE_PARTITIONING`. La consulta no se ejecuta.

### EVALUATE PARTITIONINGS

Especifica que el compilador debe obtener el rendimiento estimado de una consulta utilizando las particiones de base de datos virtuales especificadas en la tabla `ADVISE_PARTITIONING`.

### *variable-lenguaje-principal*

El tipo de datos de la *variable-lenguaje-principal* 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-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

## Notas

- El recurso Explain utiliza los ID siguientes como esquema al calificar tablas de Explain que se están llenando de datos:
  - El ID de autorización de sesión SQL dinámico
  - El ID de autorización de sentencia para SQL estático

El esquema se puede asociar con un conjunto de tablas de Explain o alias que apuntan a un conjunto de tablas de Explain en un esquema distinto. Si no se encuentran tablas de Explain en el esquema, el recurso Explain comprueba la existencia de tablas de Explain en el esquema `SYSTOOLS` e intenta utilizar dichas tablas.

- La información de Explain para las sentencias de SQL estático 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 de `CURRENT EXPLAIN MODE` es `NO`, se capturará la información de Explain para sentencias de SQL dinámico



en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN MODE no es NO, el valor de la opción de enlace **EXPLAIN** se pasará por alto.

- RECOMMEND INDEXES y EVALUATE INDEXES son modalidades especiales que sólo pueden establecerse con la sentencia SET CURRENT EXPLAIN MODE. Estas modalidades no pueden establecerse utilizando opciones **PREP** o **BIND** y no funcionan con la sentencia SET CURRENT EXPLAIN SNAPSHOT.
- Si se activa el recurso Explain, el ID de autorización actual deberá disponer de privilegio INSERT para las tablas de Explain o se generará un error (SQLSTATE 42501).
- Cuando las sentencias de SQL se explican a partir de una rutina, la rutina debe definirse con un indicador de acceso a datos de SQL de MODIFIES SQL DATA (SQLSTATE 42985).
- Si el registro especial se establece en REOPT, y la sentencia de SQL no está cualificada para la reoptimización en tiempo de ejecución (es decir, si la sentencia no tiene variables de entrada o si la opción de vínculo **REOPT** se ha establecido en NONE), no se capturará información de Explain. Si la opción de vinculación **REOPT** se establece en ONCE, la información de Explain sólo se capturará una vez, al reoptimizar inicialmente la sentencia. Después de almacenar la sentencia en antememoria, no se adquirirá más información de Explain para esta sentencia en ejecuciones subsiguientes.
- Si se habilita el recurso Explain, se establece la opción de vínculo **REOPT** en ONCE y se intenta ejecutar una sentencia de SQL que ya está en antememoria, la sentencia se compilará y reoptimizará con los valores actuales de las variables de entradas y las tablas de Explain se rellenarán de acuerdo con ello. El plan de acceso que se acaba de generar para esta sentencia no se almacenará en antememoria ni se ejecutará. Las demás aplicaciones que ejecutan simultáneamente esta sentencia almacenada en antememoria continuarán con su ejecución y las nuevas peticiones de ejecución de esta sentencia tomarán el plan de acceso que ya está en la antememoria.
- El valor de REOPT para los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT alterará temporalmente el valor de las opciones de vinculación **EXPLAIN** y **EXPLSNAP** en tiempo de vinculación si una sentencia de SQL dinámico o estático tiene variables de entrada y la opción de vinculación **REOPT** se establece en ONCE o ALWAYS.
- El control de acceso de nivel de fila y de columna (RCAC) definido en las tablas de EXPLAIN se aplica para el acceso del usuario a estas tablas, al igual que cualquier otra tabla normal. Sin embargo, el control de acceso de nivel de fila y de columna en las tablas de EXPLAIN no se aplica cuando la misma base de datos DB2 está llenando estas tablas de EXPLAIN. Se considera mantenimiento interno y no está sujeto al RCAC, de forma muy parecida al SQL interno.

### Ejemplo

El ejemplo siguiente establece el registro especial CURRENT EXPLAIN MODE, de modo que se capturará información de Explain para cualquier sentencia de SQL dinámico que pueda elegirse posteriormente 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.

La sentencia no está bajo el control de la transacción.

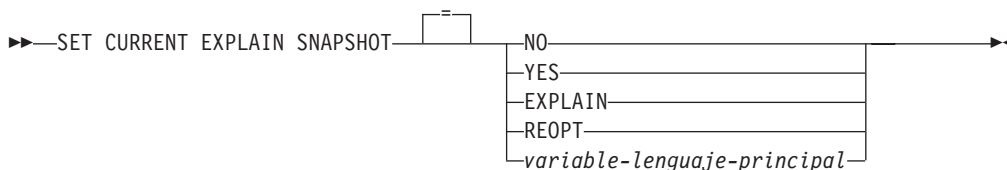
### Invocación

Esta sentencia puede incorporarse 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



### 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 dinámico elegible. Esta información se inserta en la columna SNAPSHOT de la tabla EXPLAIN\_STATEMENT.

#### EXPLAIN

Habilita el recurso de instantánea de Explain, creando una instantánea de la representación interna de cada sentencia de SQL dinámico elegible que se prepara. Sin embargo, no se ejecutan las sentencias dinámicas.

#### REOPT

Habilita el recurso Explain y hace que la información de Explain se capture para una sentencia de SQL estático o dinámico durante la reoptimización de la sentencia en tiempo de ejecución; es decir, cuando están disponibles los valores actuales de las variables del lenguaje principal, los registros especiales, las variables globales o los marcadores de parámetro.

#### *variable-lenguaje-principal*

El tipo de datos de la *variable-lenguaje-principal* debe ser CHAR o VARCHAR y la longitud de su contenido no debe ser mayor que 8. Si se proporciona un campo más largo, se devolverá 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 permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una

*variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

### Notas

- El recurso Explain utiliza los ID siguientes como esquema al calificar tablas de Explain que se están llenando de datos:
  - El ID de autorización de sesión SQL dinámico
  - El ID de autorización de sentencia para SQL estático

El esquema se puede asociar con un conjunto de tablas de Explain o alias que apuntan a un conjunto de tablas de Explain en un esquema distinto. Si no se encuentran tablas de Explain en el esquema, el recurso Explain comprueba la existencia de tablas de Explain en el esquema SYSTOOLS e intenta utilizar dichas tablas.

- Las instantáneas de Explain para las sentencias de SQL estático 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 de registro CURRENT EXPLAIN SNAPSHOT es NO, se capturarán instantáneas de Explain para las sentencias de SQL dinámico en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN SNAPSHOT no es NO, la opción EXPLSNAP se pasará por alto.
- 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).
- Cuando las sentencias de SQL se explican a partir de una rutina, la rutina debe definirse con un indicador de acceso a datos de SQL de MODIFIES SQL DATA (SQLSTATE 42985).
- Si el registro especial se establece en REOPT, y la sentencia de SQL no está calificada para la reoptimización en tiempo de ejecución (es decir, si la sentencia no tiene variables de entrada o si la opción de vínculo REOPT se ha establecido en NONE), no se capturará información de Explain. Si la opción de vinculación REOPT se establece en ONCE, la información de instantánea de Explain sólo se capturará una vez, al reoptimizar inicialmente la sentencia. Después de almacenar la sentencia en antememoria, no se adquirirá más información de Explain para esta sentencia en ejecuciones subsiguientes.
- Si se habilita el recurso Explain, se establece la opción de vinculación REOPT en ONCE y se intenta ejecutar una sentencia de SQL reoptimizable que ya está almacenada en antememoria, la sentencia se compilará y reoptimizará con los valores actuales de las variables de entrada y la instantánea de Explain se capturará de acuerdo a ello. El plan de acceso que se acaba de generar para esta sentencia no se almacenará en antememoria ni se ejecutará. Las demás aplicaciones que ejecutan simultáneamente esta sentencia almacenada en antememoria continuarán con su ejecución y las nuevas peticiones de ejecución de esta sentencia tomarán el plan de acceso que ya está en la antememoria.
- El valor de REOPT para los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT alterará temporalmente el valor de las opciones de vinculación EXPLAIN y EXPLSNAP en tiempo de vinculación si una sentencia de SQL estático o dinámico tiene variables de entrada y la opción de vinculación REOPT se establece en ONCE o ALWAYS.

## SET CURRENT EXPLAIN SNAPSHOT

### Ejemplos

- *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 dinámico 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 FEDERATED ASYNCHRONY

La sentencia SET CURRENT FEDERATED ASYNCHRONY asigna un valor al registro especial CURRENT FEDERATED ASYNCHRONY.

La sentencia no está bajo el control de la transacción.

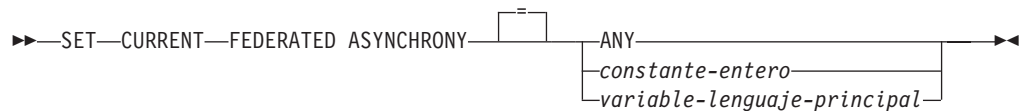
### Invocación

La sentencia se puede incorporar a un programa de aplicación o emitir a través del uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### ANY

Especifica el valor -1 para CURRENT FEDERATED ASYNCHRONY, lo que significa que la ejecución de sentencias puede implicar asincronía al utilizar un grado determinado por el gestor de bases de datos.

#### constante-entera

Especifica un valor de entero entre 0 y 32.767, incluido. La ejecución de sentencias puede implicar asincronía al utilizar el grado especificado. Si el valor es 0 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esa sentencia no utilizará la asincronía.

#### variable-lenguaje-principal

Una variable de tipo INTEGER. El valor debe estar entre 0 y 32.767, incluido, o -1 (representa ANY). Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

### Notas

- El grado de asincronía para las sentencias de SQL estático se puede controlar utilizando la opción FEDERATED\_ASYNCCHRONY del mandato PREP o BIND.
- El valor inicial del registro especial CURRENT FEDERATED ASYNCHRONY lo determina el parámetro **asincronía\_federada** de configuración del gestor de bases de datos si la sentencia dinámica se emite a través del procesador de línea de mandatos (CLP). La opción de vinculación FEDERATED\_ASYNCCHRONY determina el valor inicial si la sentencia dinámica forma parte de una aplicación que se está vinculando.

### Ejemplos

- *Ejemplo 1:* La sentencia siguiente inhabilita la asincronía, estableciendo en 0 el valor del registro especial CURRENT FEDERATED ASYNCHRONY.

## SET CURRENT FEDERATED ASYNCHRONY

**SET CURRENT FEDERATED ASYNCHRONY = 0**

- *Ejemplo 2:* La sentencia siguiente establece en 5 el grado de asincronía.

**SET CURRENT FEDERATED ASYNCHRONY 5**

- *Ejemplo 3:* La sentencia siguiente establece en -1 el valor del registro especial **CURRENT FEDERATED ASYNCHRONY**, lo que especifica que el gestor de bases de datos debe determinar el grado de asincronía.

**SET CURRENT FEDERATED ASYNCHRONY ANY**

## SET CURRENT IMPLICIT XMLPARSE OPTION

La sentencia SET CURRENT IMPLICIT XMLPARSE OPTION cambia el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION.

La sentencia no está bajo el control de la transacción.

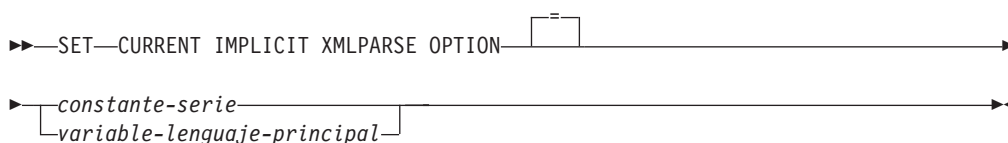
### Invocación

Esta sentencia puede incorporarse 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



### Descripción

#### *constante-serie*

Constante de serie de caracteres. El valor debe ser una serie alineada a la izquierda que sea 'PRESERVE WHITESPACE' o 'STRIP WHITESPACE' (no sensible a las mayúsculas y minúsculas) sin caracteres en blanco adicionales entre las palabras clave.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. El valor de la variable del lenguaje principal debe ser una serie alineada a la izquierda que sea 'PRESERVE WHITESPACE' o 'STRIP WHITESPACE' (no sensible a las mayúsculas y minúsculas) sin caracteres en blanco adicionales entre las palabras clave. El valor debe rellenarse con blancos por la derecha utilizando una *variable-lenguaje-principal* de carácter de longitud fija. La variable del lenguaje principal no puede ser nulo.

### Notas

- El valor inicial del registro especial CURRENT IMPLICIT XMLPARSE OPTION es 'STRIP WHITESPACE'.
- Las sentencias de SQL dinámico y estático se ven afectadas por este registro especial.

### Ejemplo

Establezca el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION en 'PRESERVE WHITESPACE'.

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

---

# SET CURRENT ISOLATION

La sentencia SET CURRENT ISOLATION asigna un valor al registro especial CURRENT ISOLATION.

La sentencia no está bajo el control de la transacción.

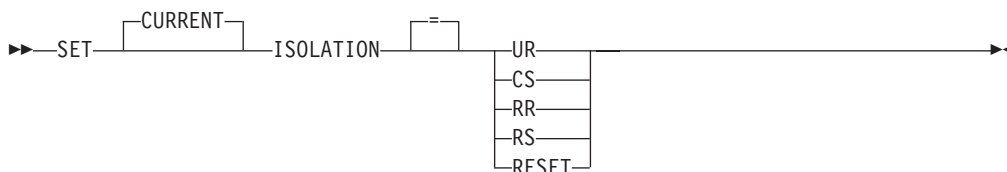
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

El valor del registro especial CURRENT ISOLATION se sustituye por el valor especificado o se establece en blancos si se especifica RESET.

### Notas

- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos.
  - TO puede especificarse en lugar del signo de igual (=)
  - DIRTY READ puede especificarse en lugar de UR
  - READ UNCOMMITTED puede especificarse en lugar de UR
  - READ COMMITTED se reconoce y se actualiza a CS
  - CURSOR STABILITY puede especificarse en lugar de CS
  - REPEATABLE READ puede especificarse en lugar de RR
  - SERIALIZABLE puede especificarse en lugar de RR



## SET CURRENT LOCALE LC\_MESSAGES

La sentencia SET CURRENT LOCALE LC\_MESSAGES cambia el valor del registro especial CURRENT LOCALE LC\_MESSAGES.

La sentencia no está bajo el control de la transacción.

### Invocación

Esta sentencia puede incorporarse 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

►► SET CURRENT LOCALE LC\_MESSAGES  *variable-lenguaje-principal*  *constante-serie* ►►

### Descripción

El registro especial CURRENT LOCALE LC\_MESSAGES identifica el entorno local que utiliza EVMON\_UPGRADE\_TABLES, así como al supervisar las rutinas en el módulo **monreport**. EVMON\_UPGRADE\_TABLES y las rutinas de supervisión utilizan el valor de CURRENT LOCALE LC\_MESSAGES para determinar el idioma en el que la salida de texto del conjunto de resultados de las rutinas se debe devolver. Las rutinas definidas por el usuario que se codifican para devolver mensajes también pueden utilizar el valor de CURRENT LOCALE LC\_MESSAGES para determinar el idioma que se debe utilizar para el texto del mensaje.

*variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. No puede establecerse en nulo.

*constante-serie*

Constante de serie de caracteres.

### Notas

- **Valor inicial:** el valor inicial del registro especial CURRENT LOCALE LC\_MESSAGES es 'en\_US'.
- **Disponibilidad de idioma:** si el idioma del entorno local no está disponible para el gestor de bases de datos de DB2, los mensajes se devolverán en inglés.
- **Compatibilidad de páginas de códigos:** el idioma del entorno local especificado debe recibir soporte de la página de códigos del parámetro de salida o el tipo de retornos de una rutina que utilice el registro especial para determinar el idioma en el que se devuelve la información de texto de los mensajes. Si la base de datos no es Unicode (y la rutina no se ha creado con PARAMETER CCSID UNICODE) y algunos caracteres del idioma del entorno local no pueden representarse en la página de códigos de la base de datos, se devolverán caracteres sustitutos como resultado de la conversión de la página de códigos.

## SET CURRENT LOCALE LC\_MESSAGES

- *Uso potencial futuro:* en un futuro release, el valor del registro especial CURRENT LOCALE LC\_MESSAGES podrá utilizarse para otras áreas del entorno de base de datos que impliquen mensajes.
- *Entornos locales válidos y denominaciones:* para obtener información sobre los entornos locales válidos y su denominación, consulte la sección sobre nombres de entorno local para SQL y XQuery en la publicación *Globalization Guide*.

### Ejemplos

- *Ejemplo 1:* la sentencia siguiente establece el registro especial CURRENT LOCALE LC\_MESSAGES en el entorno local inglés de Canadá utilizando la versión más reciente del Common Locale Data Repository (CLDR) disponible en el gestor de bases de datos de DB2.

```
SET CURRENT LOCALE LC_MESSAGES = 'en_CA'
```

- *Ejemplo 2:* la sentencia siguiente establece el registro especial CURRENT LOCALE LC\_MESSAGES en el entorno local francés de Francia utilizando Common Locale Data Repository (CLDR) versión 1.5. La rutina CONNECTION del módulo **monreport** se invoca, a continuación, para que la salida se devuelva en francés.

```
SET CURRENT LOCALE LC_MESSAGES = 'CLDR 1.5:fr_FR'
CALL MONREPORT.CONNECTION
```

- *Ejemplo 3:* supongamos que el procedimiento definido por el usuario XYZ.STORELOCATOR adopta una entrada de código postal. Devuelve un conjunto de resultados de tiendas de la empresa XYZ a una distancia de 30 minutos en coche del código postal que se ha entrado. Si el código postal no tiene el formato correcto, se devuelve un mensaje de error que indica que el problema se debe al formato. El procedimiento se codifica para poder devolver el mensaje de error en el idioma determinado a partir del valor del registro especial CURRENT LOCALE LC\_MESSAGES. La sentencia siguiente establece el registro especial CURRENT LOCALE LC\_MESSAGES en el entorno local español de México. El procedimiento definido por el usuario de localizador de tiendas se invoca, a continuación, y los mensajes de error se devuelven en español.

```
SET CURRENT LOCALE LC_MESSAGES = 'es_MX'
CALL XYZ.STORELOCATOR(:ZIP, :STATUSMSG)
```

## SET CURRENT LOCALE LC\_TIME

La sentencia SET CURRENT LOCALE LC\_TIME cambia el valor del registro especial CURRENT LOCALE LC\_TIME. No está bajo el control de la transacción.

### Invocación

Esta sentencia puede incorporarse 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

```

▶▶ SET CURRENT LOCALE LC_TIME [=] {
 variable-lenguaje-principal
 | constante-serie
}

```

### Descripción

El registro especial CURRENT LOCALE LC\_TIME es utilizado por las funciones DAYNAME, MONTHNAME, NEXT\_DAY, ROUND, ROUND\_TIMESTAMP, TIMESTAMP\_FORMAT, TRUNCATE, TRUNC\_TIMESTAMP y VARCHAR\_FORMAT cuando no se especifica de forma explícita el argumento *nombre-entorno-local*.

*variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. No puede establecerse en nulo.

*constante-serie*

Constante de serie de caracteres.

### Notas

- **Valor inicial:** El valor inicial del registro especial CURRENT LOCALE LC\_TIME es 'en\_US'.
- **Uso potencial futuro:** en un futuro release, el valor del registro especial CURRENT LOCALE LC\_TIME podrá utilizarse por otras funciones escalares y para otras áreas del entorno de base de datos que impliquen valores de fecha y hora.
- **Entornos locales válidos y denominaciones:** Para obtener información sobre los entornos locales válidos y su denominación, consulte la sección sobre nombres de entorno local para SQL y XQuery en la publicación *Globalization Guide* .

### Ejemplos

- *Ejemplo 1:* la sentencia siguiente establece el registro especial CURRENT LOCALE LC\_TIME en el entorno local inglés de Canadá y utiliza la versión más reciente del CLDR (Common Locale Data Repository) disponible en el gestor de bases de datos DB2.

```
SET CURRENT LOCALE LC_TIME = 'en_CA'
```

- *Ejemplo 2:* la sentencia siguiente establece el registro especial CURRENT LOCALE LC\_TIME en el entorno local francés de Francia y utiliza el CLDR

## SET CURRENT LOCALE LC\_TIME

(Common Locale Data Repository) versión 1.8.1. La función escalar MONTHNAME se invoca, pues, con un solo argumento de '2008-11-10-00.00.00.000000'.

```
SET CURRENT LOCALE LC_TIME = 'CLDR181_fr_FR'
VALUES MONTHNAME('2008-11-10-00.00.00.000000')
```

devuelve:

'novembre'

## SET CURRENT LOCK TIMEOUT

La sentencia SET CURRENT LOCK TIMEOUT cambia el valor del registro especial CURRENT LOCK TIMEOUT.

La sentencia no está bajo el control de la transacción.

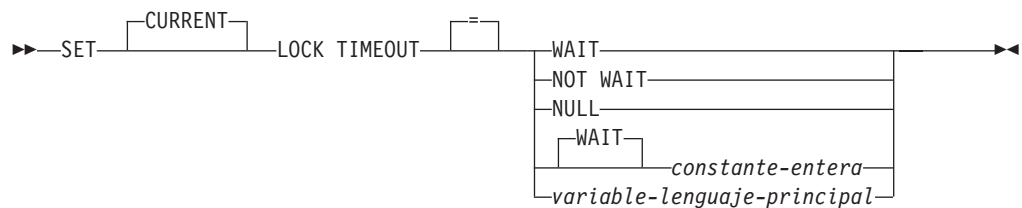
### Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

El valor especificado debe ser un entero entre -1 y 32767, inclusive (SQLSTATE 428B7) o el valor nulo.

#### WAIT

Especifica un valor CURRENT LOCK TIMEOUT de -1, lo que significa que el gestor de bases de datos debe esperar hasta que se libere el bloqueo o se detecte un punto muerto (SQLSTATE 40001 o 57033).

#### NOT WAIT

Especifica un valor de CURRENT LOCK TIMEOUT de 0, lo que significa que el gestor de bases de datos no debe esperar los bloqueos que no se pueden obtener y se devolverá un error (SQLSTATE 40001 o 57033).

#### NULL

Especifica que se debe eliminar el valor establecido de CURRENT LOCK TIMEOUT y que el valor del parámetro de configuración de base de datos **locktimeout** se debe utilizar cuando se espera un bloqueo. El valor que se devuelve para el registro especial cambiará cuando cambie el valor de **locktimeout**.

#### WAIT *constante-entera*

Especifica un valor de entero entre -1 y 32767. Un valor de -1 es equivalente a especificar la palabra clave WAIT sin un valor entero. Un valor de 0 es equivalente a especificar la cláusula NOT WAIT. Si el valor está entre 1 y 32767, el gestor de bases de datos esperará ese número de segundos (si no se puede obtener un bloqueo) antes de devolver un error (SQLSTATE 40001 o 57033).

## SET CURRENT LOCK TIMEOUT

### *variable-lenguaje-principal*

Una variable de tipo INTEGER. El valor debe estar entre -1 y 32767. Si *variable-lenguaje-principal* tiene una variable de indicador asociada y su valor especifica un valor nulo, se elimina el valor establecido de CURRENT LOCK TIMEOUT. Es equivalente a especificar la palabra clave NULL.

### Notas

- Un valor actualizado del registro especial surte efecto inmediatamente después de la ejecución satisfactoria de esta sentencia. Puesto que el valor de registro especial que se debe utilizar durante la ejecución de la sentencia se fija al principio de la ejecución de la sentencia, sólo devolverán un valor actualizado del registro especial CURRENT LOCK TIMEOUT para las sentencias que inician la ejecución después de que la sentencia SET LOCK TIMEOUT se haya completado satisfactoriamente.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con los productos de base de datos de Informix. Estas alternativas no son estándar y no deberían utilizarse.
  - Se puede especificar MODE en lugar de TIMEOUT.
  - Se puede especificar TO en lugar del operador igual (=).

### Ejemplos

- *Ejemplo 1:* Establecer el valor de tiempo de espera excedido para que espere 30 segundos antes de devolver un error.  

```
SET CURRENT LOCK TIMEOUT 30
```
- *Ejemplo 2:* Eliminar el valor establecido de tiempo de espera excedido de bloqueo para que se utilice el valor del parámetro de configuración de bases de datos **locktimeout** en su lugar.  

```
SET CURRENT LOCK TIMEOUT NULL
```

## SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

La sentencia SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION cambia el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

La sentencia no está bajo el control de la transacción.

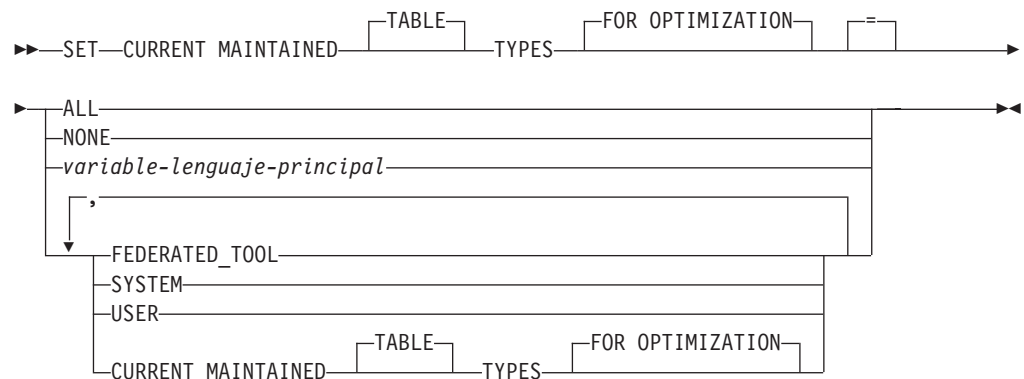
### Invocación

Esta sentencia puede incorporarse 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



### Descripción

#### ALL

Especifica que todos los tipos posibles de tablas mantenidas que este registro especial controla, ahora y en el futuro, van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

#### NONE

Especifica que ninguno de los tipos de objetos que este registro especial controla van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

#### FEDERATED\_TOOL

Especifica que las tablas de consulta materializada de renovación diferida que una herramienta federada mantiene pueden tomarse en consideración para optimizar el proceso de consultas de SQL dinámico, siempre que el valor del registro especial CURRENT QUERY OPTIMIZATION sea 2 o mayor que 5.

#### SYSTEM

Especifica que las tablas de consulta materializada con renovación diferida mantenidas por el sistema pueden considerarse para la optimización del proceso de las consultas de SQL dinámico. (Las tablas de consulta materializada inmediatas siempre están disponibles.)

## SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

### USER

Especifica que las tablas de consulta materializada con renovación diferida mantenidas por el usuario pueden considerarse para la optimización del proceso de las consultas de SQL dinámico.

### CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

El valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION antes de que se ejecute esta sentencia.

### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la variable del lenguaje principal no debe exceder de 254 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lenguaje-principal* deben alinearse a la izquierda. El contenido de la *variable-lenguaje-principal* debe ser una serie que forme una lista de palabras clave separadas por comas y que coincidan con lo que se puede especificar como palabras claves para el registro especial. Estas palabras clave se deben especificar exactamente en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas. El valor deberá rellenarse con blancos por la derecha si su longitud es menor que la de la variable del lenguaje principal.

### Notas

- El valor inicial del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION es SYSTEM.
- El registro especial CURRENT REFRESH AGE debe establecerse en un valor distinto de cero para que los tipos de tablas que se han especificado se consideren cuando se optimice el proceso de las consultas de SQL dinámico.

### Ejemplos

- *Ejemplo 1:* Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = SYSTEM, USER
```

- *Ejemplo 2:* Recupere el valor actual del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION en una variable del lenguaje principal denominada CURMAINTYPES.

```
EXEC SQL VALUES (CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION)
INTO :CURMAINTYPES
```

- *Ejemplo 3:* Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION de modo que no tenga ningún valor.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = NONE
```



## SET CURRENT MDC ROLLOUT MODE

La sentencia SET CURRENT MDC ROLLOUT MODE asigna un valor al registro especial CURRENT MDC ROLLOUT MODE. El valor especifica el tipo de borrado de lanzamiento que va a realizarse sobre sentencias DELETE de calificación para tablas de clúster multidimensional (MDC).

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis

```

▶▶ SET CURRENT MDC ROLLOUT MODE {
 NONE
 IMMEDIATE
 DEFERRED
 variable-lenguaje-principal
}

```

### Descripción

#### NONE

Especifica que no va a utilizarse la optimización de clasificación de MDC durante las operaciones de supresión. La sentencia DELETE se procesa del mismo modo que una sentencia DELETE que no se califica para la clasificación.

#### IMMEDIATE

Especifica que la optimización de clasificación de MDC va a utilizarse si se califica la sentencia DELETE. Si la tabla tiene índices RID, los índices se actualizan inmediatamente durante el proceso de supresión. Los bloques suprimidos pueden volver a utilizar después de que se confirme la transacción.

#### DEFERRED

Especifica que la optimización de clasificación de MDC va a utilizarse si se califica la sentencia DELETE. Si la tabla tiene índices RID, las actualizaciones de los índices se difieren hasta que hayan acabado las confirmaciones de las transacciones. Con esta opción, la supresión de procesos es más rápida y utiliza menos espacio de archivo de anotaciones, pero los bloques suprimidos no pueden volver a utilizarse hasta que se completen las actualizaciones de índice.

#### *variable-lenguaje-principal*

Una variable de tipo VARCHAR. La longitud *variable-lenguaje-principal* debe ser igual o inferior a 17 bytes (SQLSTATE 42815). El valor de la variable del lenguaje principal debe ser una serie alineada a la izquierda, es decir una serie 'NONE', 'IMMEDIATE' o 'DEFERRED' (no sensible a las mayúsculas y minúsculas). Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

## SET CURRENT MDC ROLLOUT MODE

### Notas

- Las sucesivas sentencias DELETE que pueden elegirse para el proceso de lanzamiento respecto del valor del registro especial CURRENT MDC ROLLOUT MODE. En la actualidad las sesiones en ejecución no resultan afectadas por un cambio en este registro especial.
- Los efectos de la ejecución de la sentencia SET CURRENT MDC ROLLOUT MODE no se retrotraen en el caso de que se retrotraiga la unidad de trabajo en la que se ejecuta la sentencia.
- En DB2 Versión 9.7 y releases posteriores, la modalidad DEFERRED no recibe soporte en una tabla MDC particionada de datos con índices RID particionados. Sólo reciben soporte las modalidades NONE e IMMEDIATE. El tipo de despliegue de supresión de limpieza será IMMEDIATE si la variable de registro **DB2\_MDC\_ROLLOUT** se establece en DEFER o si el registro especial CURRENT MDC ROLLOUT MODE se establece en DEFERRED para alterar temporalmente el valor de **DB2\_MDC\_ROLLOUT**.

Si sólo existen índices RID no particionados en la tabla MDC, el despliegue de supresión diferido de limpieza de índice recibe soporte.

### Ejemplo

Especifique comportamiento de borrado diferido para la siguiente sentencia DELETE que se califique para el proceso de lanzamiento.

```
SET CURRENT MDC ROLLOUT MODE IMMEDIATE
```

## SET CURRENT OPTIMIZATION PROFILE

La sentencia SET CURRENT OPTIMIZATION PROFILE asigna un valor al registro especial CURRENT OPTIMIZATION PROFILE. El valor especifica el perfil de optimización que el optimizador debe utilizar al preparar sentencias de DML dinámico.

La sentencia no está bajo el control de la transacción.

Cuando se evalúa la sentencia, se comprueba la validez del nombre del perfil de optimización, pero el perfil no se procesa hasta que el optimizador se encuentra con una sentencia de DML dinámico.

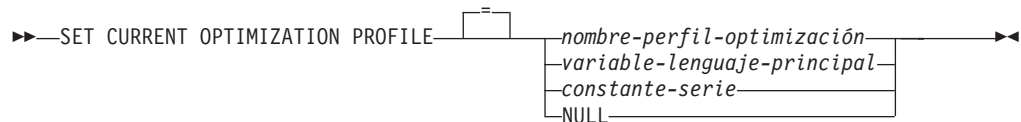
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### *nombre-perfil-optimización*

Es el nombre de dos parte del perfil de optimización. El nombre se puede especificar con un literal, una variable del lenguaje principal o un registro especial. El nombre especificado es el nombre que se entra en el registro especial CURRENT OPTIMIZATION PROFILE.

Si el nombre-perfil-optimización especificado no está calificado, se utilizará el valor del registro CURRENT DEFAULT SCHEMA como calificador implícito. El valor por omisión del registro especial es nulo.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR que incluye el nombre del perfil de optimización. Una variable del lenguaje principal que incluye un indicador nulo indica que se debe utilizar el valor de la opción de enlace OPTPROFILE si dicho valor se especifica para el paquete actual. Una variable del lenguaje principal de longitud cero, o con sólo un espacio en blanco, indica que no se debe utilizar ningún perfil de optimización.

La variable del lenguaje principal debe cumplir las especificaciones siguientes:

- El contenido de la serie es un identificador de una parte o de dos partes (separado por un punto), sin espacios en blanco iniciales.
- El identificador o los identificadores pueden ser delimitados o no delimitados.
- El contenido de la serie no se convierte a mayúsculas.

## SET CURRENT OPTIMIZATION PROFILE

- En las series no delimitadas no se pueden utilizar caracteres en minúsculas ni caracteres especiales.
- Si el primer carácter es una comilla doble, deberá haber una comilla doble de cierre antes de un punto o como último carácter (exceptuando los espacios en blanco) de la serie.
- Si el primer carácter a continuación de un punto es una comilla doble, deberá haber una comilla doble de cierre como último carácter (exceptuando los espacios en blanco) de la serie.
- Si el identificador está delimitado, para incluir comillas dobles en el identificador se deberá especificar el carácter dos veces.
- Cualquier punto que no esté dentro de un identificador delimitado se tratará como un separador y sólo puede existir un punto separador en la serie.

### *constante-serie*

Especifica una constante como serie de caracteres del nombre del perfil de optimización. El contenido de una constante de tipo serie debe cumplir las mismas especificaciones que una variable del lenguaje principal.

### **NULL**

Establece en nulo el registro CURRENT OPTIMIZATION PROFILE.

La Tabla 35 proporciona ejemplos de literales de serie e identificadores que se pueden utilizar para asignar el registro de acuerdo con las normas de denominación del perfil de optimización. El valor de la columna SCHEMA y NAME representa un nombre de perfil de optimización tal como aparecería en la tabla OPT\_PROFILE. La columna de Literales de serie válidos muestra los literales de serie que coinciden con el perfil de optimización al que dan nombre los correspondientes valores de la columna SCHEMA y NAME. La columna Identificadores válidos muestra los identificadores que identificarían este mismo perfil de optimización.

*Tabla 35. Ejemplos de literales de serie e identificadores*

| SCHEMA | NAME        | Literales de serie válidos                                           | Identificadores válidos                                          |
|--------|-------------|----------------------------------------------------------------------|------------------------------------------------------------------|
| SIMMEN | BIG_PROF    | 'BIG_PROF'<br>'SIMMEN.BIG_PROF'<br>"BIG_PROF"<br>"SIMMEN"."BIG_PROF" | BIG_PROF<br>SIMMEN.BIG_PROF<br>"BIG_PROF"<br>"SIMMEN"."BIG_PROF" |
| SIMMEN | low_profile | "low_profile"<br>'SIMMEN."low_profile"<br>"SIMMEN"."low_profile"     | "low_profile"<br>SIMMEN."low_profile"<br>"SIMMEN"."low_profile"  |
| eliaz  | DBA3        | 'DBA3'<br>"DBA3"<br>"eliaz".DBA3'<br>"eliaz"."DBA3"                  | DBA3<br>"eliaz".DBA3<br>"eliaz"."DBA3"                           |
| SNOW   | PROFILE1.0  | "PROFILE1.0"<br>'SNOW."PROFILE1.0"<br>"SNOW"."PROFILE1.0"            | "PROFILE1.0"<br>SNOW."PROFILE1.0"<br>"SNOW"."PROFILE1.0"         |

## Notas

- Si el valor del registro especifica el nombre de un perfil de optimización existente, el perfil de optimización especificado se utilizará al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es nulo, especifica el nombre de un perfil de optimización existente, el perfil de optimización especificado por la opción de enlace OPTPROFILE, si existe, se utilizará al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es nulo y no se ha establecido la opción de enlace OPTPROFILE, no se utilizará ningún perfil de optimización al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es una serie vacía, no se utilizará ningún perfil de optimización al preparar posteriores sentencias de DML dinámico, independientemente de si la opción de enlace OPTPROFILE se ha establecido o no.
- Los cambios posteriores de CURRENT DEFAULT SCHEMA no afectan al perfil de optimización. El valor de registro CURRENT OPTIMIZATION PROFILE se establece con el nombre de dos partes que está en vigor al mismo tiempo que se evalúa la sentencia SET CURRENT OPTIMIZATION PROFILE. Sólo otra sentencia SET CURRENT OPTIMIZATION PROFILE puede cambiar el perfil de optimización que se utiliza.

## Ejemplos

- *Ejemplo 1:* El perfil de optimización RICK.FOO se utiliza para las sentencias 1, 2 y 3. TOM.FOO se utiliza para la sentencia 4.

```

SET CURRENT SCHEMA = 'RICK'
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
sentencia 1
sentencia 2
SET CURRENT SCHEMA = 'TOM'
sentencia 3
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
sentencia 4

```

- *Ejemplo 2:* Una aplicación con las sentencias siguientes se ha enlazado con las opciones OPTPROFILE("Foo") y QUALIFIER("John"). El perfil de optimización KAAREL.BAR se utiliza para la sentencia 1 y el perfil de optimización "John"."Foo" se utiliza para la sentencia 2.

```

SET CURRENT SCHEMA = 'KAAREL'
SET CURRENT OPTIMIZATION PROFILE = 'BAR'
sentencia 1
SET CURRENT SCHEMA = "Tom"
SET CURRENT OPTIMIZATION PROFILE NULL
sentencia 2

```

- *Ejemplo 3:* La serie vacía es un valor especial que indica que no se debe utilizar ningún perfil de optimización. El perfil de optimización "Hamid"."Foo" se utiliza para la sentencia 1 y no se utiliza ningún perfil de optimización para la sentencia 2.

```

SET CURRENT OPTIMIZATION PROFILE = '"Hamid"."Foo"'
sentencia 1
SET CURRENT OPTIMIZATION PROFILE = ''
sentencia 2

```

### SET CURRENT PACKAGE PATH

La sentencia SET CURRENT PACKAGE PATH asigna un valor al registro especial CURRENT PACKAGE PATH.

La sentencia no está bajo el control de la transacción.

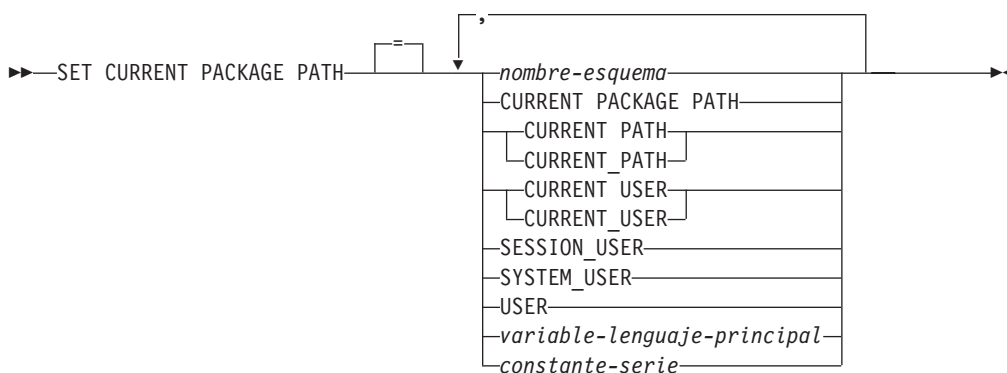
#### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

#### Autorización

No se necesita.

#### Sintaxis



#### Descripción

##### *nombre-esquema*

Identifica un esquema. El nombre no debe ser un identificador delimitado que esté vacío o que sólo contenga espacios en blanco (SQLSTATE 42815).

##### **CURRENT PACKAGE PATH**

El valor del registro especial CURRENT PACKAGE PATH antes de que se ejecute esta sentencia.

##### **CURRENT PATH**

El valor del registro especial CURRENT PATH.

##### **CURRENT\_USER**

El valor del registro especial CURRENT\_USER.

##### **SESSION\_USER**

El valor del registro especial SESSION\_USER.

##### **SYSTEM\_USER**

El valor del registro especial SYSTEM\_USER.

##### **USER**

El valor del registro especial USER.

##### *variable-lenguaje-principal*

Contiene uno o varios nombres de esquema, separados mediante comas. La variable del lenguaje principal debe:

- Ser una variable de serie de caracteres (CHAR o VARCHAR). La longitud real del contenido de la variable del lenguaje principal no debe exceder de la longitud del registro especial CURRENT PACKAGE PATH.
- No ser un valor nulo. Si se proporciona una variable de indicador, su valor no debe indicar un valor nulo.
- Contener una serie vacía o en blanco o uno o varios nombres de esquema separados por comas.
- Rellenarse a la derecha con espacios en blanco si la longitud real de la variable del lenguaje principal es mayor que el contenido.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT\_PATH, CURRENT USER, CURRENT\_USER, SESSION\_USER, SYSTEM\_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío o que contenga sólo espacios en blanco.

### *constante-serie*

Especifica una constante de serie de caracteres que contiene cero, uno o varios nombres de esquema que están separados por comas. La constante de serie debe:

- Tener una longitud que no exceda de la longitud máxima del registro especial CURRENT PACKAGE PATH.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT\_PATH, CURRENT USER, CURRENT\_USER, SESSION\_USER, SYSTEM\_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío o que contenga sólo espacios en blanco.

## Normas

- Si el mismo esquema aparece más de una vez en la lista, se utiliza la primera ocurrencia del esquema (SQLSTATE 01625).
- El número de esquemas que se pueden especificar está limitado por la longitud total del registro especial CURRENT PACKAGE PATH. La serie de registro especial se crea tomando el nombre del esquema especificado y eliminando los blancos de cola, delimitando el nombre con comillas dobles y separando los nombres de esquema con comas. La longitud de la lista resultante no puede exceder de la longitud máxima del registro especial (SQLSTATE 0E000).
- Un nombre de esquema que no se ajuste a las normas para un identificador normal (por ejemplo, un nombre de esquema que contenga caracteres en minúsculas o caracteres que no se pueden especificar en un identificador normal), se debe especificar como un nombre de esquema delimitado y no se debe especificar en una variable del lenguaje principal ni constante de serie.
- Para indicar que el valor actual de un registro especial (especificado como una sola palabra clave) se debe utilizar en la vía de acceso del paquete, especifique el nombre del registro especial como palabra clave. Si el nombre del registro especial se especifica como identificador delimitado en su lugar (por ejemplo, "USER"), se interpreta como un nombre de esquema de ese valor ('USER').
- Las normas siguientes se utilizan para determinar si un valor especificado en una sentencia SET CURRENT PACKAGE PATH es una variable o un nombre de esquema:
  - Si el *nombre* es igual que un parámetro o variable de SQL en el procedimiento de SQL, *nombre* se interpreta como parámetro o variable de SQL y el valor de *nombre* se asigna a la vía de acceso del paquete.

## SET CURRENT PACKAGE PATH

- Si *nombre* no es igual que un parámetro o una variable de SQL del procedimiento de SQL, *nombre* se interpreta como un nombre de esquema y el valor de *nombre* se asigna a la vía de acceso del paquete.

### Notas

- **Consideraciones de la transacción:** La sentencia SET CURRENT PACKAGE PATH no es una operación que se pueda confirmar. ROLLBACK no tiene ningún efecto en el registro especial CURRENT PACKAGE PATH.
- **Comprobación de existencia de esquemas:** No se hace ninguna validación de que existan los esquemas especificados en el momento en que se establece el registro especial CURRENT PACKAGE PATH. Por ejemplo, un esquema que no está bien escrito no se detecta, lo que podría afectar a la manera en que funciona el SQL subsiguiente. En tiempo de ejecución de paquetes, se comprueba la autorización para un paquete coincidente y si la comprobación de autorización falla, se devuelve un error (SQLSTATE 42501).
- **Contenido de la variable del lenguaje principal o constante de serie:** El contenido de una variable del lenguaje principal o una constante de serie se interpreta como una lista de nombres de esquema. Si se especifican múltiples nombres de esquema, deben estar separados por comas. Cada nombre de esquema de la lista debe ajustarse a las normas para formar un identificador normal, o especificarse como identificador delimitado. El contenido de la variable del lenguaje principal o constante de serie no se convierte a mayúsculas.
- **Restricciones específicas para el SQL incorporado para aplicaciones COBOL:** Pueden aparecer un máximo de diez valores literales (no variables de lenguaje principal) en el lado derecho de una sentencia SET CURRENT PACKAGE PATH. Estos valores tienen una longitud máxima de 130 (no delimitado) o 128 (delimitado).

### Ejemplos

- *Ejemplo 1:* Establecer el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: MYPKGS, 'ABC E', SYSIBM

```
SET CURRENT PACKAGE PATH = MYPKGS, 'ABC E', SYSIBM
```

La sentencia siguiente establece una variable del lenguaje principal en el valor de la lista resultante:

```
SET :hvpklist = CURRENT PACKAGE PATH
```

El valor de la variable del lenguaje principal es: "MYPKGS", "ABC E", "SYSIBM".

- *Ejemplo 2:* Establecer el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: "SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH :hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH4","SCH5".

- *Ejemplo 3:* Establecer el registro especial CURRENT PACKAGE PATH en la lista siguiente de esquemas: "SCH1","SCH#2","SCH3","SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH = SCH1,'SCH#2',"SCH3",:hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH1","SCH#2","SCH3","SCH4","SCH5".

- *Ejemplo 4:* Borrar el registro especial CURRENT PACKAGE PATH.



## SET CURRENT PACKAGE PATH

```
SET CURRENT PACKAGE PATH = ''
```

- *Ejemplo 5:* Añadir temporalmente el esquema "SCH\_PROD" (contenido en la variable del lenguaje principal :prodschema) y el esquema "SCH\_PROD2" (contenido en la variable del lenguaje principal :prod2schema) al final del registro especial CURRENT PACKAGE PATH para la ejecución del procedimiento SUMMARIZE. Después, volver a cambiar el registro especial CURRENT PACKAGE PATH a este valor anterior.

```
SET :oldCPP = CURRENT PACKAGE PATH
```

```
SET CURRENT PACKAGE PATH = CURRENT PACKAGE PATH, :prodschema, :prod2schema
```

```
CALL SUMMARIZE(:V1, :V2)
```

```
SET CURRENT PACKAGE PATH = :oldCPP
```

- *Ejemplo 6:* Establecer el registro especial CURRENT PACKAGE PATH en una lista de nombres de esquema delimitados: "MY.SCHEMA" (punto intercalado), "OLD SCHEMA" (blanco intercalado). Utilice una sola variable del lenguaje principal que contenga ambos identificadores delimitados:

```
hv = 'MY.SCHEMA', 'OLD SCHEMA'
```

```
SET CURRENT PACKAGE PATH = :hv
```

o utilice una sola constante de serie que contenga ambos identificadores delimitados:

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

o utilice una lista de esquemas delimitados:

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

## 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 debe utilizar para las sentencias de SQL posteriores.

La sentencia no está bajo el control de la transacción.

### Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse dinámicamente. Esta sentencia no está soportada en REXX.

### Autorización

No se necesita.

### Sintaxis

►► SET CURRENT PACKAGESET [ ] *constante-serie* [ ] *variable-lenguaje-principal* ►►

### Descripción

#### *constante-serie*

Constante de serie de caracteres. Si el valor excede de 128 bytes, sólo se utilizarán los primeros 128 bytes.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. No puede establecerse en nulo. Si el valor excede de 128 bytes, sólo se utilizarán los primeros 128 bytes.

### 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.
- A diferencia de DB2 para z/OS, la sentencia SET CURRENT PACKAGESET se implementa sin soporte para un registro especial denominado CURRENT PACKAGESET.

### Ejemplos

- *Ejemplo 1:* supongamos que el ID de usuario PRODUSA ha compilado previamente una aplicación denominada TRYIT, donde 'PRODUSA' es el nombre de esquema por omisión en el archivo de vinculación. 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.

- *Ejemplo 2:* supongamos 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 dinámico.

La sentencia no está bajo el control de la transacción.

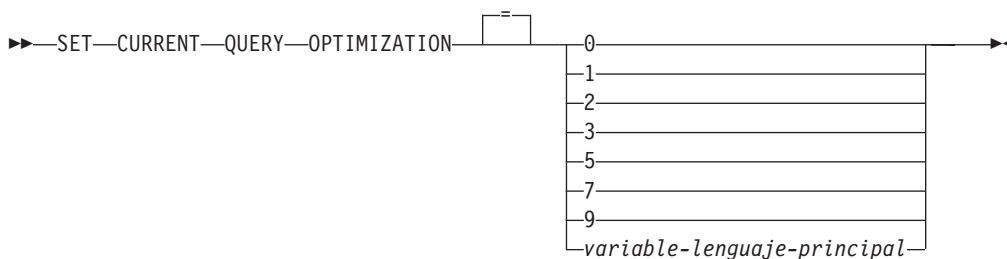
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### 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 se facilita una visión general de las clases.

- 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.
- 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 dinámico complejas, se utilizan las normas heurísticas para limitar el tiempo empleado en seleccionar un plan de acceso. Cuando sea posible, las consultas utilizarán tablas de consulta materializada en lugar de las tablas base subyacentes.

- 7 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Similar al 5 pero sin las normas 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-lenguaje-principal*

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 o 9. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador 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 normas 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 las 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 dinámico 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.
- 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 los casos en los que todavía no se ha ejecutado una sentencia SET CURRENT QUERY OPTIMIZATION, la clase de optimización de la consulta la determina 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 vinculación para especificar la clase de optimización necesaria para las sentencias vinculadas estáticamente. Si no se especifica QUERYOPT, se utilizará el valor por omisión que especifica el parámetro de configuración de la base de datos **dft\_queryopt**.
- 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.

## SET CURRENT QUERY OPTIMIZATION

### 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.

Mediante la utilización de 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.

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

## SET CURRENT REFRESH AGE

La sentencia SET CURRENT REFRESH AGE cambia el valor del registro especial CURRENT REFRESH AGE.

La sentencia no está bajo el control de la transacción.

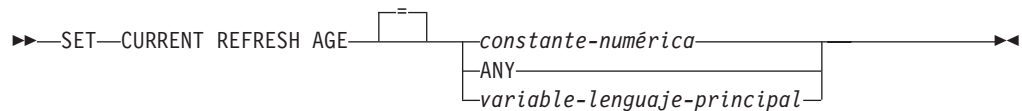
### Invocación

Esta sentencia puede incorporarse 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



### 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 consiguiente, puede ser cualquier valor).

#### ANY

Es una notación abreviada de 99.999.999.999.999.

#### *variable-lenguaje-principal*

Una variable de tipo DECIMAL(20,6) u otro tipo que se pueda asignar a DECIMAL(20,6). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815). El valor de *variable-lenguaje-principal* debe ser 0 o 99.999.999.999.999.

### Notas

- El valor inicial del registro especial CURRENT REFRESH AGE es cero.
- El valor de CURRENT REFRESH AGE se sustituye por el valor especificado. El valor debe ser 0 o 99.999.999.999.999. El valor 99.999.999.999.999 representa 9999 años, 99 meses, 99 días, 99 horas, 99 minutos y 99 segundos.

Si el valor de CURRENT REFRESH AGE es 0, las tablas de consulta materializada afectadas por este registro especial no se utilizarán para optimizar el proceso de una consulta. Si el valor de CURRENT REFRESH AGE es 99.999.999.999.999, las tablas de consulta materializada afectadas por este registro especial pueden utilizarse para optimizar el proceso de una consulta, pero sólo si el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION las incluye y el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o en un valor mayor que o igual a 5. Las

## SET CURRENT REFRESH AGE

tablas de consulta materializada afectadas por este registro especial son REFRESH DEFERRED MAINTAINED BY USER y REFRESH DEFERRED MAINTAINED BY SYSTEM.

Las tablas de consulta materializada REFRESH IMMEDIATE MAINTAINED BY SYSTEM siempre pueden utilizarse para optimizar el proceso de una consulta si el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o un valor mayor o igual a 5.

Las tablas de consulta materializada REFRESH DEFERRED MAINTAINED BY FEDERATED\_TOOL se utilizan para la optimización si el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o en un valor mayor o igual a 5 y el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se establece en ALL o incluye FEDERATED\_TOOL.

- Debe tener cuidado cuando establezca el registro especial CURRENT REFRESH AGE en un valor que no sea cero. Un tipo de tabla especificado por el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION puede no representar los valores de la tabla base subyacente. Si se utiliza un tipo de tabla así para optimizar el proceso de una consulta, el resultado de la consulta podría *no* representar de forma precisa los datos de la tabla subyacente. Puede ser razonable utilizarlo si sabe que los datos subyacentes no han cambiado, o si puede aceptar un grado de error en los resultados, basándose en su conocimiento de los datos almacenados en antememoria.
- El valor CURRENT REFRESH AGE de 99.999.999.999.999 no se puede utilizar en operaciones aritméticas de indicación de fecha y hora, porque el resultado estará fuera del rango válido para las fechas (SQLSTATE 22008).

### Ejemplos

- *Ejemplo 1:* La sentencia siguiente establece el registro especial CURRENT REFRESH AGE.

```
SET CURRENT REFRESH AGE ANY
```

- *Ejemplo 2:* El siguiente ejemplo recupera el valor del registro especial CURRENT REFRESH AGE de una variable del lenguaje principal denominada CURMAXAGE. El valor, establecido en el ejemplo anterior, es 999999999999.000000.

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO :CURMAXAGE;
```



## SET CURRENT SQL\_CCFLAGS

La sentencia SET CURRENT SQL\_CCFLAGS cambia el valor del registro especial CURRENT SQL\_CCFLAGS.

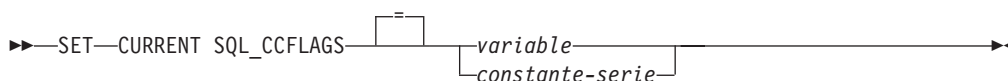
### Invocación

Esta sentencia puede incorporarse 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



### Descripción

#### *variable*

Especifica una variable que contiene uno o varios pares de nombres y valores separados mediante comas.

La variable debe tener las características siguientes (SQLSTATE 42815):

- El tipo de datos debe ser CHAR o VARCHAR. La longitud real del contenido de la variable no debe exceder la longitud máxima del registro especial.
- Debe ser una serie de caracteres en blanco, una serie vacía, o bien incluir uno o varios pares de nombres y valores, donde el nombre se separa del valor mediante un carácter de dos puntos. El nombre debe ser un identificador ordinario válido. El valor asociado con un nombre debe ser una constante de tipo BOOLEAN, una constante de tipo INTEGER o la palabra clave NULL.
- Debe rellenarse por la derecha con espacios en blanco si se utiliza una variable de caracteres de longitud fija.
- Puede incluir espacios en blanco adicionales al principio o al final de la serie, alrededor del carácter de la coma o alrededor del carácter de los dos puntos. Los espacios en blanco se pasan por alto.
- No debe ser el valor nulo.

#### *constante-serie*

Especifica una constante de serie de caracteres que contiene uno o varios pares de nombres y valores que se separan mediante comas.

La constante de la serie debe tener las características siguientes (SQLSTATE 42815):

- Debe ser una constante de serie de caracteres. La longitud de la constante no debe exceder la longitud máxima del registro especial.
- Debe ser una serie de caracteres en blanco, una serie vacía o bien incluir uno o varios pares de nombres y valores, donde el nombre se separa del valor mediante un carácter de dos puntos. El nombre debe ser un identificador

## SET CURRENT SQL\_CCFLAGS

ordinario válido. El valor asociado con un nombre debe ser una constante de tipo BOOLEAN, una constante de tipo INTEGER o la palabra clave NULL.

- Puede incluir espacios en blanco adicionales al principio o al final de la serie, alrededor del carácter de la coma o alrededor del carácter de los dos puntos. Los espacios en blanco se pasan por alto.

### Notas

- Si aparece un nombre duplicado en el contenido del registro especial CURRENT SQL\_FLAGS, sólo se utiliza el último valor (el más cercano a la parte derecha). El valor del registro especial sólo incluirá una única aparición del nombre duplicado con el valor que se utiliza. Puede utilizarse la concatenación de un nombre duplicado con un valor distinto para el valor de CURRENT SQL\_CCFLAGS con el fin de alterar temporalmente algunos valores de compilación condicional mientras se retienen otros valores.
- Cuando se recupera CURRENT SQL\_CCFLAGS, la serie que se devuelve incluye los pares de nombres y valores exclusivos en caracteres en mayúsculas, con los diversos pares separados mediante una coma y un espacio en blanco. Los pares aparecen en el orden en el que se han especificado, donde un nombre duplicado sólo se mostraría en su primera aparición, pero reflejando el valor de la posición en la que ha aparecido por última vez.
- El registro especial CURRENT SQL\_CCFLAGS puede establecerse en el valor por omisión definido para la base de datos recuperando la columna VALUE de SYSIBMADM.DBCFG donde NAME='sql\_ccflags' en una variable y, a continuación, asignando esa variable al registro especial.
- *Consideraciones sobre transacciones:* La sentencia SET SQL\_CCFLAGS no es una operación que pueda confirmarse. ROLLBACK no tiene ningún efecto en CURRENT SQL\_CCFLAGS.

### Ejemplos

- *Ejemplo 1:* definir un valor de compilación condicional para la sesión para indicar que el servidor es DB2 9.7 y que la depuración se ha establecido en false.

```
SET CURRENT SQL_CCFLAGS 'db2v97:true, debug:false'
```

- *Ejemplo 2:* ampliar la asignación CURRENT SQL\_CCFLAGS existente para establecer la depuración en true y definir el nivel de rastreo.

```
BEGIN
 DECLARE LIST VARCHAR(1024);
 SET LIST = CASE WHEN (CURRENT SQL_CCFLAGS = ' ')
 THEN 'tracelvl:3,debug:true'
 ELSE CURRENT SQL_CCFLAGS
 concat ',tracelvl:3,debug:true'
 END;
 SET CURRENT SQL_CCFLAGS = LIST;
END
```

En la asignación se utiliza una expresión CASE para gestionar la posibilidad de que el registro especial CURRENT SQL\_CCFLAGS no incluya ningún valor de compilación condicional, dando como resultado una coma inicial en el valor de la variable LIST.

Una consulta del registro especial CURRENT SQL\_CCFLAGS tras la ejecución de la sentencia en Ejemplo 1 y la sentencia compuesta de este ejemplo devolvería lo siguiente:

```
DB2V97:TRUE, DEBUG:TRUE, TRACELVL:3
```

Aunque el valor de compilación condicional para DEBUG aparecía dos veces en la variable LIST, sólo aparece una vez en el valor del registro especial, donde ha aparecido por primera vez.

## SET CURRENT TEMPORAL BUSINESS\_TIME

La sentencia SET CURRENT TEMPORAL BUSINESS\_TIME cambia el valor del registro especial CURRENT TEMPORAL BUSINESS\_TIME.

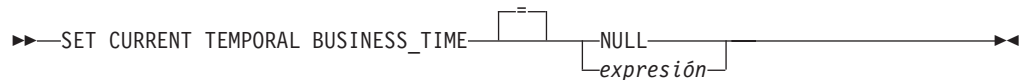
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### NULL

Especifica el valor nulo.

#### expresión

Cada expresión puede contener alguno de los operandos soportados siguientes (SQLSTATE 428HY):

- Constante
- Registro especial
- Variable (variable del lenguaje principal, variable de SQL, parámetro de SQL, variable de transición, variable global)
- Función escalar incorporada cuyos argumentos son operandos soportados. Las funciones definidas por el usuario y las funciones no deterministas no se soportan en este contexto.
- Especificación CAST donde el operando de conversión es un operando soportado
- Expresión que utiliza operadores y operandos aritméticos

### Notas

- **Consideraciones de transacción:** la sentencia SET CURRENT TEMPORAL BUSINESS\_TIME no es una operación que se pueda confirmar. ROLLBACK no tiene ningún efecto en CURRENT TEMPORAL BUSINESS\_TIME.
- **Efectos en otros registros especiales:** el valor del registro especial CURRENT TEMPORAL BUSINESS\_TIME no tiene ningún efecto en los valores de los otros registros especiales, específicamente en los registros especiales CURRENT DATE y CURRENT TIMESTAMP.

### Ejemplos

- *Ejemplo 1:* establezca el registro especial CURRENT TEMPORAL BUSINESS\_TIME en el mes anterior.  

```
SET CURRENT TEMPORAL BUSINESS_TIME = CURRENT TIMESTAMP - 1 MONTH
```

## SET CURRENT TEMPORAL BUSINESS\_TIME

- *Ejemplo 2:* establezca el registro especial CURRENT TEMPORAL BUSINESS\_TIME en el valor nulo.

```
SET CURRENT TEMPORAL BUSINESS_TIME = NULL
```

## SET CURRENT TEMPORAL SYSTEM\_TIME

La sentencia SET CURRENT TEMPORAL SYSTEM\_TIME cambia el valor del registro especial CURRENT TEMPORAL SYSTEM\_TIME.

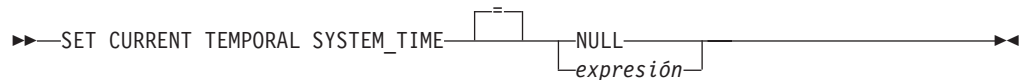
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis



### Descripción

#### NULL

Especifica el valor nulo.

#### expresión

Cada expresión puede contener alguno de los operandos soportados siguientes (SQLSTATE 428HY):

- Constante
- Registro especial
- Variable (variable del lenguaje principal, variable de SQL, parámetro de SQL, variable de transición, variable global)
- Función escalar incorporada cuyos argumentos son operandos soportados. Las funciones definidas por el usuario y las funciones no deterministas no se soportan en este contexto.
- Especificación CAST donde el operando de conversión es un operando soportado
- Expresión que utiliza operadores y operandos aritméticos

### Notas

- **Consideraciones de la transacción:** La sentencia SET CURRENT TEMPORAL SYSTEM\_TIME no es una operación que se pueda confirmar. ROLLBACK no tiene ningún efecto en CURRENT TEMPORAL SYSTEM\_TIME.
- **Efectos en otros registros especiales:** el valor del registro especial CURRENT TEMPORAL SYSTEM\_TIME no tiene ningún efecto en los valores de los otros registros especiales, específicamente en los registros especiales CURRENT DATE y CURRENT TIMESTAMP.

### Ejemplos

- *Ejemplo 1:* establezca el registro especial CURRENT TEMPORAL SYSTEM\_TIME en el mes anterior.

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT_TIMESTAMP - 1 MONTH
```

## SET CURRENT TEMPORAL SYSTEM\_TIME

- *Ejemplo 2:* establezca el registro especial CURRENT TEMPORAL SYSTEM\_TIME en el valor nulo.

```
SET CURRENT TEMPORAL SYSTEM_TIME = NULL
```

## 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 y descifrado de datos.

La sentencia no está bajo el control de la transacción.

### Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### Sintaxis

```

▶▶ SET ENCRYPTION PASSWORD [] []
 |
 |-----variable-lenguaje-principal-----▶
 |-----constante-serie-----▶

```

### Descripción

Las funciones incorporadas ENCRYPT, DECRYPT\_BIN y DECRYPT\_CHAR pueden utilizar la contraseña de cifrado para el cifrado basado en contraseña. La longitud de la contraseña debe estar comprendida entre 6 y 127 bytes y deben especificarse todos los caracteres en mayúsculas o minúsculas exactamente tal como se desea que estén, ya que no se realiza ninguna conversión a caracteres en mayúsculas. Para mantener el mejor nivel de seguridad en el sistema, se recomienda que utilice marcadores de parámetro dinámicos o una variable del lenguaje principal para especificar la contraseña, en vez de utilizar una serie literal en la sentencia SET ENCRYPTION PASSWORD.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. La longitud de *variable-lenguaje-principal* debe tener una longitud de 6 a 127 bytes (SQLSTATE 428FC). No puede establecerse 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 una serie vacía.
- La *variable-lenguaje-principal* o *constante-serie* se transmite al servidor de bases de datos siguiendo mecanismos de DB2 normales.

## SET ENCRYPTION PASSWORD

### Ejemplo

El ejemplo siguiente muestra cómo se puede establecer el registro especial ENCRYPTION PASSWORD en una aplicación de SQL incorporado usando marcadores de parámetro. Se recomienda que este registro especial se configure siempre mediante los marcadores de parámetro en las aplicaciones.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hostVarSetEncPassStmt[200];
 char hostVarPassword[128];
EXEC SQL END DECLARE SECTION;

/* preparar la sentencia con un marcador de parámetros */
strcpy(hostVarSetEncPassStmt, "SET ENCRYPTION PASSWORD = ?");
EXEC SQL PREPARE hostVarSetEncPassStmt FROM :hostVarSetEncPassStmt;

/* ejecutar la sentencia correspondiente a hostVarPassword = 'Gre89Ea' */
strcpy(hostVarPassword, "Gre89Ea");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;
```



## 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 no activo) se determina utilizando la función incorporada EVENT\_MON\_STATE.

La sentencia SET EVENT MONITOR STATE no está bajo el control de la transacción.

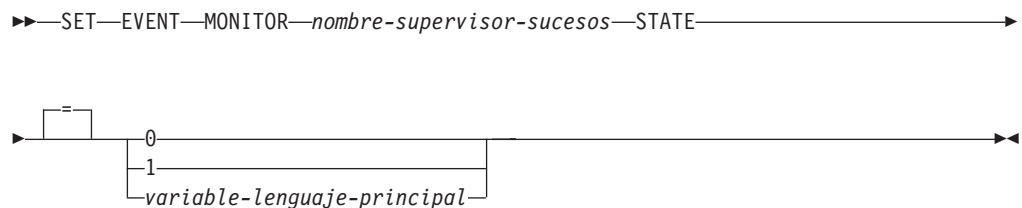
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM o SQLADM.

### Sintaxis



### Descripción

#### *nombre-supervisor-sucesos*

Identifica el supervisor de sucesos que se debe 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 pueden especificar los siguientes valores:

- 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-lenguaje-principal*

El tipo de datos es INTEGER. El valor especificado debe ser 0 ó 1 (SQLSTATE 42815). Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

### Normas

- Aunque puede definirse un número no limitado de supervisores de sucesos, puede haber un máximo de 128 supervisores de sucesos activos

## SET EVENT MONITOR STATE

simultáneamente en cada partición de base de datos. En un entorno de base de datos de varias particiones, puede haber un máximo de 32 supervisores de sucesos GLOBAL activos simultáneamente en cada base de datos.

- Para activar un supervisor de sucesos, la transacción en la que se ha creado el supervisor de sucesos debe haberse confirmado (SQLSTATE 55033). Esta norma 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 que no es WLM realiza una restauración de cualquier contador asociado al mismo. La restauración de los contadores no se realiza al activar supervisores de sucesos de unidad de trabajo, bloqueo y WLM.
- Cuando se inicia un supervisor de sucesos WRITE TO TABLE utilizando SET EVENT MONITOR STATE, éste actualiza la columna EVMON\_ACTIVATES de la vista de catálogo SYSCAT.EVENTMONITORS. Si la unidad de trabajo en la que se ha realizado la operación de conjuntos se retrotrae por cualquier motivo, la actualización del catálogo se perderá. Cuando se reinicie el supervisor de sucesos, volverá a utilizar el valor de EVMON\_ACTIVATES que se haya retrotraído.
- Si la partición de base de datos en la que se debe ejecutar el supervisor de sucesos no está activa, la activación de este se produce la próxima vez que se activa la partición de base de datos.
- Una vez activado un supervisor de sucesos, este se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva una partición de base de datos y esta se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- Si un supervisor de sucesos de actividades está activo cuando se desactiva la base de datos, se descartan los registros de actividades anotados anteriormente de la cola. Para asegurarse de obtener todos los registros del supervisor de sucesos de actividades y que no se descarte ninguno de ellos, desactive explícitamente el supervisor de sucesos de actividades antes de desactivar la base de datos. Cuando se desactiva explícitamente un supervisor de sucesos de actividades, se procesan todos los registros de actividades anotados anteriormente en la cola antes de que se desactive el supervisor de sucesos.

### Ejemplos

- *Ejemplo 1:* Active un supervisor de sucesos denominado SMITHPAY.  

```
SET EVENT MONITOR SMITHPAY STATE = 1
```
- *Ejemplo 2:* Suponga que MYSAMPLE es una base de datos de varias particiones con dos particiones de base de datos, 0 y 2. La partición 2 todavía no está activa. En la partición de base de datos 0:

```
CONNECT TO MYSAMPLE;
CREATE EVENT MONITOR MYEVMON ON DBPARTITIONNUM 2;
SET EVENT MONITOR MYEVMON STATE 1;
```

## SET EVENT MONITOR STATE

MYEVMON se activa automáticamente siempre que se activa MYSAMPLE en la partición de base de datos 2. Esto se producirá hasta que se emita **SET EVENT MONITOR MYEVMON STATE 0** o se detenga la partición 2.

---

## SET INTEGRITY

La sentencia SET INTEGRITY se utiliza para establecer el estado pendiente de integridad en las tablas, establece las tablas en un estado de acceso completo y poda el contenido de una o varias tablas de etapas.

A continuación se indican las operaciones que pueden realizarse con una sentencia SET INTEGRITY:

- Sacar una o varias tablas del estado pendiente de establecer integridad (antes denominado "estado pendiente de comprobación") realizando el proceso de integridad necesario en esas tablas.
- Sacar una o varias tablas del estado pendiente de establecer integridad sin efectuar el proceso de integridad necesario en esas tablas.
- Colocar una o varias tablas en estado pendiente de establecer integridad.
- Colocar una o varias tablas en estado de acceso completo.
- Podar el contenido de una o más tablas de etapas.

Cuando la sentencia se utiliza para efectuar el proceso de integridad de una tabla tras haberse cargado o enlazado esta, el sistema puede realizar un proceso incremental en la tabla comprobando sólo la parte añadida para verificar si existen violaciones de restricciones. Si la tabla sujeto es una tabla de consulta materializada o una tabla de etapas y se realizan operaciones de carga, enlace o desenlace en sus tablas subyacentes, el sistema puede realizar una renovación incremental en la tabla de consulta materializada o realizar una propagación incremental en la tabla de etapas utilizando sólo las partes delta de sus tablas subyacentes. Sin embargo, existen algunas situaciones en las que el sistema no podrá realizar tales optimizaciones y, en lugar de ello, realizará el proceso de integridad completo para garantizar la integridad de los datos. El proceso de integridad completo se realiza comprobando toda la tabla para verificar si existen violaciones de restricciones, volviendo a calcular la definición de una tabla de consulta materializada o marcando una tabla de etapas como incoherente. Esta última acción implica la necesidad de realizar una renovación completa de su tabla de consulta materializada asociada. Existe también una situación en que puede que desee solicitar explícitamente un proceso incremental especificando la opción INCREMENTAL.

La sentencia SET INTEGRITY está bajo control de transacciones.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios necesarios para ejecutar la sentencia SET INTEGRITY dependen de la finalidad, tal como se describe en la lista siguiente.

- Sacar tablas del estado pendiente de establecer integridad y efectuar el proceso de integridad necesario.

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para:

- Las tablas en las que se realiza el proceso de integridad y, si se proporcionan tablas de excepciones para una o más de dichas tablas, privilegio INSERT para las tablas de excepciones
- Todas las tablas de claves foráneas descendientes, tablas de consulta materializada inmediatas descendientes y tablas de etapas inmediatas descendientes que la sentencia colocará implícitamente en estado pendiente de establecer integridad
- Autorización LOAD (con condiciones). Para que la autorización LOAD pueda proporcionar privilegios válidos, deben satisfacerse todas las condiciones siguientes:
  - El proceso de integridad necesario no implica las acciones siguientes:
    - Renovar una tabla de consulta materializada
    - Propagar hacia una tabla de etapas
    - Actualizar una columna generada o de identidad
  - Si se proporcionan tablas de excepciones para una o más tablas, se otorga el acceso necesario para la duración del proceso de integridad a las tablas en las que se realiza el proceso de integridad y a las tablas de excepciones asociadas. Es decir:
    - Privilegio SELECT y DELETE para cada tabla en la que se realiza el proceso de integridad y
    - Privilegio INSERT para las tablas de excepciones
- Autorización DATAACCESS
- Sacar tablas del estado pendiente de establecer integridad sin efectuar el proceso de integridad necesario.  
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  - Privilegio CONTROL para las tablas que se procesan; privilegio CONTROL para cada tabla de claves foráneas descendiente, tabla de consulta materializada inmediata descendiente y tabla de etapas inmediata descendiente que la sentencia colocará implícitamente en estado pendiente de establecer integridad
  - Autorización LOAD
  - Autorización DATAACCESS
  - Autorización DBADM
- Colocar tablas en estado pendiente de establecer integridad.  
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  - Privilegio CONTROL para:
    - Las tablas especificadas y
    - las tablas de claves foráneas descendientes que la sentencia colocará en estado pendiente de establecer integridad, y
    - las tablas de consulta materializada inmediatas descendientes que la sentencia colocará en estado pendiente de establecer integridad, y
    - las tablas de etapas inmediatas descendientes que la sentencia colocará en estado pendiente de establecer integridad
  - Autorización LOAD
  - Autorización DATAACCESS
  - Autorización DBADM
- Colocar una tabla en estado de acceso completo

## SET INTEGRITY

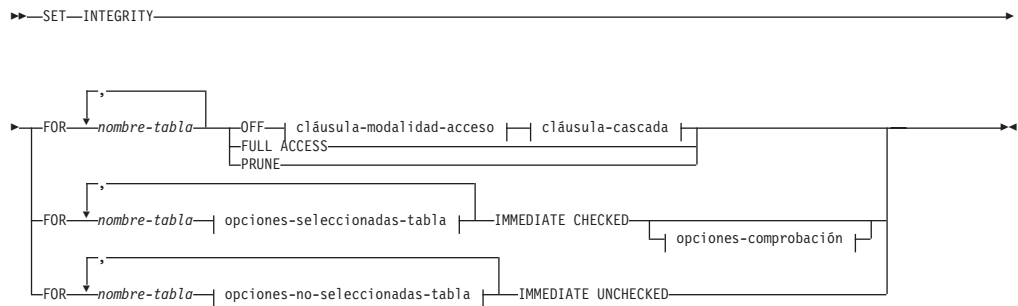
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para las tablas que se colocan en estado de acceso completo
- Autorización LOAD
- Autorización DATAACCESS
- Autorización DBADM
- Podar una tabla de etapas.

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para la tabla que está podándose
- Autorización DATAACCESS

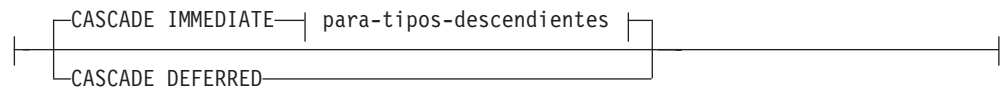
### Sintaxis



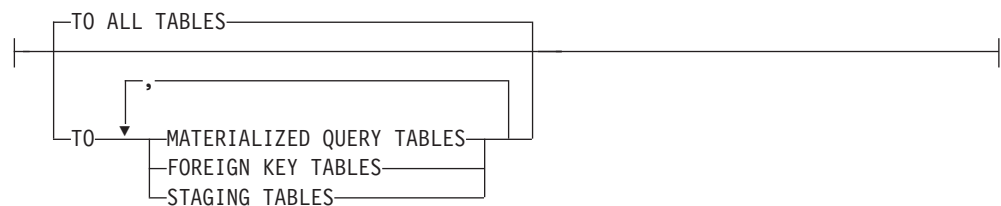
#### cláusula-modalidad-acceso:



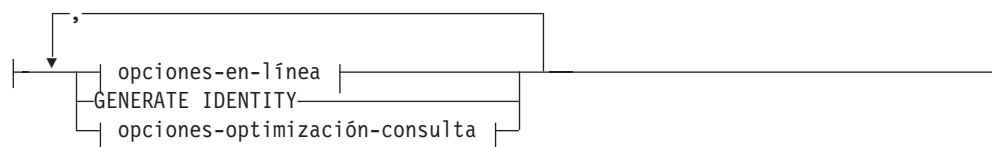
#### cláusula-cascada:



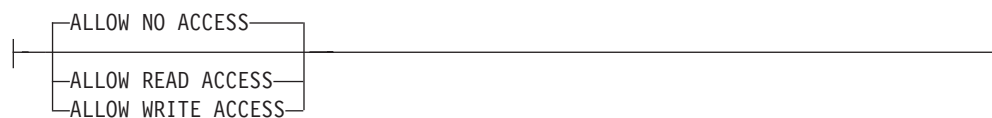
#### para-tipos-descendientes:



#### opciones-seleccionadas-tabla:



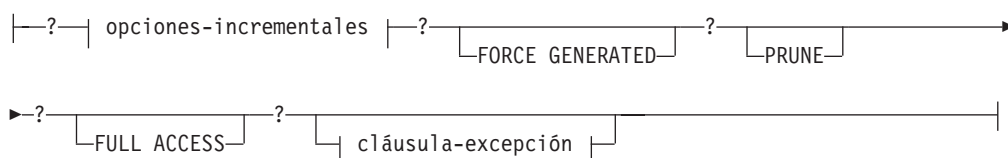
**opciones-en-línea:**



**opciones-optimización-consulta:**



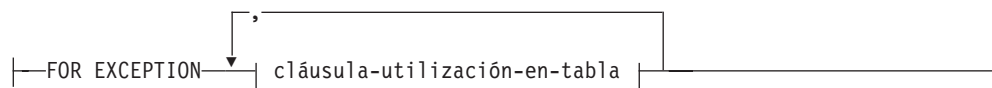
**opciones-comprobación:**



**opciones-incrementales:**



**cláusula-excepción:**



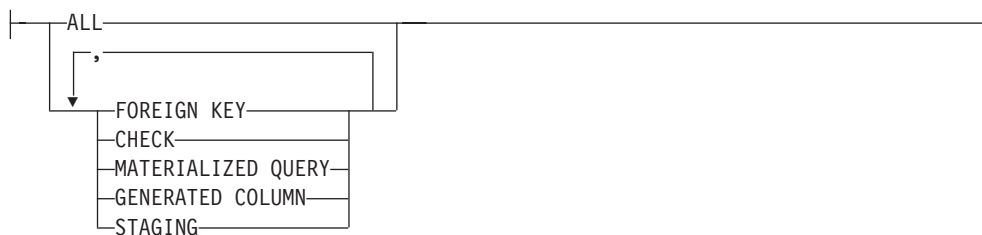
**cláusula-utilización-en-tabla:**



**opciones-no-seleccionadas-tabla:**



## opciones-integridad:



## Descripción

### **FOR** *nombre-tabla*

Identifica una o más tablas 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 se colocan en estado pendiente de establecer integridad. En una tabla que se encuentra en estado pendiente de establecer integridad sólo está permitida una actividad muy limitada.

### *cláusula-modalidad-acceso*

Especifica la posibilidad de lectura de la tabla mientras se encuentra en estado pendiente de establecer integridad.

### **NO ACCESS**

Especifica que la tabla debe colocarse en estado sin acceso pendiente de establecer integridad, en que no está permitido el acceso de lectura o grabación a la tabla.

### **READ ACCESS**

Especifica que la tabla debe colocarse en estado de acceso de lectura pendiente de establecer integridad, en que está permitido el acceso de lectura a la parte no añadida de la tabla. Esta opción no está permitida en una tabla en estado sin acceso pendiente de establecer integridad (SQLSTATE 428FH).

### *cláusula-cascada*

Especifica si el estado pendiente de establecer integridad de la tabla a la que se hace referencia en la sentencia SET INTEGRITY debe aplicarse en cascada inmediatamente a las tablas descendientes.

### **CASCADE IMMEDIATE**

Especifica que el estado pendiente de establecer integridad debe extenderse inmediatamente a las tablas descendientes.

### *para-tipos-descendientes*

Especifica el tipo de tablas descendientes a las que se aplica en cascada de inmediato el estado pendiente de establecer integridad.

### **TO ALL TABLES**

Especifica que el estado pendiente de establecer integridad debe aplicarse inmediatamente en cascada a todas las tablas descendientes de las tablas de la lista de invocación. Las tablas descendientes incluyen todas las tablas de claves foráneas descendientes, las tablas de etapas inmediatas y las tablas de consulta materializada inmediatas que son descendientes de las tablas de la lista de invocación o que son descendientes de las tablas de claves foráneas descendientes.



Especificar TO ALL TABLES equivale a especificar TO FOREIGN KEY TABLES, TO MATERIALIZED QUERY TABLES y TO STAGING TABLES, todas ellas en la misma sentencia.

#### **TO MATERIALIZED QUERY TABLES**

Si sólo se especifica TO MATERIALIZED QUERY TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente sólo a las tablas de consulta materializada inmediatas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad. Si se especifica TO FOREIGN KEY TABLES y también TO MATERIALIZED QUERY TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendientes, a todas las tablas de consulta materializada inmediatas descendientes de las tablas de la lista de invocación y a todas las tablas de consulta materializada inmediatas que son descendientes de las tablas de claves foráneas descendientes.

#### **TO FOREIGN KEY TABLES**

Especifica que el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a las tablas de claves foráneas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad.

#### **TO STAGING TABLES**

Especifica que el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a las tablas de etapas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad. Si se especifica TO FOREIGN KEY TABLES y también TO STAGING TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendientes, a todas las tablas de etapas inmediatas descendientes de las tablas de la lista de invocación y a todas las tablas de etapas inmediatas que son descendientes de las tablas de claves foráneas descendientes.

#### **CASCADE DEFERRED**

Especifica que sólo las tablas de la lista de invocación deben colocarse en estado pendiente de establecer integridad. Los estados de las tablas descendientes no se cambiarán. Posteriormente, las tablas de claves foráneas descendientes pueden colocarse implícitamente en estado pendiente de establecer integridad cuando se comprueben sus tablas padre para verificar si existen violaciones de las restricciones. Las tablas de consulta materializada inmediatas descendientes y las tablas de etapas inmediatas descendientes pueden colocarse implícitamente en estado pendiente de establecimiento de integridad al comprobarse una de sus tablas subyacentes para verificar si existen violaciones de integridad. Una consulta de una tabla en estado pendiente de establecimiento de integridad podría realizarse correctamente si la consulta, en lugar de la tabla especificada, accede a una tabla de consulta materializada admisible que no se encuentra en estado pendiente de establecimiento de integridad.

Si no se especifica la *cláusula-cascada*, el estado pendiente de establecer integridad se aplica en cascada inmediatamente a todas las tablas descendientes.

## SET INTEGRITY

### IMMEDIATE CHECKED

Especifica que la tabla debe sacarse del estado pendiente de establecer integridad realizando el proceso de integridad necesario en la tabla. Esto se realiza de acuerdo con la información que se ha establecido en las columnas STATUS y CONST\_CHECKED de la vista de catálogo SYSCAT.TABLES. Es decir:

- El valor de la columna STATUS debe ser 'C' (la tabla se encuentra en estado pendiente de establecer integridad) o se devuelve un error (SQLSTATE 51027), a menos que la tabla sea una tabla de claves foráneas descendiente, una tabla de consulta materializada descendiente o una tabla de etapas descendiente de una tabla especificada en la lista, que se encuentre en estado pendiente de establecer integridad y cuyos ancestros inmediatos también estén en la lista.
- Si la tabla que está comprobándose está en estado pendiente de establecer integridad, el valor de CONST\_CHECKED indica qué opciones de integridad deben comprobarse.

Cuando la tabla se saca del estado pendiente de establecer integridad, sus tablas descendientes se colocan, si es necesario, en estado pendiente de establecer integridad. Se devuelve un aviso para indicar que las tablas descendientes se han colocado en estado pendiente de establecer integridad (SQLSTATE 01586).

Si la tabla es una tabla de consulta materializada mantenida por el sistema, los datos se comprueban con la consulta y se renuevan en función de las necesidades. (No se puede utilizar IMMEDIATE CHECKED para las tablas de consulta materializada mantenidas por el usuario.) Si la tabla es una tabla de etapas, los datos se comprueban con su definición de consulta y se propagan en función de las necesidades.

Cuando se comprueba la integridad de una tabla hija:

- Ninguno de sus padres puede estar en estado pendiente de establecer integridad, o
- cada uno de sus padres debe comprobarse para verificar si existen violaciones de restricciones en la misma sentencia SET INTEGRITY.

Cuando se renueva una tabla de consulta materializada inmediata o cuando se propagan deltas a una tabla de etapas:

- Ninguna de sus tablas subyacentes puede estar en estado pendiente de establecer integridad, o
- cada una de sus tablas subyacentes debe comprobarse en la misma sentencia SET INTEGRITY.

De lo contrario, se devolverá un error (SQLSTATE 428A8).

#### *opciones-seleccionadas-tabla*

##### *opciones-en-línea*

Especifica la accesibilidad de la tabla mientras se está procesando.

##### **ALLOW NO ACCESS**

Especifica que ningún otro usuario puede acceder a la tabla mientras se está procesando, excepto si utiliza el nivel de aislamiento de lectura no confirmada.

##### **ALLOW READ ACCESS**

Especifica que los demás usuarios tienen acceso de sólo lectura a la tabla mientras se procesa.

**ALLOW WRITE ACCESS**

Especifica que los demás usuarios tienen acceso de lectura y grabación a la tabla mientras se procesa.

**GENERATE IDENTITY**

Especifica que, si la tabla contiene una columna de identidad, la sentencia SET INTEGRITY genera los valores. Por omisión, cuando se especifica la opción GENERATE IDENTITY, la sentencia SET INTEGRITY únicamente genera los valores de la columna de identidad de las filas enlazadas. Debe especificarse la opción NOT INCREMENTAL con la opción GENERATE IDENTITY para que la sentencia SET INTEGRITY genere los valores de la columna de identidad de todas las filas de la tabla, entre ellas las filas enlazadas, las filas cargadas y las filas existentes. Si no se especifica la opción GENERATE IDENTITY, los actuales valores de la columna de identidad de todas las filas de la tabla permanecen invariables. Cuando la tabla es una tabla temporal de período de sistema, sólo está permitido utilizar GENERATE IDENTITY con la opción NOT INCREMENTAL si primero emite una sentencia ALTER TABLE con la cláusula DROP VERSIONING (SQLSTATE 428FH).

*opciones-optimización-consulta*

Especifica las opciones de optimización de consulta para el mantenimiento de las tablas de consulta materializada REFRESH DEFERRED.

**ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY**

Especifica que, si el registro especial CURRENT REFRESH AGE se establece en 'ANY', el mantenimiento del *nombre-tabla* permitirá el uso de tablas de consulta materializada REFRESH DEFERRED para optimizar la consulta que mantiene el *nombre-tabla*. Si el *nombre-tabla* no es una tabla de consulta materializada REFRESH DEFERRED, se devuelve un error (SQLSTATE 428FH). Las tablas de consulta materializada REFRESH IMMEDIATE siempre se tienen en cuenta en la optimización de consulta.

*opciones-comprobación**opciones-incrementales***INCREMENTAL**

Especifica la aplicación del proceso de integridad a la parte añadida (si existe) de la tabla. Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que debe comprobarse toda la tabla para verificar la integridad de los datos), se devuelve un error (SQLSTATE 55019).

**NOT INCREMENTAL**

Especifica la aplicación del proceso de integridad a toda la tabla. Si la tabla es una tabla de consulta materializada, se vuelve a calcular la definición de la tabla de consulta materializada. Si la tabla incluye la definición de, como mínimo, una restricción, esta opción causa el proceso completo de las tablas de claves foráneas descendientes y las tablas de consulta materializada inmediatas descendientes. Si la tabla es una tabla de etapas, se establece en un estado incoherente.

## SET INTEGRITY

Si no se especifica la cláusula *opciones-incrementales*, el sistema determina si es posible el proceso incremental; si no es posible, se comprueba toda la tabla.

### FORCE GENERATED

Si la tabla incluye columnas generadas por expresión, los valores se calculan basándose en la expresión y se almacenan en la columna. Si no se especifica esta opción, los valores actuales se comparan con el valor calculado de la expresión, como si hubiera en vigor una restricción de comprobación de igualdad. Si se realiza el proceso de integridad de la tabla de manera incremental, las columnas generadas sólo se calculan para la parte añadida. Cuando la tabla es una tabla temporal de período de sistema, la opción FORCE GENERATED sólo está permitida si se emite primero una sentencia ALTER TABLE con la cláusula DROP VERSIONING (SQLSTATE 428FH).

### PRUNE

Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH). Si también se especifica la opción de comprobación INCREMENTAL, se devuelve un error (SQLSTATE 428FH).

### FULL ACCESS

Especifica que la tabla debe pasar a ser una tabla de acceso completo tras ejecutarse la sentencia SET INTEGRITY.

Cuando se realiza el proceso incremental de una tabla subyacente (que tiene tablas de consulta materializada inmediatas dependientes o tablas de etapas inmediatas dependientes) de la lista de invocación, la tabla subyacente se establece, en función de las necesidades, en la modalidad sin movimiento de datos tras ejecutarse la sentencia SET INTEGRITY. Cuando se sacan del estado pendiente de establecer integridad todas las tablas de etapas y tablas de consulta materializada inmediatas dependientes que pueden renovarse de manera incremental, la tabla subyacente pasa automáticamente del estado sin movimiento de datos al estado de acceso completo. Si se especifica la opción FULL ACCESS con la opción IMMEDIATE CHECKED, la tabla subyacente se establece directamente en estado de acceso completo (y elude el estado sin movimiento de datos). En DB2 Versión 9.7 Fixpack 1 y releases posteriores, la especificación de la opción FULL ACCESS sólo elimina la dependencia entre las tablas de dependencia y la tabla subyacente. La tabla subyacente sigue sin estar disponible hasta que la tarea de desenlace de partición asíncrona completa el proceso de desenlace de partición de datos.

Las tablas de consulta materializada inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH TABLE posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden marcarse como incoherentes.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consulta materializada inmediatas dependientes ni tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en estado de acceso

completo tras ejecutarse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

*cláusula-excepción*

**FOR EXCEPTION**

Especifica que cualquier fila en que se produzca la violación de una restricción que se comprueba debe moverse a una tabla de excepciones. Aunque se detecten errores, la tabla se saca del estado pendiente de establecer integridad. Se devuelve un aviso para indicar que se ha movido una o varias filas a las tablas de excepciones (SQLSTATE 01603).

Si no se especifica la opción FOR EXCEPTION y se produce una violación de alguna restricción, sólo se devuelve la primera violación que se ha detectado (SQLSTATE 23514). Si se produce una violación en alguna de las tablas, todas las tablas quedan en estado pendiente de establecer integridad.

Se recomienda utilizar siempre la opción FOR EXCEPTION al comprobar la existencia de violaciones de restricciones para evitar la retrotracción de la sentencia SET INTEGRITY en caso de hallarse una violación.

Cuando la tabla especificada después de la palabra clave IN es una tabla temporal de período de sistema, la opción FOR EXCEPTION sólo está permitida si se emite primero una sentencia ALTER TABLE con la cláusula DROP VERSIONING (SQLSTATE 428FH).

**IN** *nombre-tabla*

Especifica la tabla de la que se deben mover las filas que violan restricciones. Debe haber una tabla de excepciones especificada para cada tabla que se comprueba. Esta cláusula no puede especificarse para una tabla de consulta materializada o una tabla de etapas (SQLSTATE 428A7).

**USE** *nombre-tabla*

Especifica la tabla de excepciones a la que se deben mover las filas de errores.

**FULL ACCESS**

Si se especifica la opción FULL ACCESS como única operación de la sentencia, la tabla se coloca en estado de acceso completo sin volver a comprobar las violaciones de integridad. Sin embargo, las tablas de consulta materializada inmediatas dependientes que no se han renovado pueden necesitar un nuevo cálculo completo en las sentencias REFRESH TABLE posteriores, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes delta de la tabla pueden establecerse en estado incompleto. Esta opción sólo puede especificarse para una tabla que se encuentra en estado sin movimiento de datos o sin acceso, pero no en estado pendiente de establecer integridad (SQLSTATE 428FH).

**PRUNE**

Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH).

*opciones-no-seleccionadas-tabla*

## SET INTEGRITY

### *opciones-integridad*

Permite definir los tipos de proceso de integridad necesarios que deben eludirse cuando se saca la tabla del estado pendiente de establecer integridad.

#### **ALL**

La tabla se sacará inmediatamente del estado pendiente de establecer integridad sin efectuar ningún proceso de integridad necesario.

#### **FOREIGN KEY**

No se realizará la comprobación de las restricciones de clave foránea necesaria al sacar la tabla del estado pendiente de establecer integridad.

#### **CHECK**

No se realizará la comprobación de las restricciones de comprobación necesaria al sacar la tabla del estado pendiente de establecer integridad.

#### **MATERIALIZED QUERY**

No se realizará la renovación necesaria de una tabla de consulta materializada al sacar la tabla del estado pendiente de establecer integridad.

#### **GENERATED COLUMN**

No se realizará la comprobación de las restricciones de columna generada necesaria al sacar la tabla del estado pendiente de establecer integridad.

#### **STAGING**

No se realizará la propagación de datos necesaria a una tabla de etapas al sacar la tabla del estado pendiente de establecer integridad.

Si no es necesario ningún otro tipo de proceso de integridad en la tabla una vez marcado como eludido un tipo de proceso de integridad concreto, la tabla se saca de inmediato del estado pendiente de establecer integridad.

#### **FULL ACCESS**

Especifica que las tablas van a ser tablas de acceso completo tras ejecutarse la sentencia SET INTEGRITY.

Cuando se realiza el proceso incremental de una tabla subyacente de la lista de invocación y esta tiene tablas de consulta materializada inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establece, en función de las necesidades, en estado sin movimiento de datos tras ejecutarse la sentencia SET INTEGRITY. Una vez sacadas del estado pendiente de establecer integridad todas las tablas de etapas y tablas de consulta materializada inmediatas dependientes que pueden renovarse de manera incremental, la tabla subyacente pasa automáticamente del estado sin movimiento de datos al estado de acceso completo. Si se especifica la opción FULL ACCESS con la opción IMMEDIATE UNCHECKED, la tabla subyacente se establece directamente en estado de acceso completo (elude el estado sin movimiento de datos). Las tablas de consulta materializada inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH TABLE posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden marcarse como incoherentes.

En DB2 V9.7 Fixpack 1 y releases posteriores, la especificación de la opción FULL ACCESS sólo elimina la dependencia entre las tablas de dependencia y la tabla subyacente. La tabla subyacente sigue sin estar disponible hasta que la tarea de desenlace de partición asíncrona completa el proceso de desenlace de partición de datos.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consulta materializada inmediatas dependientes ni tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en estado de acceso completo tras ejecutarse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

Si se ha especificado la opción FULL ACCESS con la opción IMMEDIATE UNCHECKED y la sentencia no saca la tabla del estado pendiente de establecer integridad, se devuelve un error (SQLSTATE 428FH).

#### IMMEDIATE UNCHECKED

Especifica uno de los siguientes:

- La tabla debe sacarse del estado pendiente de establecer integridad inmediatamente sin ningún proceso de integridad necesario.
- La tabla debe eludir uno o varios tipos de proceso de integridad necesarios al sacarse la tabla del estado pendiente de establecer integridad mediante una sentencia SET INTEGRITY posterior con la opción IMMEDIATE CHECKED.

Antes de utilizar esta opción, examine cómo afecta a la integridad de los datos. Consulte la sección "Notas".

#### Notas

- Efectos en las tablas que están en uno de los estados restringidos relacionados con establecer integridad:
  - La utilización de INSERT, UPDATE o DELETE no está permitida en una tabla que está en estado de acceso de lectura o sin acceso. Además, se rechazará cualquier sentencia que requiera una modificación de este tipo para una tabla que se encuentra en un estado de este tipo. Por ejemplo, no está permitido suprimir una fila de una tabla padre que se aplica en cascada a una tabla dependiente que está en estado sin acceso.
  - La utilización de SELECT no está permitida en una tabla que está en estado sin acceso. Además, se rechazará cualquier sentencia que requiera acceso de lectura a una tabla que está en estado sin acceso.
  - Normalmente las nuevas restricciones añadidas a una tabla se imponen inmediatamente. Sin embargo, si la tabla está en estado pendiente de establecer integridad, la comprobación de cualquier restricción nueva se diferirá hasta que se saque la tabla del estado pendiente de establecer integridad. Si la tabla está en estado pendiente de establecer integridad, la adición de una nueva restricción causará que la tabla se coloque en estado sin acceso pendiente de establecer integridad, pues la validez de los datos está en peligro.
  - La sentencia CREATE INDEX no puede hacer referencia a ninguna tabla que esté en estado de acceso de lectura o sin acceso. De forma similar, una sentencia ALTER TABLE para añadir una clave primaria o una restricción de unicidad no puede hacer referencia a ninguna tabla que esté en estado de acceso de lectura o sin acceso.
  - El programa de utilidad de importación no puede ejecutarse en una tabla que está en estado de acceso de lectura o sin acceso.

## SET INTEGRITY

- El programa de utilidad de exportación no puede ejecutarse en una tabla que está en estado sin acceso, pero sí en una tabla que está en estado de acceso de lectura. Si una tabla está en estado de acceso de lectura, el programa de utilidad de exportación sólo exportará los datos de la parte no añadida.
- Las operaciones (como REORG, REDISTRIBUTE, la actualización de la clave de distribución, la actualización de la clave de clúster multidimensional, la actualización de la clave de agrupación en clúster de rangos, la actualización de la clave de particionamiento de tabla, etc.) que pueden implicar un movimiento de datos dentro de una tabla no están permitidas en una tabla que está en alguno de los estados siguientes: acceso de lectura, sin acceso o sin movimiento de datos.
- Los programas de utilidad LOAD, BACKUP, RESTORE, UPDATE STATISTICS, RUNSTATS, REORGCHK, LIST HISTORY y ROLLFORWARD pueden utilizarse en una tabla que está en cualquiera de los estados siguientes: acceso completo, acceso de lectura, sin acceso o sin movimiento de datos.
- Las sentencias ALTER TABLE, COMMENT, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE y SET INTEGRITY pueden hacer referencia a una tabla que está en cualquiera de los estados siguientes: acceso completo, acceso de lectura, sin acceso o sin movimiento de datos. Sin embargo, pueden hacer que la tabla se establezca en el estado sin acceso.
- Los paquetes, las vistas y cualquier otro objeto que dependa de una tabla que está en estado sin acceso devolverán un error al accederse a la tabla en tiempo de ejecución. Los paquetes que dependan de una tabla que está en estado de acceso de lectura devolverán un error cuando se intente realizar una operación de inserción, actualización o supresión en 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 activadores. Del mismo modo, la actualización de columnas generadas mediante la opción FORCE GENERATED no produce la activación de activadores.

- Aviso acerca de la utilización de la cláusula IMMEDIATE UNCHECKED:
  - Esta cláusula se ha diseñado para que la utilicen los programas de utilidad y no se recomienda que la utilicen los programas de aplicación. Si en la tabla existen datos que no cumplen las especificaciones de integridad que se han definido para la tabla y se utiliza la opción IMMEDIATE UNCHECKED, puede que se devuelvan resultados de consulta incorrectos.

El hecho de haber sacado la tabla del estado pendiente de establecer integridad sin realizar el proceso de integridad necesario quedará registrado en el catálogo (el byte respectivo de la columna CONST\_CHECKED de la vista SYSCAT.TABLES se establecerá 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 no cambia hasta que:

    - La tabla vuelve a colocarse en estado pendiente de establecer integridad (haciéndose referencia a la tabla en una sentencia SET INTEGRITY con la opción OFF), momento en que los valores 'U' de la columna CONST\_CHECKED se cambian por valores 'W', para indicar que el usuario había asumido anteriormente la responsabilidad de comprobar la integridad de los datos y el sistema debe verificar los datos.
    - Se descartan todas las restricciones no comprobadas de la tabla.



El estado 'W' se diferencia del estado 'N' en que registra el hecho de que el usuario comprobaba anteriormente la integridad, pero que el sistema todavía no lo hace. Si el usuario emite la sentencia SET INTEGRITY ... IMMEDIATE CHECKED con la opción NOT INCREMENTAL, el sistema vuelve a comprobar toda la tabla para verificar la integridad de los datos (o realiza una renovación completa en una tabla de consulta materializada) y, a continuación, cambia el estado 'W' por el estado 'Y'. Si se especifica IMMEDIATE UNCHECKED, o si no se especifica NOT INCREMENTAL, el estado 'W' vuelve a cambiar por el estado 'U' para registrar el hecho de que el sistema todavía no ha verificado algunos de los datos. En este último caso (cuando no se especifica NOT INCREMENTAL), se devuelve un aviso (SQLSTATE 01636).

Si se ha comprobado la integridad de una tabla subyacente utilizando la cláusula IMMEDIATE UNCHECKED, los valores 'U' de la columna CONST\_CHECKED de la tabla subyacente se propagarán a la columna CONST\_CHECKED correspondiente de:

- Las tablas de consulta materializada inmediatas dependientes
- Las tablas de consulta materializada diferidas dependientes
- Las tablas de etapas dependientes

Para una tabla de consulta materializada inmediata dependiente, esta propagación tiene lugar cada vez que la tabla subyacente se saca del estado pendiente de establecer integridad y cada vez que se renueva la tabla de consulta materializada. Para una tabla de consulta materializada diferida dependiente, esta propagación tiene lugar siempre que se renueva la tabla de consulta materializada. Para las tablas de etapas dependientes, esta propagación tiene lugar cada vez que se saca la tabla subyacente del estado pendiente de establecer integridad. Estos valores 'U' propagados de las columnas CONST\_CHECKED de las tablas de consulta materializada dependientes y tablas de etapas registran el hecho de que estas tablas de consulta materializada y tablas de etapas dependen de alguna tabla subyacente cuyo proceso de integridad necesario se ha eludido con la opción IMMEDIATE UNCHECKED.

Para una tabla de consulta materializada, el valor 'U' de la columna CONST\_CHECKED que la tabla subyacente ha propagado se conservará hasta que la tabla de consulta materializada se haya renovado por completo y ninguna de sus tablas subyacentes tenga un valor 'U' en su correspondiente columna CONST\_CHECKED. Tras realizarse esta renovación, el valor 'U' de la columna CONST\_CHECKED de la tabla de consulta materializada cambiará por 'Y'.

Para una tabla de etapas, el valor 'U' de la columna CONST\_CHECKED que la tabla subyacente ha propagado se conservará hasta que se haya renovado la correspondiente tabla de consulta materializada diferida de la tabla de etapas. Tras realizarse esta renovación, el valor 'U' de la columna CONST\_CHECKED de la tabla de etapas cambiará por 'Y'.

- Si una tabla hija y su tabla padre se comprueban en la misma sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED y la tabla padre necesita que se realice una comprobación completa de sus restricciones, se comprobarán las restricciones de clave foránea de la tabla hija, con independencia de si la tabla hija tiene o no un valor 'U' en la columna CONST\_CHECKED para las restricciones de clave foránea.
- Si la tabla es una tabla particionada de datos y existen índices no particionados (a excepción del índice de vía de acceso de columna XML) que han de mantenerse, el comportamiento de IMMEDIATE UNCHECKED cuando se especifica una única tabla de destino es el mismo que el comportamiento de

## SET INTEGRITY

IMMEDIATE CHECKED con la opción ALLOW WRITE ACCESS: se realiza todo el proceso de integridad y se devuelven los errores resultantes. Si la sentencia hace referencia a más de una tabla de destino, se devuelve un error (SQLSTATE 428FH).

- Después de haber añadido datos utilizando LOAD INSERT o ALTER TABLE ATTACH, la sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED comprueba la tabla para verificar si existen violaciones de restricciones. El sistema determina si es posible o no el proceso incremental en la tabla. Si es posible, sólo se comprueba la parte añadida para las violaciones de integridad. Si no es posible, el sistema comprueba toda la tabla para verificar si existen violaciones de integridad.

- Veamos la sentencia siguiente:

```
SET INTEGRITY FOR T IMMEDIATE CHECKED
```

En los escenarios siguientes, no reciben soporte ni la opción de comprobación INCREMENTAL para T ni una renovación incremental de T (si T es una tabla de consulta materializada (MQT) o una tabla de etapas):

- Se han añadido nuevas restricciones para T mientras se encuentra en estado pendiente de establecimiento de integridad.
  - Cuando ha tenido lugar una operación LOAD REPLACE para T, su tabla padre o sus tablas subyacentes.
  - Cuando se ha activado la opción NOT LOGGED INITIALLY WITH EMPTY TABLE tras la última comprobación de integridad realizada en T, en sus tablas padre o en sus tablas subyacentes.
  - El efecto de aplicación en cascada del proceso completo, cuando se ha comprobado la integridad de T (o una tabla subyacente, si T es una tabla de consulta materializada o una tabla de etapas) de forma no incremental.
  - Si el espacio de tablas que contiene la tabla o su tabla padre (o tabla subyacente de una tabla de consulta materializada o una tabla de etapas) se ha retrotraído hasta un momento dado y la tabla y su tabla padre (o tabla subyacente si la tabla es una tabla de consulta materializada o una tabla de etapas) residen en distintos espacios de tablas.
  - T es una MQT, y ha tenido lugar una operación LOAD REPLACE o LOAD INSERT directamente en T tras la última renovación.
- El proceso incremental se utilizará siempre que la situación lo permita, pues es más eficaz. En la mayoría de los casos, no se necesita la opción INCREMENTAL. Sin embargo, sí se necesita para garantizar que las comprobaciones de integridad realmente se procesan de forma incremental. Si el sistema detecta que es necesario realizar un proceso completo para garantizar la integridad de los datos, se devuelve un error (SQLSTATE 55019).
  - Si las condiciones del proceso completo que se describen en el punto anterior no se satisfacen, el sistema intentará comprobar sólo la integridad de la parte añadida o realizar una renovación incremental (si es una tabla de consulta materializada) cuando el usuario no especifique la opción NOT INCREMENTAL para la sentencia SET INTEGRITY FOR T IMMEDIATE CHECKED.
  - Si se produce un error durante el proceso de integridad, se retrotraerán todos los efectos del proceso (incluidas la supresión realizada en la tabla original y la inserción en las tablas de excepciones).
  - Si la emisión de una sentencia SET INTEGRITY con la opción FORCE GENERATED no se ejecuta satisfactoriamente porque el espacio de anotaciones cronológicas es insuficiente, incrementa el espacio de anotaciones cronológicas activo disponible y vuelva a emitir la sentencia SET INTEGRITY. También puede utilizar la sentencia SET INTEGRITY con las opciones GENERATED COLUMN e

IMMEDIATE UNCHECKED para eludir la comprobación de las columnas generadas para la tabla. A continuación, emita una sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED y sin la opción FORCE GENERATED para comprobar si en la tabla existen otras violaciones de la integridad (si corresponde) y sacar la tabla del estado pendiente de establecer integridad. Cuando se haya sacado la tabla del estado pendiente de establecer integridad, las columnas generadas podrán actualizarse para que tomen sus valores por omisión (generados) asignándolos a la palabra clave DEFAULT en una sentencia UPDATE. Esta acción se realiza utilizando varias sentencias de actualización buscada basadas en rangos (cada una de ellas seguida de una confirmación) o bien haciendo uso de un enfoque basado en cursores utilizando confirmaciones intermitentes. Deberá utilizarse un cursor “con retención” si van a retenerse bloqueos tras las confirmaciones intermitentes del enfoque basado en cursores.

- Cuando una tabla se ha colocado en estado pendiente de establecer integridad mediante la opción CASCADE DEFERRED de la sentencia SET INTEGRITY o el mandato LOAD, o mediante la sentencia ALTER TABLE con la cláusula ATTACH, y se ha comprobado para verificar si existen violaciones de integridad mediante la opción IMMEDIATE CHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendientes, tablas de consulta materializada inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad, según proceda:
  - Si se comprueba toda la tabla para verificar si existen violaciones de integridad, sus tablas de claves foráneas descendientes, tablas de consulta materializada inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad.
  - Si la tabla se comprueba para verificar si existen violaciones de integridad de manera incremental, sus tablas de consulta materializada inmediatas descendientes y tablas de etapas se colocarán en estado pendiente de establecer integridad y sus tablas de claves foráneas descendientes seguirán en sus estados originales.
  - Si la tabla no necesita ninguna comprobación, sus tablas de consulta materializada inmediatas descendientes, tablas de etapas descendientes y tablas de claves foráneas descendientes seguirán en sus estados originales.
- Cuando una tabla se ha colocado en estado pendiente de establecer integridad mediante la opción CASCADE DEFERRED (de la sentencia SET INTEGRITY o del mandato LOAD) y se ha sacado del estado pendiente de establecer integridad mediante la opción IMMEDIATE UNCHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendientes, tablas de consulta materializada inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad, según proceda:
  - Si la tabla se ha cargado utilizando la modalidad REPLACE, sus tablas de claves foráneas descendientes, tablas de consulta materializada inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad.
  - Si la tabla se ha cargado utilizando la modalidad INSERT, sus tablas de consulta materializada inmediatas descendientes y tablas de etapas se colocarán en estado pendiente de establecer integridad y sus tablas de claves foráneas descendientes seguirán en sus estados originales.
  - Si la tabla no se ha cargado, sus tablas de consulta materializada inmediatas descendientes, tablas de etapas descendientes y tablas de claves foráneas descendientes seguirán en sus estados originales.
- SET INTEGRITY normalmente es una sentencia de larga ejecución. Por ello, para reducir el riesgo de retrotracción de toda la sentencia debido a un tiempo de

espera excedido de bloqueo, puede emitir la sentencia SET CURRENT LOCK TIMEOUT con la opción WAIT antes de ejecutar la sentencia SET INTEGRITY y, a continuación, restaurar el valor anterior del registro especial una vez confirmada la transacción. No obstante, tenga presente que el registro especial CURRENT LOCK TIMEOUT sólo afecta a un conjunto de tipos de bloqueo concreto.

- Si utiliza la opción ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY, compruebe que el orden de mantenimiento es correcto para las tablas de consulta materializada REFRESH DEFERRED. Por ejemplo, imagine dos tablas de consulta materializada, MQT1 y MQT2, cuyas consultas materializadas comparten las mismas tablas subyacentes. La consulta materializada de MQT2 se puede calcular utilizando MQT1, en lugar de las tablas subyacentes. Si se emplean distintas sentencias para mantener estas dos tablas de consulta materializada, y se mantiene MQT2 en primer lugar, el sistema puede elegir utilizar el contenido de MQT1, que todavía no se ha mantenido, para mantener MQT2. En ese caso, MQT1 contendría los datos actuales pero MQT2 podría contener todavía datos obsoletos, aunque el mantenimiento de ambas se realizara casi al mismo tiempo. El orden de mantenimiento correcto, si se utilizan dos sentencias SET INTEGRITY en lugar de una, consiste en mantener MQT1 en primer lugar.
- Al utilizar la sentencia SET INTEGRITY para realizar el proceso de integridad en una tabla base cargada o enlazada, se recomienda procesar sus tablas de consulta materializada REFRESH IMMEDIATE dependientes y sus tablas de etapas PROPAGATE IMMEDIATE en la misma sentencia SET INTEGRITY para evitar colocar estas tablas dependientes en el estado sin acceso pendiente de establecer integridad al final del proceso de SET INTEGRITY. Tenga en cuenta que, en el caso de las tablas base con un gran número de tablas de consulta materializada REFRESH IMMEDIATE dependientes y tablas de etapas PROPAGATE IMMEDIATE, puede ser imposible procesar todos los dependientes en la misma sentencia que la tabla base debido a restricciones de memoria.
- Si se especifica la opción FORCE GENERATED o GENERATE IDENTITY y la columna generada forma parte de un índice exclusivo, la sentencia SET INTEGRITY devuelve un error (SQLSTATE 23505) y se retrotrae si detecta claves duplicadas en el índice exclusivo. Este error se devuelve aunque haya una tabla de excepciones para la tabla que se procesa.

Este escenario puede producirse cuando se dan las circunstancias siguientes:

- Se ejecuta la sentencia SET INTEGRITY después de un mandato LOAD para la tabla, y en la operación de carga se ha especificado el modificador de tipo de archivo GENERATEDOVERRIDE o IDENTITYOVERRIDE. Para evitar que se produzca este escenario, se recomienda utilizar el modificador de tipo de archivo GENERATEDIGNORE o GENERATEDMISSING en lugar de GENERATEDOVERRIDE, y utilizar el modificador IDENTITYIGNORE o IDENTITYMISSING en lugar de IDENTITYOVERRIDE. Si se utilizan los modificadores recomendados, no será necesario procesar columnas generadas por expresión o columnas de identidad durante la ejecución de la sentencia SET INTEGRITY.
- Se ejecuta la sentencia SET INTEGRITY después de que una sentencia ALTER TABLE modifique la expresión de una columna generada por expresión.

Para sacar una tabla del estado pendiente de establecer integridad después de producirse un escenario de este tipo:

- No utilice la opción FORCE GENERATED o GENERATE IDENTITY para volver a generar los valores de columna. Utilice, en cambio, la opción IMMEDIATE CHECKED con la opción FOR EXCEPTION para mover las filas en que se produce una violación de la expresión de columna generada a una

tabla de excepciones. A continuación, vuelva a insertar las filas en la tabla a partir de la tabla de excepciones, con lo que se generará la expresión correcta y se realizará la comprobación de la clave exclusiva. Esta operación evita tener que procesar nuevamente toda la tabla, ya que únicamente deben procesarse de nuevo las filas en que se ha producido una violación de la expresión de columna generada.

- Si la tabla que se procesa tiene particiones enlazadas, desenlace esas particiones realizando las acciones que se describen en el punto anterior. A continuación, vuelva a enlazar las particiones y ejecute una sentencia SET INTEGRITY para efectuar el proceso de integridad en las particiones enlazadas por separado.
- Si se especifica una tabla protegida para la sentencia SET INTEGRITY junto con una tabla de excepciones, deben satisfacerse todos los criterios de tabla siguientes; de lo contrario, se devuelve un error (SQLSTATE 428A5):
  - Las tablas deben estar protegidas con la misma política de seguridad.
  - Si una columna de la tabla protegida tiene el tipo de datos DB2SECURITYLABEL, la columna correspondiente de la tabla de excepciones también debe tener el tipo de datos DB2SECURITYLABEL.
  - Si una columna de la tabla protegida está protegida con una etiqueta de seguridad, la columna correspondiente de la tabla de excepciones también debe estar protegida con la misma etiqueta de seguridad.
- Las filas que infringen la integridad que se comprueba en una tabla temporal de período de sistema no se puede desplazar a una tabla de excepción. Si las filas infractoras deben moverse a una tabla de excepción, la tabla debe modificarse para descartar la creación de versiones antes de emitir la sentencia SET INTEGRITY con la cláusula FOR EXCEPTION.
- *Alternativas de sintaxis:* Las siguientes alternativas se admiten para mantener la compatibilidad con versiones anteriores de productos DB2 y otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
  - SET CONSTRAINTS puede especificarse en lugar de SET INTEGRITY
  - SUMMARY puede especificarse en lugar de MATERIALIZED QUERY

## Ejemplos

- *Ejemplo 1:* El siguiente es un ejemplo de una consulta que proporciona información acerca del estado pendiente de establecer integridad y los estados de restricción de acceso relacionados con establecer integridad de las tablas. SUBSTR se utiliza para extraer bytes individuales de la columna CONST\_CHECKED de SYSCAT.TABLES. El primer byte representa las restricciones de clave foránea; el segundo byte representa las restricciones de comprobación; el quinto byte representa la integridad de tabla de consulta materializada; el sexto byte representa las restricciones de columna generada; el séptimo byte representa la integridad de tabla de etapas, y el octavo byte representa las restricciones de particionamiento de datos. STATUS proporciona el estado pendiente de establecer integridad y ACCESS\_MODE proporciona el estado de restricción de acceso relacionado con establecer integridad.

```
SELECT TABNAME, STATUS, ACCESS_MODE,
 SUBSTR(CONST_CHECKED,1,1) AS FK_CHECKED,
 SUBSTR(CONST_CHECKED,2,1) AS CC_CHECKED,
 SUBSTR(CONST_CHECKED,5,1) AS MQT_CHECKED,
 SUBSTR(CONST_CHECKED,6,1) AS GC_CHECKED,
 SUBSTR(CONST_CHECKED,7,1) AS STG_CHECKED,
 SUBSTR(CONST_CHECKED,8,1) AS DP_CHECKED
FROM SYSCAT.TABLES
```

## SET INTEGRITY

- *Ejemplo 2:* Coloque la tabla PARENT en estado sin acceso pendiente de establecer integridad y aplique en cascada de inmediato el estado pendiente de establecer integridad a sus descendientes.

```
SET INTEGRITY FOR PARENT OFF
NO ACCESS CASCADE IMMEDIATE
```

- *Ejemplo 3:* Coloque la tabla PARENT en estado de acceso de lectura pendiente de establecer integridad sin aplicar en cascada de inmediato el estado pendiente de establecer integridad a sus descendientes.

```
SET INTEGRITY FOR PARENT OFF
READ ACCESS CASCADE DEFERRED
```

- *Ejemplo 4:* Compruebe la integridad de una tabla denominada FACT\_TABLE. Si no se detecta ninguna violación de integridad, la tabla se saca del estado pendiente de establecer integridad. Si se detecta alguna violación de integridad, toda la sentencia se retrotrae y la tabla permanece en estado pendiente de establecer integridad.

```
SET INTEGRITY FOR FACT_TABLE IMMEDIATE CHECKED
```

- *Ejemplo 5:* Compruebe la integridad de las tablas SALES y PRODUCTS y mueva las filas en que se produce una violación de integridad a las tablas de excepciones denominadas SALES\_EXCEPTIONS y PRODUCTS\_EXCEPTIONS. Las tablas SALES y PRODUCTS se quitan del estado de pendiente de establecimiento de integridad, tanto si existen violaciones de integridad como si no existen.

```
SET INTEGRITY FOR SALES, PRODUCTS IMMEDIATE CHECKED
FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS,
IN PRODUCTS USE PRODUCTS_EXCEPTIONS
```

- *Ejemplo 6:* Habilite la comprobación de restricciones FOREIGN KEY en la tabla MANAGER y la comprobación de restricciones CHECK en la tabla EMPLOYEE de modo que se eludan con la opción IMMEDIATE UNCHECKED.

```
SET INTEGRITY FOR MANAGER FOREIGN KEY,
EMPLOYEE CHECK IMMEDIATE UNCHECKED
```

- *Ejemplo 7:* Añada una restricción de comprobación y una clave foránea a la tabla EMP\_ACT utilizando dos sentencias ALTER TABLE. La sentencia SET INTEGRITY con la opción OFF se utiliza para colocar la tabla en estado pendiente de establecer integridad, de modo que las restricciones no se comprueban de inmediato al ejecutarse las dos sentencias ALTER TABLE. La única sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED se utiliza para comprobar las dos restricciones añadidas en un solo recorrido por la tabla.

```
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
FOR EXCEPTION IN EMP_ACT USE EMP_ACT_EXCEPTIONS
```

- *Ejemplo 8:* Actualice las columnas generadas con los valores correctos.

```
SET INTEGRITY FOR SALES IMMEDIATE CHECKED
FORCE GENERATED
```

- *Ejemplo 9:* Realice adiciones (utilizando LOAD INSERT) de distintas fuentes a una tabla subyacente (SALES) de una tabla de consulta materializada REFRESH IMMEDIATE (SALES\_SUMMARY). Compruebe la integridad de los datos de SALES de manera incremental y renueve SALES\_SUMMARY de manera incremental. En este escenario, la comprobación de la integridad que se realiza para SALES y la renovación de SALES\_SUMMARY son incrementales, pues el sistema elige el proceso incremental. La opción ALLOW READ ACCESS se

utiliza en la tabla SALES para permitir las lecturas simultáneas de los datos existentes mientras se realiza la comprobación de integridad de la parte cargada de la tabla.

```
LOAD FROM 2000_DATA.DEL OF DEL
INSERT INTO SALES ALLOW READ ACCESS;
LOAD FROM 2001_DATA.DEL OF DEL
INSERT INTO SALES ALLOW READ ACCESS;
SET INTEGRITY FOR SALES ALLOW READ ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS;
REFRESH TABLE SALES_SUMMARY;
```

- *Ejemplo 10:* Enlace una partición nueva a una tabla particionada de datos denominada SALES. Efectúe una comprobación incremental de las violaciones de restricciones en los datos enlazados de la tabla SALES y renueve de modo incremental la tabla SALES\_SUMMARY dependiente. La opción ALLOW WRITE ACCESS se utiliza en ambas tablas para permitir las actualizaciones simultáneas mientras se realiza la comprobación de integridad.

```
ALTER TABLE SALES
ATTACH PARTITION STARTING (100) ENDING (200)
FROM SOURCE;
SET INTEGRITY FOR SALES ALLOW WRITE ACCESS, SALES_SUMMARY ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN SALES
USE SALES_EXCEPTIONS;
```

- *Ejemplo 11:* Desenlace una partición de una tabla particionada de datos denominada SALES. Renueve de manera incremental la tabla SALES\_SUMMARY dependiente.

```
ALTER TABLE SALES
DETACH PARTITION 2000_PART INTO ARCHIVE_TABLE;
SET INTEGRITY FOR SALES_SUMMARY
IMMEDIATE CHECKED;
```

- *Ejemplo 12:* Una nueva tabla de consulta materializada mantenida por el usuario se quita del estado de pendiente de establecimiento de integridad.

```
CREATE TABLE YEARLY_SALES
AS (SELECT YEAR, SUM(SALES) AS SALES
FROM FACT_TABLE GROUP BY YEAR)
DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY USER

SET INTEGRITY FOR YEARLY_SALES
ALL IMMEDIATE UNCHECKED
```

- *Ejemplo 13:* enlazar una partición nueva a una tabla particionada de datos denominada SALES. Supongamos que esta tabla no tiene índices de usuario no particionados. Supongamos también que la comprobación de integridad de los datos, incluida la validación de rangos y otra comprobación de restricciones, ya se haya realizado (a través de una lógica de aplicación independiente del servidor de datos). Optimice el proceso de incorporación de datos utilizando la sentencia SET INTEGRITY...ALL IMMEDIATE UNCHECKED para omitir la comprobación de violaciones de rango y restricciones.

```
ALTER TABLE SALES
ATTACH PARTITION STARTING (300) ENDING (400)
FROM SOURCE_TABLE;
SET INTEGRITY FOR SALES ALL IMMEDIATE UNCHECKED;
```

La tabla SALES sale del estado pendiente de SET INTEGRITY y los datos nuevos están disponibles para que las aplicaciones los utilicen de inmediato.

---

## 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 la 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 esté descrita en el catálogo.

#### RESET

Cierra una sesión de paso a través.

### Notas

- A las fuentes de datos de Microsoft SQL Server, Sybase y Oracle se aplican las restricciones siguientes:
  - Las transacciones definidas por el usuario no pueden utilizarse para fuentes de datos de Microsoft SQL Server y Sybase en modalidad de paso a través, pues Microsoft SQL Server y Sybase restringen qué sentencias de SQL pueden especificarse dentro de una transacción definida por el usuario. Puesto que DB2 no analiza las sentencias de SQL que se procesan en modalidad de paso a través, no es posible detectar si el usuario ha especificado una sentencia de SQL que está permitida dentro de una transacción definida por el usuario.
  - La cláusula COMPUTE no recibe soporte en fuentes de datos de Microsoft SQL Server y Sybase.
  - Las sentencias de DDL no están sujetas a la semántica de transacción en fuentes de datos de Microsoft SQL Server, Oracle y Sybase. Microsoft SQL Server, Oracle o Sybase confirman la operación automáticamente cuando ésta se ha completado. Si se produce una retrotracción, el DDL no se retrotrae.

### 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.

```
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 para que tenga lugar el paso a través para una fuente de datos, cree un índice de clúster 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 1a tabla no es un
 WITH IGNORE DUP KEY"; alias
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.

La sentencia no está bajo el control de la transacción.

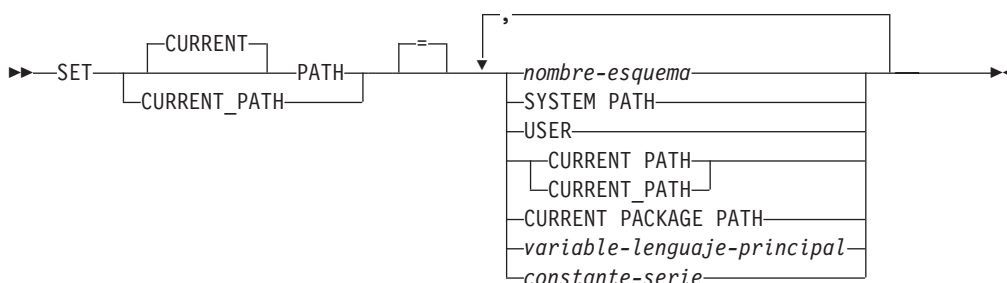
### Invocación

Esta sentencia puede incorporarse 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



### 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, el error no se capturará y podría afectar a las operaciones posteriores de SQL.

#### **SYSTEM PATH**

Este valor equivale a especificar los nombres de esquema "SYSIBM", "SYSFUN", "SYSPROC", "SYSIBMADM".

#### **USER**

El valor del registro especial USER.

#### **CURRENT PATH**

El valor del registro especial CURRENT PATH antes que se ejecute esta sentencia.

#### **CURRENT PACKAGE PATH**

El valor del registro especial CURRENT PACKAGE PATH.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lenguaje-principal* no debe ser mayor 128 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lenguaje-principal* deben alinearse a la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lenguaje-principal*, 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*

Una constante de serie de caracteres con una longitud máxima de 128 bytes.

### Normas

- Un nombre de esquema no puede aparecer más de una vez en la vía de acceso a SQL (SQLSTATE 42732).
- El nombre de esquema SYSPUBLIC no puede especificarse en la vía de acceso de SQL (SQLSTATE 42815).
- 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 mediante una coma. La longitud de la serie resultante no puede exceder de 2048 bytes (SQLSTATE 42907).

### Notas

- El valor inicial del registro especial CURRENT PATH es "SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","X" donde la 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 SQL que se utiliza para resolver nombres de función, nombres de procedimiento, nombres de tipo de datos, nombres de variable global y nombres de objeto de módulo en sentencias de SQL dinámico. La opción de vinculación FUNCPATH especifica la vía de acceso SQL que se debe utilizar para resolver los nombres de función, los nombres de procedimiento, los nombres de tipo de datos, los nombres de variable global y los nombres de objeto de módulo en las sentencias de SQL estático.
- *Alternativas de sintaxis:* Las siguientes alternativas de sintaxis reciben soporte para mantener la compatibilidad con las versiones anteriores de los productos DB2 y con otros productos de base de datos. Estas alternativas no son estándar y no deberían utilizarse.
  - CURRENT FUNCTION PATH puede utilizarse en lugar de CURRENT PATH

### Ejemplos

- *Ejemplo 1:* La sentencia siguiente establece el registro especial CURRENT 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 ROLE

La sentencia SET ROLE comprueba si el ID de autorización de la sesión es un miembro de un rol específico. Un ID de autorización llega a ser miembro de un rol cuando dicho rol se otorga al ID de autorización o a un grupo o rol en el que es miembro el ID de autorización.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

No se necesita.

### Sintaxis

►► SET ROLE  nombre-rol ◀◀

### Descripción

*nombre-rol*

Especifica un ID de rol en la que ha de verificarse su pertenencia como miembro del ID de autorización de la sesión. El *nombre-rol* debe identificar un rol existente en el servidor actual (SQLSTATE 42704). Si el ID de autorización de la sesión no es miembro de *nombre-rol*, se devuelve un error (SQLSTATE 42501).

### Notas

- Todos los roles que se han otorgado a un ID de autorización se utilizan para la comprobación de autorizaciones. La sentencia SET ROLE no afecta a los roles que se utilizan para esta comprobación de autorizaciones. Utilice las sentencias GRANT ROLE y REVOKE ROLE para cambiar los roles de los que es miembro un ID de autorización.

### Ejemplos

- *Ejemplo 1:* Al usuario WALID se le ha otorgado el rol EDITOR, pero no el rol AUTHOR. Compruebe que WALID sea miembro del rol EDITOR.

```
SET ROLE EDITOR
```

- *Ejemplo 2:* Compruebe que WALID no sea miembro del rol AUTHOR. La siguiente sentencia devuelve un error (SQLSTATE 42501).

```
SET ROLE AUTHOR
```



## SET SCHEMA

Los caracteres de la *variable-lenguaje-principal* deben alinearse a la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lenguaje-principal*, 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*

Una constante de serie de caracteres con una longitud máxima de 128 bytes.

## Normas

- Si el valor especificado no se adapta a las normas 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 dinámico, 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 QUALIFIER especifica el nombre de esquema que debe utilizarse como calificador para los nombres de objetos de base de datos no calificados en las sentencias de SQL estático.

## 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

La sentencia SET SERVER OPTION especifica un valor de opción de servidor que debe 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 la transacción.

### Invocación

Esta sentencia puede emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

No se necesita.

### 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 debe establecer.

**TO** *constante-serie*

Especifica un valor para *nombre-opción-servidor* como una constante de serie de caracteres.

**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.
- 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.
- SYSCAT.SERVEROPTIONS no se actualizará basándose en una sentencia SET SERVER OPTION, pues este cambio sólo afecta a la conexión actual.
- En el caso del SQL estático, si se utiliza la sentencia SET SERVER OPTION sólo se afecta a la ejecución de la sentencia de SQL estático. Si se utiliza la sentencia SET SERVER OPTION, no tiene ningún efecto en los planes generados por el optimizador.

### Ejemplos

- *Ejemplo 1:* Se ha definido una fuente de datos Oracle denominado ORASERV para una base de datos federada denominada DJDB. ORASERV se ha configurado para inhabilitar la indicación de planes. Sin embargo, el DBA

## SET SERVER OPTION

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 oraserv");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"select c1 from ora_t1 where c1 > 100"); /*Generar indicac. plan*/
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 que una aplicación se conecta con la base de datos federada para acceder a una fuente de datos de Oracle 8 específica -uno que se ha definido para 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) en 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 SESSION AUTHORIZATION

La sentencia SET SESSION AUTHORIZATION cambia el valor del registro especial SESSION\_USER.

La sentencia no está bajo el control de la transacción. La sentencia SET SESSION AUTHORIZATION tiene como objetivo proporcionar soporte para un solo usuario, suponiendo que los ID de autorización son distintos en la misma conexión y no se debe utilizar en escenarios en los que distintos usuarios reutilizan la misma conexión, lo que normalmente se denomina agrupación de conexiones.

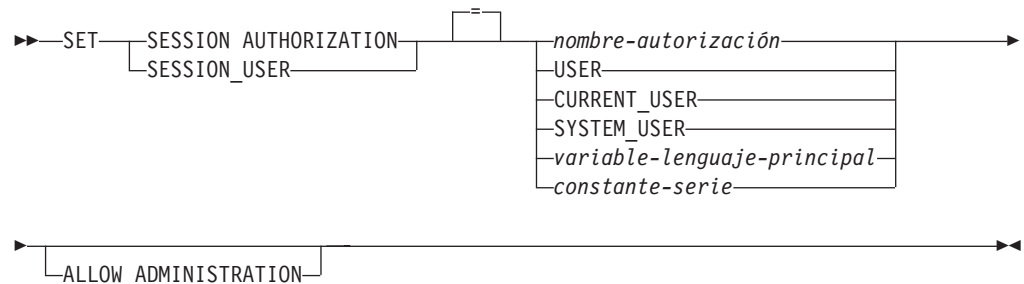
### Invocación

La sentencia puede incorporarse en un programa de aplicación o 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 incluir SETSESSIONUSER en el valor de ID de autorización en el que se establece el registro especial.

### Sintaxis



### Descripción

#### *nombre-autorización*

Especifica el ID de autorización que se debe utilizar como el nuevo valor para el registro especial SESSION\_USER.

#### **USER**

El valor del registro especial USER.

#### **CURRENT\_USER**

El valor del registro especial CURRENT USER.

#### **SYSTEM\_USER**

El valor del registro especial SYSTEM\_USER.

#### *variable-lenguaje-principal*

Una variable de tipo CHAR o VARCHAR. La longitud del contenido de *variable-lenguaje-principal* no debe ser mayor 128 bytes (SQLSTATE 28000). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable de indicador, el valor de dicha variable de indicador no debe indicar un valor nulo (SQLSTATE 28000).

## SET SESSION AUTHORIZATION

Los caracteres de la *variable-lenguaje-principal* deben alinearse a la izquierda. Cuando se especifica un *nombre-autorización* con una variable del lenguaje principal, todos los caracteres se deben especificar en mayúsculas, porque no hay conversión a caracteres en mayúsculas.

### *constante-serie*

Una constante de serie de caracteres con una longitud máxima de 128 bytes.

### **ALLOW ADMINISTRATION**

Especifica que pueden especificarse sentencias de esquema de SQL antes que esta sentencia en la misma unidad de trabajo.

### **Normas**

- El valor especificado para el registro especial SESSION\_USER que cumple con las normas de un ID de autorización de tipo USER (SQLSTATE 42602).
- La opción de vinculación OWNER especifica el ID de autorización que se debe utilizar para sentencias de SQL estático.
- Esta sentencia sólo se puede emitir como la primera sentencia (que no sea una sentencia de registro especial SET) en una nueva unidad de trabajo sin ningún cursor WITH HOLD abierto (SQLSTATE 25001). Esta restricción incluye cualquier petición PREPARE correspondiente a una sentencia que no sea una sentencia de registro especial SET.
- El valor del registro especial SESSION\_USER se utiliza como el ID de autorización para todas las sentencias de SQL dinámico de un paquete vinculado con la opción de vinculación DYNAMICRULES(RUN). (Esto incluye INVOKERUN y DEFINERUN cuando el paquete no lo utiliza ninguna rutina). Si un paquete utiliza autorización de propietario, invocador o definidor basado en la opción DYNAMICRULES, esta sentencia no tiene ningún efecto sobre las sentencias de SQL dinámico emitidas desde dentro del paquete.

### **Notas**

- La sentencia SET SESSION AUTHORIZATION le permite modificar el ID de autorización de sesión. El ID de autorización de sesión representa el usuario actual de la conexión y es el ID de autorización que DB2 tiene en cuenta para todos los controles de autorizaciones relativos al SQL dinámico dentro de un paquete de ejecución DYNAMICRULES. El registro especial SESSION\_USER se puede utilizar para visualizar el valor actual de este ID de autorización de sesión.
- El valor inicial del registro especial SESSION\_USER para una nueva conexión es el mismo que el valor del registro especial SYSTEM\_USER.
- La información de grupo correspondiente al ID de autorización de sesión especificado en esta sentencia se adquiere en el momento de ejecutar la sentencia.
- El hecho de establecer el registro especial SESSION\_USER no afecta a los registros especiales CURRENT SCHEMA ni CURRENT PATH.
- Si se produce algún error durante el establecimiento del registro especial SESSION\_USER, el registro vuelve a tomar su valor anterior.
- Esta sentencia no se debe utilizar para permitir que varios usuarios diferentes reutilicen la misma conexión, puesto que cada usuario heredará la capacidad de cambiar el valor del registro especial SESSION\_USER que tenía el propietario original de la conexión. Esta sentencia depende del valor de SYSTEM\_USER correspondiente a la comprobación de privilegios y la sentencia SET SESSION AUTHORIZATION no modifica el ID de autorización de conexión inicial.

Además, esta sentencia no está relacionada con los siguientes comportamientos que afectan a la reutilización de la conexión:

- El privilegio CONNECT no se comprueba para el nuevo ID de autorización
- El contenido de cualquier registro especial actualizable no se restablece; en concreto, el contenido del registro especial ENCRYPTION PASSWORD no se modifica y está disponible para el nuevo ID de autorización para las funciones de cifrado y descifrado.
- El contenido de cualquier tabla temporal global declarada no se ve afectado y resulta accesible para el nuevo ID de autorización
- Los enlaces existentes con servidores remotos no se restablecerán
- Si se especifica la cláusula ALLOW ADMINISTRATION, los siguientes tipos de sentencias u operaciones pueden preceder la sentencia SET SESSION AUTHORIZATION:
  - Lenguaje de definición de datos (DDL), incluida la definición de los puntos de salvaguarda y la declaración de tablas temporales globales, pero no incluyendo SET INTEGRITY
  - Sentencias GRANT y REVOKE
  - LOCK TABLE, sentencia
  - Sentencias COMMIT y ROLLBACK
  - SET de registros especiales
  - SET de variables globales

### Ejemplos

- *Ejemplo 1:* La siguiente sentencia establecer el registro especial SESSION\_USER.  
**SET SESSION\_USER = RAJIV**
- *Ejemplo 2:* Establezca el ID de autorización de sesión (el registro especial SESSION\_USER) de modo que sea un valor del ID de autorización del sistema, que es el ID que ha establecido la conexión en la que se ha emitido la sentencia.  
**SET SESSION AUTHORIZATION SYSTEM\_USER**

## SET USAGE LIST STATE

La sentencia SET USAGE LIST STATE gestiona el estado de una lista de uso y de los datos y la memoria asociados.

La sentencia no está bajo el control de la transacción.

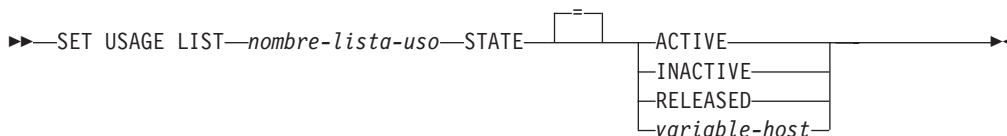
### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización DBADM o SQLADM.

### Sintaxis



### Descripción

#### nombre-lista-uso

Identifica la lista de uso. El *nombre-lista-uso*, incluido el calificador implícito o explícito, debe identificar una lista de uso descrita en el catálogo (SQLSTATE 42704).

#### ACTIVE

Indica que la lista de uso está activada para la supervisión. La memoria de la lista de uso se asigna cuando una sección hace referencia por primera vez a la tabla o al índice. Si la lista de uso es para una tabla o índice particionados, la memoria de la lista de uso se asigna cuando una sección hace referencia por primera vez a la tabla o al índice. En un entorno de base de datos particionada o un entorno DB2 pureScale, se asigna memoria en cada miembro. Si la lista de uso ya está en estado ACTIVO, se devuelve un aviso (SQLSTATE 01598).

Al activarse, se eliminan los datos de la lista de uso y la recopilación empieza por el principio de la lista.

#### INACTIVE

Indica que la lista de uso está desactivada para la supervisión. Si el estado de una lista de uso ya se ha establecido, esta palabra clave se ignorará. Si el estado de la lista de uso para una tabla o índice particionados se establece en INACTIVO, el estado de la lista de uso para cada partición se establecerá en INACTIVO. Si el estado ya es INACTIVO, se ignorará la palabra clave. De forma similar, en un entorno de base de datos particionada o en un entorno DB2 pureScale, el estado de la lista de uso para cada miembro se establece en INACTIVO. Si el estado ya es INACTIVO, se ignorará la palabra clave.

Los datos recopilados en la lista no se eliminarán cuando el estado de la lista de uso se establezca en INACTIVO.

**RELEASED**

Indica la memoria asociada a una lista de uso se ha liberado. Si el estado de la lista de uso para una tabla o índice particionados se establece en RELEASED, se publicará la memoria asociada con cada partición de datos. En un entorno de base de datos particionada o en un entorno DB2 pureScale, se publicará la memoria asociada con cada miembro.

**Notas**

- **Estado actual determinante:** el estado actual de una lista de uso se determina utilizando la función incorporada MON\_GET\_USAGE\_LIST\_STATUS.
- **Consideraciones para DB2 pureScale o entornos de bases de datos particionadas:** si se activa una lista de uso para una tabla o índice particionados, se asignará memoria a cada partición de datos. De forma similar, en un entorno de base de datos particionada o en un entorno DB2 pureScale, se asignará memoria a cada miembro activo.
- **Asignación de memoria para miembros no disponibles:** si un miembro no está disponible en el momento de la activación, la memoria se asigna cuando el miembro se vuelve a activar (si el estado de la lista de uso todavía está establecido en activo). Esto también se aplica cuando se añade un miembro al clúster.
- **Asignación de memoria a particiones de datos que se están añadiendo o adjuntando:** para las particiones de datos que se están añadiendo o adjuntando, la memoria asociada con la lista de uso para la partición recién añadida o adjuntada se asigna cuando se ejecuta la siguiente sección que hace referencia a la tabla o al índice particionados.
- **Establecimiento de INACTIVE independientemente:** si la lista de uso se ha creado con la propiedad, WHEN FULL DEACTIVATE, el estado de la lista de uso de cada partición de datos o miembro se establece en INACTIVE independientemente.
- **Reactivación implícita de una lista de uso activa:** Si el estado de una lista de uso INACTIVE ON START DATABASE se establece en ACTIVE en un entorno de base de datos particionada o en un entorno DB2 pureScale, su comportamiento es similar al de ACTIVE ON START DATABASE hasta que la lista de uso se desactiva explícitamente o hasta que la instancia se recicla. Es decir, si el estado de una lista de uso está activo cuando se desactiva un miembro de base de datos y el miembro de la base de datos se vuelve activar posteriormente, la lista de uso para este miembro también se vuelve a activar de forma implícita.
- **Definición del estado publicado:** una lista de uso se considera en estado liberado si se ha definido y no se ha activado (explícita o automáticamente) o se ha liberado utilizando la sentencia SET USAGE LIST STATE. La función MON\_GET\_USAGE\_LIST\_STATUS no devuelve las listas de uso en el estado publicado.
- **Estados pendientes de activación, activos y anómalos:** si se activa una lista de uso (explícita o automáticamente), el estado de la lista de uso se establece en pendiente de activación y la memoria se asigna cuando la sección hace referencia por primera vez a la tabla o al índice. Llegado este punto, el estado de la lista de uso se establece en activo. Si la memoria de la lista de uso no se puede asignar, el estado de la lista de uso se establece en anómalo y se debe activar explícitamente utilizando la sentencia SET USAGE LIST STATE.
- **Las listas de uso inactivas permanecen inactivas cuando se reactiva el miembro de la base de datos:** Si el estado de una lista de uso ACTIVE ON START DATABASE se establece en INACTIVE en un entorno de base de datos particionada o en un entorno DB2 pureScale, su comportamiento es similar al de

## SET USAGE LIST STATE

INACTIVE ON START DATABASE hasta que la lista de uso se activa explícitamente o hasta que se recicla la instancia. Es decir, si el estado de una lista de uso está inactivo cuando un miembro de la base de datos se desactiva o está fuera de línea y un miembro de la base de datos se vuelve a activar posteriormente, el estado de la lista de uso para este miembro permanecerá inactivo.

- **Activación, desactivación o publicación de una lista de uso para una tabla o índice particionados:** si se activa, desactiva o publica una lista de uso para una tabla o índice particionados, el cambio de estado se aplica a las particiones de datos. De forma similar, en un entorno de base de datos particionada o en un entorno DB2 pureScale, el cambio de estado se aplica a todos los miembros.
- **Consideraciones sobre el tamaño de la lista de uso:** al activarse, la memoria asociada con la lista de uso se asigna desde el almacenamiento dinámico de supervisión. El valor máximo del tamaño de lista de uso es de 2 MB aproximadamente. Para las tablas e índices particionados, la memoria se asigna a cada partición de datos. Por ejemplo, si una tabla particionada tiene definidas tres particiones de datos, se asignarán aproximadamente 6 MB de memoria total. Por tanto, la activación de varias listas de uso impone más requisitos de memoria en el almacenamiento dinámico de supervisión. Por tanto, se recomienda seleccionar un tamaño de lista razonable o establecer el parámetro de configuración **mon\_heap\_sz** en AUTOMATIC para que el gestor de bases de datos gestione el tamaño del almacenamiento dinámico de supervisión.
- **Colección de datos cuando una lista de uso se establece en INACTIVE** Los datos recopilados en la lista no se eliminarán cuando el estado de la lista de uso se establezca en INACTIVE.
- **Acceso a datos y memoria:** El acceso a los datos de la lista sigue siendo posible (mediante la utilización de las funciones de tabla **MON\_GET\_TABLE\_USAGE\_LIST** y **MON\_GET\_INDEX\_USAGE\_LIST**) siempre que se asigne memoria para la lista.
- **Liberación de memoria:** La memoria que se asocia a la lista de uso se libera cuando se produce uno de los sucesos siguientes.
  - Se descarta la lista de uso.
  - Se descarta la tabla o el índice en el que se ha definido la lista de uso. La memoria que se asigna con el uso se publica para todas las particiones de datos. En un entorno de base de datos particionada o en un entorno DB2 pureScale, se publicará la memoria asociada con cada miembro activo.
  - Cuando una partición de datos se desenlaza de una tabla o índice particionados. Sólo se publica la memoria asociada con la partición de datos.
  - Cuando se desactiva un miembro de base de datos. Sólo se publica la memoria asociada con el miembro.
  - Cuando se desactiva toda la instancia o base de datos. Los datos de la lista de uso no permanecen cuando la base de datos se desactiva y se reinicia.
  - Cuando la memoria asociada con la lista de uso se publica explícitamente utilizando la sentencia **SET USAGE LIST STATE**.

---

## SET variable

La sentencia SET variable asigna valores a variables.

La sentencia no está bajo el control de la transacción.

### Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Para hacer referencia a una variable de transición, los privilegios del ID de autorización del creador de activadores debe tener, como mínimo, una de las autorizaciones siguientes:

- Privilegio UPDATE para las columnas a las que se hace referencia en la parte izquierda de la asignación y privilegio SELECT para las columnas a las que se hace referencia en la parte derecha.
- Privilegio CONTROL para la tabla (tabla sujeto del activador)
- Autorización DATAACCESS

Si se hace referencia a una variable global en la parte derecha de la sentencia de asignación, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes:

- el privilegio READ sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

Si se asigna un valor a una variable global en la parte izquierda de la sentencia de asignación, los privilegios del ID de autorización de la sentencia deben incluir una de las autorizaciones siguientes:

- el privilegio WRITE sobre la variable global que no está definida en un módulo
- el privilegio EXECUTE sobre el módulo de la variable global que está definida en un módulo

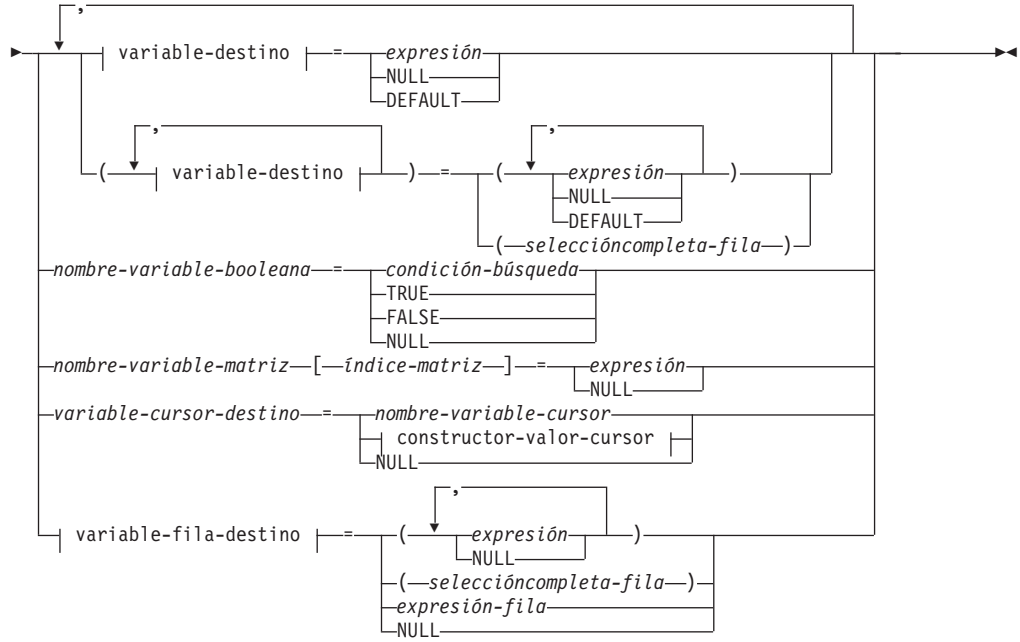
Para ejecutar esta sentencia con una *selección-completa-fila* como la parte derecha de la asignación, los privilegios del ID de autorización de la sentencia deben incluir los privilegios necesarios para ejecutar la *selección-completa-fila*. Consulte la sección Autorización en "Consultas de SQL".

Para ejecutar esta sentencia con un *constructor-valor-cursor* que utiliza una *sentencia-select*, los privilegios del ID de autorización de la sentencia deben incluir los privilegios necesarios para ejecutar la *sentencia-select*. Consulte la sección Autorización en "Consultas de SQL".

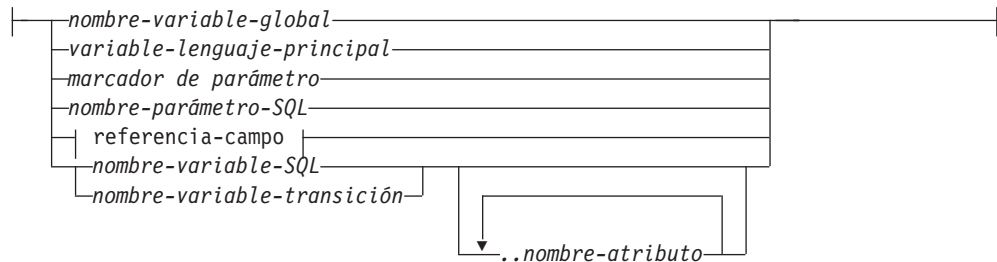
### Sintaxis

►►—SET—►►

# SET variable



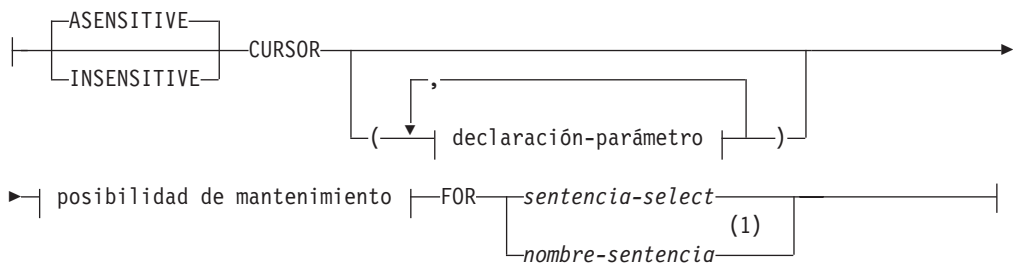
## variable-destino:



## referencia-campo:



## constructor-valor-cursor:



## declaración-parámetro:



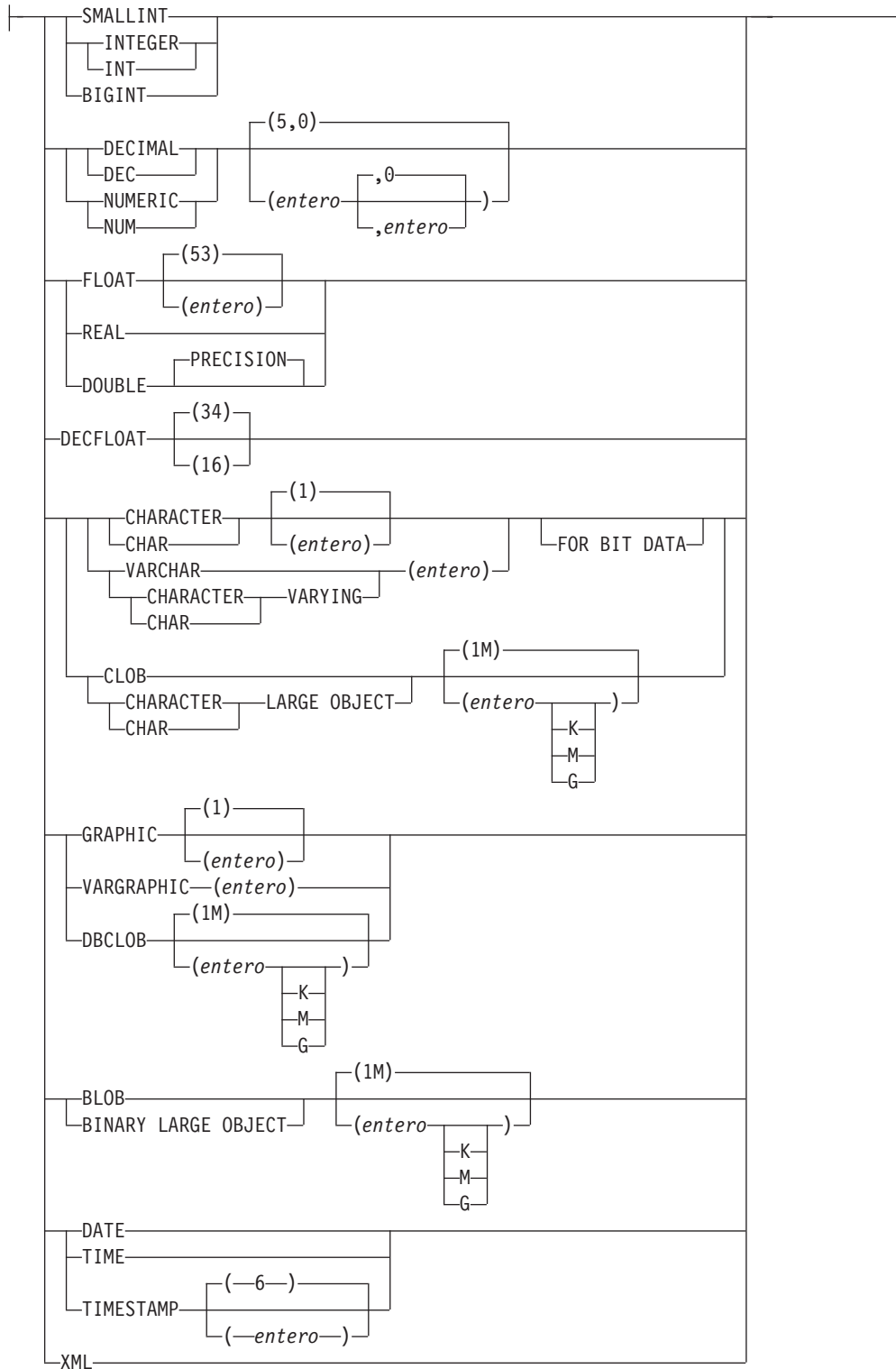
| *nombre-parámetro* | tipo-datos | \_\_\_\_\_ |

**tipo-datos:**

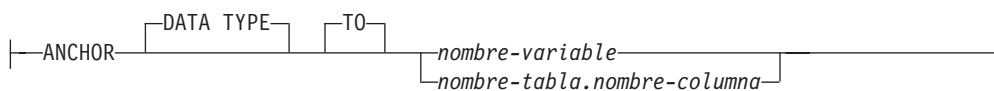
| *tipo-incorporado* \_\_\_\_\_ |  
| tipo-datos-parámetro-anclado |  
| *nombre-tipo-diferenciado* \_\_\_\_\_ |

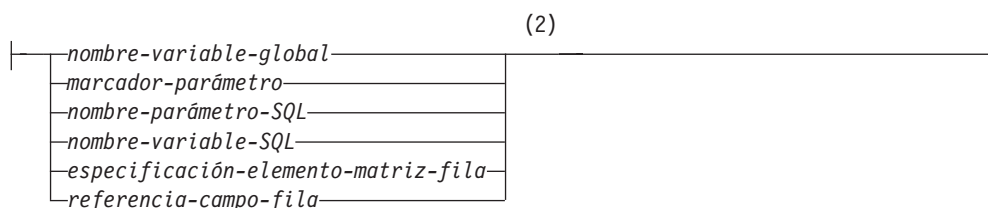
**tipo-incorporado:**

# SET variable



## tipo-datos-parámetro-anclado:



**posibilidad de mantenimiento:****variable-fila-destino:****Notas:**

- 1 El *nombre-sentencia* no se puede especificar si se ha especificado la *declaración-parámetro*.
- 2 El tipo de datos debe ser un tipo de fila.

**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-global*

Identifica la variable global que es el sujeto de la asignación. El *nombre-variable-global* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

*variable-lenguaje-principal*

Identifica la variable del lenguaje principal que es el sujeto de la asignación.

*marcador-parámetro*

Identifica el marcador de parámetro que es el sujeto de la asignación.

*nombre-parámetro-SQL*

Identifica el parámetro que es el sujeto de la asignación. El parámetro se debe especificar en la *declaración-parámetro* de la sentencia CREATE PROCEDURE.

*referencia-campo*

Identifica el campo de un valor de tipo de fila que es el destino de la asignación.

*nombre-variable-fila*

El nombre de una variable con un tipo de datos que es un tipo de fila.

*nombre-campo*

El nombre de un campo dentro de un tipo de fila.

*nombre-variable-SQL*

Identifica la variable de SQL que es el sujeto de la asignación. Las variables de SQL se deben declarar antes de utilizarlas.

## SET variable

### *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 activador, calificado opcionalmente por un nombre de correlación que identifique el nuevo valor (SQLSTATE 42703).

### *..nombre-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-atributo* se denomina *asignación convencional*.

### *expresión*

Indica el nuevo valor del destino de la asignación. La expresión es cualquier expresión del tipo que se describe en el apartado "Expresiones". La expresión no puede incluir una función agregada salvo cuando aparece 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 y NEW. Las variables de transición debe calificarlas el *nombre-correlación* (SQLSTATE 42702).

### NULL

Especifica el valor nulo. Si el destino de la asignación es una variable de fila, se asigna a cada campo el valor nulo. NULL no puede ser el valor en una asignación de atributos a menos que se convirtiera específicamente al tipo de datos del atributo (SQLSTATE 429B9).

### DEFAULT

Especifica que se debe utilizar el valor por omisión.

En procedimientos SQL, la cláusula DEFAULT puede especificarse sólo para sentencias de SQL estático. La excepción es que la cláusula DEFAULT puede especificarse cuando la *variable-destino* es una variable global de una sentencia de SQL dinámico.

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 que se ha definido para la columna (consulte el apartado *cláusula-valor-por-omisión* en "ALTER TABLE").
- 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 la cláusula WITH DEFAULT, la cláusula IDENTITY ni la cláusula NOT NULL, 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 la *variable-destino* es una variable de SQL, el valor que se ha insertado es el valor por omisión, tal como se ha especificado o está implícito en la declaración de variable.

Si la *variable-destino* es una variable global, el valor que se ha insertado es el valor por omisión, tal como se ha especificado en la creación de la variable.

Si la *variable-destino* es una variable de SQL o un parámetro de SQL de un procedimiento de SQL, una variable del lenguaje principal o un marcador de parámetro, no puede especificarse la palabra clave DEFAULT (SQLSTATE 42608).

#### *selección-completa-fila*

Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de variables de destino campos de la variable de fila especificados para la asignación. Los valores se asignan a cada campo o variable de destino correspondiente. Si la selección completa de fila da como resultado ninguna fila, se asignan valores nulos a las variables de destino de la lista o, en el caso de la asignación a una variable de fila, se asigna un valor nulo único. En el contexto de una sentencia CREATE TRIGGER, una *selección-completa-fila* puede contener referencias a las variables de transición OLD y NEW, que se deben calificar por 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).

#### *nombre-variable-booleana*

Identifica una variable o un parámetro de SQL o una variable global. La variable o el parámetro debe ser de tipo booleano (SQLSTATE 428H0). La sentencia SET se debe emitir dentro de una sentencia de SQL compuesto (compilado) (SQLSTATE 428H2).

#### *condición-búsqueda*

Condición de búsqueda cuyo resultado es verdadero, falso o desconocido. Se devuelve el resultado de desconocido como valor booleano NULL.

#### **TRUE**

Especifica el valor booleano TRUE.

#### **FALSE**

Especifica el valor booleano FALSE.

#### **NULL**

Especifica el valor booleano NULL.

#### *nombre-variable-matriz*

Identifica una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz (SQLSTATE 428H0).

#### **[índice-matriz]**

Expresión que especifica qué elemento de la matriz será el destino de la asignación. Para una matriz común, el índice-matriz debe poderse asignar a INTEGER (SQLSTATE 22018 o 428H1). Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz y no puede ser el valor nulo (SQLSTATE 2202E).

Para una matriz asociativa, la expresión de índice de matriz se debe poder asignar al tipo de datos de índice de la matriz asociativa (SQLSTATE 22018 o 428H1) y no puede ser el valor nulo (SQLSTATE 2202E).

#### *variable-cursor-destino*

Identifica una variable de cursor. El tipo de datos de una *variable-cursor-destino* tiene que ser un tipo de cursor (SQLSTATE 42821).

#### *nombre-variable-cursor*

Identifica una variable de cursor del mismo tipo de cursor que la *variable-cursor-destino*.

### *constructor-valor-cursor*

Un *constructor-valor-cursor* especifica la *sentencia-select* asociada con la variable de destino. La asignación de un *constructor-valor-cursor* a una variable de cursor define el cursor subyacente de esa variable de cursor.

### **ASENSITIVE o INSENSITIVE**

Especifica si el cursor es sensible o insensible a los cambios. Consulte el apartado "DECLARE CURSOR" para obtener más información. El valor por omisión es ASENSITIVE.

### **ASENSITIVE**

Especifica que el cursor debe ser todo lo sensible que sea posible para las inserciones, actualizaciones o supresiones realizadas en las filas subyacentes de la tabla de resultados, en función de cómo se haya optimizado *sentencia-select*. ASENSITIVE es el valor por omisión.

### **INSENSITIVE**

Especifica que el cursor no debe contar con sensibilidad para las inserciones, actualizaciones o supresiones que se han realizado en las filas subyacentes de la tabla de resultados. Si se especifica INSENSITIVE, el cursor es de sólo lectura y la tabla de resultados se materializa al abrirse el cursor. Como resultado de ello, el tamaño de la tabla de resultados, el orden de las filas y los valores de cada fila no cambian tras abrirse el cursor. La sentencia SELECT no puede contener una cláusula FOR UPDATE, y el cursor no puede utilizarse para las actualizaciones o supresiones posicionadas.

### *(declaración-parámetro, ...)*

Especifica los parámetros de entrada del cursor, incluido el nombre y el tipo de datos de cada parámetro. Sólo se pueden especificar parámetros de entrada con nombre si la *sentencia-select* también está especificada en el *constructor-valor-cursor* (SQLSTATE 428HU).

### *nombre-parámetro*

Asigna un nombre al parámetro de cursor que se debe utilizar como variable de SQL dentro de la *sentencia-select*. El nombre no puede ser igual que ningún otro nombre de parámetro del cursor. Los nombres deben elegirse también evitando nombres de columna que se puedan utilizar en la *sentencia-select*, ya que los nombres de columna se resuelven antes que los nombres de parámetro.

### *tipo-datos*

Especifica el tipo de datos del parámetro de cursor utilizado dentro de una *sentencia-select*. No se pueden especificar tipos estructurados ni tipos de referencia (SQLSTATE 429BB).

### *tipo-incorporado*

Especifica un tipo de datos incorporado. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

### *tipo-datos-parámetro-anclado*

Identifica otro objeto que se utiliza para determinar el tipo de datos del parámetro de cursor. El tipo de datos del objeto de anclaje se ve afectado por las mismas limitaciones que se aplican cuando se especifica el tipo de datos directamente.

### **ANCHOR DATA TYPE TO**

Indica que se utiliza un tipo de datos anclados para especificar el tipo de datos.

*nombre-variable*

Identifica una variable de SQL local, un parámetro SQL o una variable global. El tipo de datos de la variable a la que se hace referencia se utiliza como tipo de datos para el parámetro de cursor.

*nombre-tabla.nombre-columna*

Identifica un nombre de columna de una tabla o vista existente. El tipo de datos de la columna se utiliza como tipo de datos para el parámetro de cursor.

*nombre-tipo-diferenciado*

Especifica el nombre de un tipo diferenciado. Si se especifica el *nombre-tipo-diferenciado* sin un nombre de esquema, el tipo diferenciado se resuelve buscando en los esquemas de la vía de acceso de SQL.

*posibilidad de mantenimiento*

Especifica si se impedirá que el cursor se cierre como consecuencia de una operación de confirmación. Consulte el apartado "DECLARE CURSOR" para obtener más información. El valor por omisión es WITHOUT HOLD.

**WITHOUT HOLD**

No impide que el cursor se cierre como consecuencia de una operación de confirmación.

**WITH HOLD**

Mantiene recursos en varias unidades de trabajo. Impide que el cursor se cierre como consecuencia de una operación de confirmación.

*sentencia-select*

Especifica la sentencia SELECT del cursor. Consulte el apartado "sentencia-select" para obtener más información. Si se incluye una *declaración-parámetro* en el *constructor-valor-cursor*, la *sentencia-select* no debe incluir ninguna variable de SQL local ni parámetros de SQL de rutina (SQLSTATE 42704).

*nombre-sentencia*

Especifica la *sentencia-select* preparada del cursor. Consulte el apartado "PREPARE" para obtener una explicación de las sentencias preparadas. La variable del cursor de destino no debe tener un tipo de datos que sea un tipo de cursor definido por el usuario con tipo firme (SQLSTATE 428HU). No se deben especificar parámetros de entrada con nombre en el *constructor-valor-cursor* si se especifica un *nombre-sentencia* (SQLSTATE 428HU).

*variable-fila-destino*

Identifica la variable de fila de destino de la asignación. El tipo de datos debe ser un tipo de fila.

*expresión-fila*

Especifica el nuevo valor de fila para el destino de la asignación. Puede ser cualquier expresión de fila del tipo descrito en "Expresión de fila". El número de campos de la fila debe coincidir con el destino de la asignación, y cada campo de la fila debe poderse asignar al campo correspondiente en el destino de la asignación. Si los valores fuente y de destino son un tipo de fila definido por el usuario, los nombres de los tipos deben ser iguales (SQLSTATE 42821).

### Normas

- El número de valores que se debe asignar a partir de las expresiones, NULL, DEFAULT o la *selección-completa-fila* debe coincidir con el número de *variables-destino* especificado para la asignación (SQLSTATE 42802).
- Una sentencia SET variable no puede asignar una variable de SQL y una variable de transición en una sentencia (SQLSTATE 42997).
- Las variables globales no se pueden asignar dentro de activadores que no se han definido mediante una sentencia de SQL compuesto (compilado), funciones que no se han definido mediante una sentencia de SQL compuesto (compilado), métodos o sentencias de SQL compuesto (en línea) (SQLSTATE 428GX).
- Si el valor que se asigna es una matriz resultante de un constructor de matriz o de ARRAY\_AGG, los tipos base de la matriz y de la variable de destino deben ser idénticos (SQLSTATE 42821).
- **Utilización de tipos de datos anclados:** Un tipo de datos anclado no puede hacer referencia a (SQLSTATE 428HS): un apodo, una tabla con tipo, una vista con tipo, una tabla temporal declarada, una definición de fila asociada con un cursor de tipo débil, un objeto con una página de códigos o una clasificación que es diferente de la página de códigos de la base de datos o la asignación de base de datos.
- **Asignaciones que implican variables de cursor:** las asignaciones que hacen referencia a una variable de cursor y que la establecen en el valor de un constructor de valor de cursor sólo se pueden utilizar en sentencias de SQL compuesto (compilado). Todas las sentencias OPEN que utilizan una variable de cursor deben aparecer en el mismo ámbito que la asignación (SQLSTATE 51044).

### Notas

- Se asignan valores a las variables de destino en función de normas de asignación específicas.
- **Sentencia de asignación en procedimientos de SQL:** las sentencias de asignación de los procedimientos de SQL deben ajustarse a las normas de asignación de SQL. Las asignaciones de series utilizan normas de asignación de recuperación.
- **Asignaciones de elementos de matriz:** si la asignación tiene la forma SET A[idx] = rhs, donde A es un nombre de variable de matriz, idx es una expresión utilizada como índice-matriz y rhs es una expresión del mismo tipo que el elemento de matriz, entonces:
  1. Si la matriz A es el valor nulo, defina A en la matriz vacía.
  2. C tiene que ser la cardinalidad de la matriz A.
  3. Si A es una matriz común:
    - Si idx es menor que o igual a C, el valor en la posición que idx identifica se sustituye por el valor de rhs.
    - Si idx es mayor que C, entonces:
      - El valor en posición *i*, dado que *i* es mayor que C y menor que idx, se establece en el valor nulo.
      - El valor en posición idx se establece en el valor de rhs.
      - La cardinalidad de A se establece en idx.
  4. Si A es una matriz asociativa:
    - Si idx coincide con un valor de índice de matriz existente, el valor de elemento cuyo índice de matriz es idx se sustituye por el valor de rhs.
    - Si idx no coincide con ningún valor de índice de matriz existente, entonces:
      - La cardinalidad de A aumenta en 1.



- El nuevo valor de elemento se establece en rhs con el valor de índice de matriz asociado idx.
- 5. Si idx es menor que o igual a C, el valor en la posición que idx identifica se sustituye por el valor de rhs.
- 6. Si idx es mayor que C, entonces:
  - a. El valor en posición *i*, dado que *i* es mayor que C y menor que idx, se establece en el valor nulo.
  - b. El valor en posición idx se establece en el valor de rhs.
  - c. La cardinalidad de A se establece en idx.
- Si una variable se ha declarado con un identificador que coincide con el nombre de un registro especial (por ejemplo, PATH), se debe delimitar la variable para impedir una asignación no intencionada al registro especial (por ejemplo, SET "PATH" = 1; para una variable llamada PATH que se ha declarado como entero).
- Si se incluye más de una asignación, se evalúa cada *expresión y selección-completa-fila* antes de que se realicen las asignaciones. Por lo tanto, las referencias a variables de destino en una expresión o una selección completa de fila siempre son el valor de la variable de destino antes de cualquier asignación en una única sentencia SET.
- Cuando se actualiza una columna de identidad que se ha definido como un tipo diferenciado, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior en el tipo de fuente antes de realizarse el cálculo.)
- Para que el gestor de bases de datos 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 una columna de identidad, y el valor utilizado para actualizar la columna es generado por el gestor de bases de datos.

- Para obtener más información acerca de la utilización máxima de los valores de una secuencia generada para una columna de identidad y para obtener información acerca de cuándo se excede el valor máximo de una columna de identidad, consulte el apartado "INSERT".

## Ejemplos

- *Ejemplo 1:* Establezca la columna del salario de la fila para la que se ejecuta actualmente la acción del activador en 50000.

```
SET NEW_VAR.SALARY = 50000;
```

O bien:

```
SET (NEW_VAR.SALARY) = (50000);
```

- *Ejemplo 2:* Establezca las columnas del salario y la comisión de la fila para la que se está ejecutando actualmente la acción de activador en 50000 y 8000, respectivamente.

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
```

O bien:

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

## SET variable

- *Ejemplo 3:* Establezca la columna del salario y la comisión de la fila para la que se está ejecutando actualmente la acción del activador en el salario y la comisión promedio de los empleados del departamento que está asociado a la fila actualizada.

```
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 ejecuta actualmente la acción del activador en 10000 y el valor original del salario (es decir, antes de ejecutar la sentencia SET), respectivamente.

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
```

O bien:

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

- *Ejemplo 5:* Incremente la variable de SQL P\_SALARY en un 10 por ciento.

```
SET P_SALARY = P_SALARY + (P_SALARY * .10)
```

- *Ejemplo 6:* Establezca la variable de SQL P\_SALARY en el valor nulo.

```
SET P_SALARY = NULL
```

- *Ejemplo 7:* Asigne los números 2.71828183 y 3.1415926 al primer y décimo elemento de la variable de matriz SPECIALNUMBERS. Después de la primera asignación, la cardinalidad de P\_PHONENUMBERS es 1. Después de la segunda asignación, la cardinalidad es 10 y se ha asignado implícitamente a los elementos del 2 al 9 el valor nulo.

```
SET SPECIALNUMBERS[1] = 2.71828183;
```

```
SET SPECIALNUMBERS[10] = 3.14159265;
```

- *Ejemplo 8:* A partir de una tabla llamada SECURITY.USERS, que tiene una fila para cada usuario que puede conectarse a la base de datos, asigne la hora actual y el nivel de autorización a las variables globales USERINFO.GV\_CONNECT\_TIME y USERINFO.GV\_AUTH\_LEVEL, respectivamente.

```
SET USERINFO.GV_CONNECT_TIME = CURRENT_TIMESTAMP,
 USERINFO.GV_AUTH_LEVEL = (
 SELECT AUTHLEVEL FROM SECURITY.USERS
 WHERE USERID = CURRENT USER)
```

- *Ejemplo 9:* Asigne valores a la variable de matriz asociativa, CAPITALS, que se ha declarado como el tipo de matriz CAPITALSARRAY.

```
SET CAPITALS['British Columbia'] = 'Victoria';
SET CAPITALS['Alberta'] = 'Edmonton';
SET CAPITALS['Manitoba'] = 'Winnipeg';
SET CAPITALS['Canada'] = 'Ottawa';
```

Al rellenar los valores de la matriz CAPITALS, los índices de matriz son nombres de provincia, territorio y país especificados con series y los elementos de matriz asociados son capitales, que también se especifican con series.

- *Ejemplo 10:* Asigne nombres fáciles de recordar como índices para los números de teléfono personales almacenados en la variable de matriz PHONELIST con el tipo de matriz PERSONAL\_PHONENUMBERS.

```
SET PHONELIST['Casa'] = '4163053745';
SET PHONELIST['Trabajo'] = '4163053746';
SET PHONELIST['Madre'] = '4164789683';
```

## SIGNAL

La sentencia SIGNAL se utiliza para transmitir una condición de error o de aviso. Da lugar a que se devuelva un error o un aviso especificándose el SQLSTATE, junto con el texto de mensaje opcional.

### Invocación

Esta sentencia se puede incluir en:

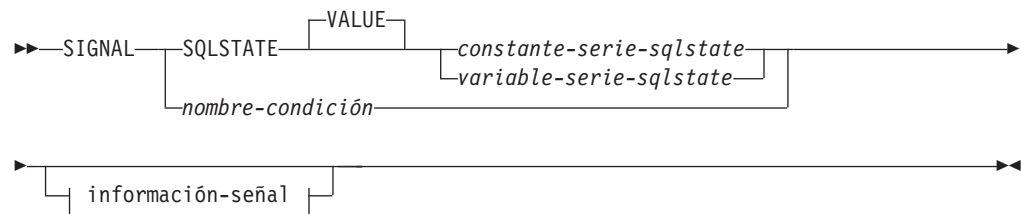
- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

Si se hace referencia a una condición de módulo, los privilegios del ID de autorización de la sentencia deben incluir el privilegio EXECUTE en el módulo.

### Sintaxis



#### información-senál:



### Descripción

#### SQLSTATE VALUE

Especifica el SQLSTATE que se devolverá. Puede utilizarse cualquier valor válido de SQLSTATE. El valor especificado debe respetar las reglas para los SQLSTATE:

- Cada carácter debe formar parte de un conjunto de dígitos (de '0' a '9') o de un conjunto de letras en mayúsculas (de 'A' a 'Z') no acentuadas.
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

En el contexto de una sentencia de SQL compuesto (en línea), se deben aplicar también las normas siguientes:

- La clase SQLSTATE (dos primeros caracteres) no puede ser '01' o '02', dado que éstos no son clases de error.

## SIGNAL

- 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 'T' a 'Z', la subclase puede ser cualquier carácter de '0' a '9' o de 'A' a 'Z'.

Si SQLSTATE no se ajusta a estas normas, se devuelve un error.

### *constante-serie-sqlstate*

El valor de *constante-serie-sqlstate* debe ser una constante de serie de caracteres que contenga exactamente 5 caracteres.

### *variable-serie-sqlstate*

La variable de SQL o parámetro de SQL que se especifica debe corresponder al tipo de datos CHAR(5) y no debe ser un valor nulo.

### *nombre-condición*

Especifica el nombre de una condición que se devolverá. El nombre-condición debe declararse en la sentencia-compuesta o identificar una condición que existe en el servidor actual (SQLSTATE 42373).

### **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.

### *expresión-serie-diagnóstico*

Una serie literal, o una variable o parámetro local, que describe la condición del error. Si la serie tiene más de 70 bytes, se truncará.

### *(expresión-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, se truncará. Esta opción sólo se proporciona en el ámbito de una sentencia CREATE TRIGGER para compatibilidad con versiones anteriores de DB2. Se recomienda no utilizarla de forma continua.

## Notas

- Si una sentencia SIGNAL se emite usando un *nombre-condición* que no tiene un valor de SQLSTATE asociado y la condición no se gestiona, se devuelve SQLSTATE 45000 y el SQLCODE se establece en -438. Tenga en cuenta que una condición de este tipo no se manejará por un manejador de condiciones para SQLSTATE 45000 que se encuentre en el ámbito de la rutina que emite la sentencia SIGNAL.
- Si una sentencia SIGNAL se emite utilizando un valor SQLSTATE o un *nombre-condición* con un valor SQLSTATE asociado, el valor SQLCODE devuelto se basa en el valor SQLSTATE de la siguiente manera:
  - Si la clase de SQLSTATE especificada es '01' ó '02', se devuelve un aviso o una condición de no encontrado y el SQLCODE se establece en +438.
  - De otro modo, se devuelve una condición de excepción y el SQLCODE se establece en -438.
- Una sentencia SIGNAL tiene los campos de la SQLCA establecidos de la forma siguiente:
  - Los campos sqlerrd se establecen en cero.
  - Los campos sqlwarn se establecen en blancos.
  - El campo sqlerrmc se establece en los 70 primeros bytes de MESSAGE\_TEXT.

- El campo `sqlerrml` se establece en la longitud de `sqlerrmc` o bien en cero si no se ha especificado ninguna cláusula `SET MESSAGE_TEXT`.
- El campo `sqlerrp` se establece en `ROUTINE`
- Los valores de `SQLSTATE` constan de un valor de código de clase de dos caracteres, seguido de un valor de código de subclase de 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'.

## Ejemplo

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 = 'No se conoce el número del cliente';
 INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
 VALUES (ONUM, CNUM, PNUM, QNUM);
 END

```

## TRANSFER OWNERSHIP

La sentencia TRANSFER OWNERSHIP transfiere la propiedad de un objeto de base de datos.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Propiedad del objeto
- Autorización SECADM

### Sintaxis

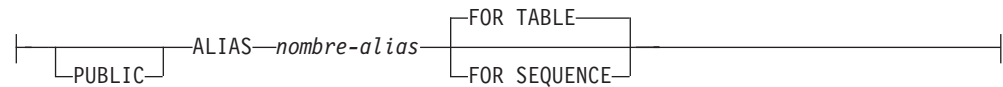
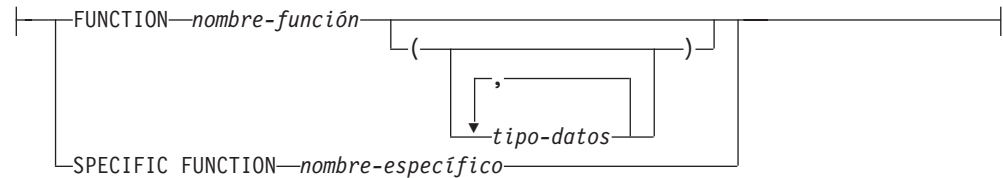
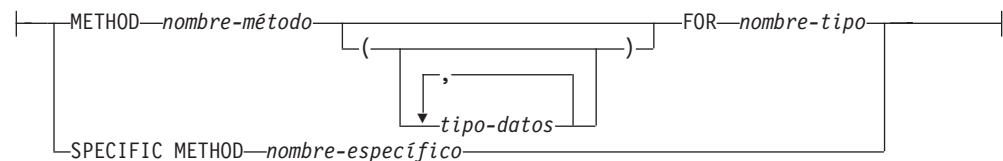
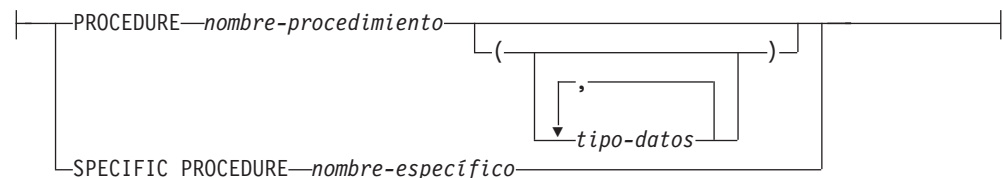
```

▶▶—TRANSFER OWNERSHIP OF—| objetos |—TO—| propietario-nuevo |—————▶
▶—PRESERVE PRIVILEGES—————▶▶

```

#### objetos:

|                          |                                 |                  |
|--------------------------|---------------------------------|------------------|
| designador-alias         |                                 |                  |
| CONSTRAINT               | nombre-tabla.nombre-restricción |                  |
| DATABASE PARTITION GROUP | nombre-grupo-particiones-bd     |                  |
| EVENT MONITOR            | nombre-supervisor-sucesos       |                  |
| designador-función       |                                 |                  |
| FUNCTION MAPPING         | nombre-correlación-función      |                  |
| INDEX                    | nombre-índice                   |                  |
| INDEX EXTENSION          | nombre-extensión-índice         |                  |
| designador-método        |                                 |                  |
| NICKNAME                 | apodo                           |                  |
| PACKAGE                  | nombre-esquema.                 | id-paquete       |
|                          |                                 | VERSION          |
|                          |                                 | id-versión       |
| designador-procedimiento |                                 |                  |
| SCHEMA                   | nombre-esquema                  |                  |
| SEQUENCE                 | nombre-secuencia                |                  |
| TABLE                    | nombre-tabla                    |                  |
| TABLE HIERARCHY          | nombre-tabla-raíz               |                  |
| TABLESPACE               | nombre-espacio-tablas           |                  |
| TRIGGER                  | nombre-activador                |                  |
|                          |                                 | TYPE nombre-tipo |
|                          |                                 | DISTINCT         |
| TYPE MAPPING             | nombre-correlación-tipos        |                  |
| VARIABLE                 | nombre-variable                 |                  |
| VIEW                     | nombre-vista                    |                  |
| VIEW HIERARCHY           | nombre-vista-raíz               |                  |
| XSROBJECT                | nombre-objetosr                 |                  |

**designador-alias:****designador-función:****designador-método:****designador-procedimiento:****propietario-nuevo:****Descripción***designador-alias***ALIAS** *nombre-alias*

Identifica el alias cuya propiedad se debe transferir. El *nombre-alias* debe designar un alias descrito en el catálogo (SQLSTATE 42704). Si se especifica PUBLIC, el *nombre-alias* debe identificar un alias público que existe en el servidor actual (SQLSTATE 42704).

**FOR TABLE o FOR SEQUENCE**

Especifica el tipo de objeto del alias.

**FOR TABLE**

El alias es para una tabla, vista o apodo. Cuando se transfiere la propiedad del alias, el valor de la columna OWNER del alias de la

## TRANSFER OWNERSHIP

vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.

### FOR SEQUENCE

El alias es para una secuencia. Cuando se transfiere la propiedad del alias, el valor de la columna OWNER del alias de la vista de catálogo SYSCAT.SEQUENCES se sustituye por el ID de autorización del propietario nuevo.

### CONSTRAINT *nombre-tabla.nombre-restricción*

Identifica la restricción cuya propiedad se debe transferir. La combinación *nombre-tabla.nombre-restricción* debe identificar una restricción y la tabla que restringe. El *nombre-restricción* debe identificar una restricción descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la restricción, el valor de la columna OWNER de la restricción de la vista de catálogo SYSCAT.TABCONST se sustituye por el ID de autorización del propietario nuevo.

- Si la restricción es una restricción FOREIGN KEY, la columna OWNER de la vista de catálogo SYSCAT.REFERENCES se sustituye por el ID de autorización del propietario nuevo.
- Si la restricción es una restricción PRIMARY KEY o UNIQUE, la columna OWNER de la vista de catálogo SYSCAT.INDEXES para el índice creado implícitamente para esta restricción se sustituye por el ID de autorización del propietario nuevo. Si el índice ya existía y se reutiliza en este caso, el propietario del índice no se cambia.

### DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*

Identifica el grupo de particiones de base de datos cuya propiedad se debe transferir. El *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del grupo de particiones de base de datos, el valor de la columna OWNER del grupo de particiones de base de datos de la vista de catálogo SYSCAT.DBPARTITIONGROUPS se sustituye por el ID de autorización del propietario nuevo.

### EVENT MONITOR *nombre-supervisor-sucesos*

Identifica el supervisor de sucesos cuya propiedad se debe transferir. El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que se haya descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del supervisor de sucesos, el valor de la columna OWNER del supervisor de sucesos de la vista de catálogo SYSCAT.EVENTMONITORS se sustituye por el ID de autorización del propietario nuevo.

Si el supervisor de sucesos identificado está activo, se devuelve un error (SQLSTATE 429BT).

Si hay archivos de sucesos en la vía de acceso de destino de un supervisor de sucesos WRITE TO FILE cuya propiedad se transfiere, los archivos de sucesos no se suprimirán.

Cuando se transfiere la propiedad de supervisores de sucesos WRITE TO TABLE, se conserva la información de tabla de la vista de catálogo SYSCAT.EVENTTABLES.

### *designador-función*

Identifica la función cuya propiedad se debe transferir. Para obtener más información, consulte el apartado "Designadores de función, método y



procedimiento” en la página 22. La instancia de función especificada debe ser una función definida por el usuario o una plantilla de función descrita en el catálogo. La propiedad de las funciones que las sentencias CREATE TYPE generan implícitamente por no puede transferirse (SQLSTATE 429BT).

Cuando se transfiere la propiedad de la función, el valor de la columna OWNER de la función de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo. La transferencia de la propiedad de una función de SQL que tiene asociado un paquete también transfiere implícitamente la propiedad del paquete al nuevo propietario.

#### **SPECIFIC FUNCTION** *nombre-específico*

Identifica la función definida por el usuario determinada cuya propiedad se debe transferir, utilizando el nombre concreto que se ha especificado o tomado por omisión al crear la función. 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 o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema especificado o implícito; de lo contrario, se devuelve un error (SQLSTATE 42704).

Cuando se transfiere la propiedad de la función específica, el valor de la columna OWNER de la función específica de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

#### **FUNCTION MAPPING** *nombre-correlación-funciones*

Identifica la correlación de funciones cuya propiedad se debe transferir. El *nombre-correlación-funciones* debe identificar una correlación de funciones que está descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la correlación de funciones, el valor de la columna OWNER de la correlación de funciones de la vista de catálogo SYSCAT.FUNCMAPPINGS se sustituye por el ID de autorización del propietario nuevo.

#### **INDEX** *nombre-índice*

Identifica el índice o la especificación de índice cuya propiedad se debe transferir. El *nombre-índice* debe identificar un índice o especificación de índice que se describa en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del índice, el valor de la columna OWNER del índice de la vista de catálogo SYSCAT.INDEXES se sustituye por el ID de autorización del propietario nuevo.

La propiedad de un índice no se puede transferir si la tabla en que está definido el índice es una tabla temporal global (SQLSTATE 429BT).

#### **INDEX EXTENSION** *nombre-extensión-índice*

Identifica la extensión de índice cuya propiedad se debe transferir. El *nombre-extensión-índice* debe identificar una extensión de índice que esté descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la extensión de índice, el valor de la columna OWNER de la extensión de índice de la vista de catálogo SYSCAT.INDEXEXTENSIONS se sustituye por el ID de autorización del propietario nuevo.

## TRANSFER OWNERSHIP

### *designador-método*

Identifica el método cuya propiedad se debe transferir. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22. El cuerpo del método especificado debe ser un método descrito en el catálogo (SQLSTATE 42704). La propiedad de los métodos generados implícitamente por la sentencia CREATE TYPE no se puede transferir (SQLSTATE 429BT).

Cuando se transfiere la propiedad del método, el valor de la columna OWNER del método de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

### **NICKNAME** *apodo*

Identifica el apodo cuya propiedad se debe transferir. El *apodo* debe ser un apodo descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del apodo, el valor de la columna OWNER del apodo de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.

### **PACKAGE** *nombre-esquema.id-paquete*

Identifica el paquete cuya propiedad se debe transferir. Si no se especifica un nombre de esquema, el esquema por omisión califica implícitamente el identificador del paquete. El nombre de esquema y el identificador de paquete, junto con el identificador de versión especificado implícita o explícitamente, deben identificar un paquete que esté descrito en el catálogo (SQLSTATE 42704).

### **VERSION** *id-versión*

Identifica de qué versión del paquete se debe transferir la propiedad. Si no se especifica ningún valor, la versión se establece por omisión en la serie de caracteres vacía y se transfiere la propiedad de este paquete. Si existen varios paquetes con el mismo nombre de paquete pero con distintas versiones, sólo se transfiere la propiedad del paquete cuyo *id-versión* se ha especificado en la sentencia TRANSFER OWNERSHIP. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

Si la sentencia se invoca desde un indicador de mandatos del sistema operativo, preceda cada delimitador de comillas dobles con una barra inclinada invertida para asegurarse de que el sistema operativo no divide los delimitadores.

Cuando se transfiere la propiedad del paquete, el valor de la columna BOUNDBY del paquete de la vista de catálogo SYSCAT.PACKAGES se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de los paquetes asociados a procedimientos de SQL (SQLSTATE 429BT).

### *designador-procedimiento*

Identifica el procedimiento cuya propiedad se debe transferir. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 22. La instancia de procedimiento especificada debe ser un procedimiento descrito en el catálogo.

Cuando se transfiere la propiedad del procedimiento, el valor de la columna OWNER del procedimiento de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

Al transferir la propiedad de un procedimiento de SQL que tiene un paquete asociado, también se transfiere implícitamente la propiedad del paquete al propietario nuevo.

**SPECIFIC PROCEDURE** *nombre-específico*

Identifica el procedimiento determinado cuya propiedad se debe transferir, utilizando el nombre concreto que se ha especificado o tomado por omisión al crear el procedimiento. 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 o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de procedimiento específica en el esquema especificado o implícito; de lo contrario, se devuelve un error (SQLSTATE 42704).

Cuando se transfiere la propiedad del procedimiento específico, el valor de la columna OWNER del procedimiento específico de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

**SCHEMA** *nombre-esquema*

Identifica el esquema cuya propiedad se debe transferir. El *nombre-esquema* debe identificar un esquema que se describe en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del esquema, el valor de la columna OWNER y la columna DEFINER para el esquema de la vista de catálogo SYSCAT.SCHEMATA se sustituyen por el ID de autorización del propietario nuevo.

La propiedad de esquemas incorporados (donde la persona que realiza la definición es SYSIBM) no se puede transferir (SQLSTATE 42832).

**SEQUENCE** *nombre-secuencia*

Identifica la secuencia cuya propiedad se debe transferir. El *nombre-secuencia* debe identificar una secuencia descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la secuencia, el valor de la columna OWNER del esquema de la vista de catálogo SYSCAT.SEQUENCES se sustituye por el ID de autorización del propietario nuevo.

**TABLE** *nombre-tabla*

Identifica la tabla cuya propiedad se debe transferir. El *nombre-tabla* debe identificar una tabla que exista en la base de datos (SQLSTATE 42704) y no puede identificar una tabla temporal declarada (SQLSTATE 42995).

Cuando se transfiere la propiedad de la tabla:

- El valor de la columna OWNER de la tabla de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la tabla de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de las subtablas de una jerarquía de tablas (SQLSTATE 429BT).

## TRANSFER OWNERSHIP

En un sistema federado, se puede transferir la propiedad de una tabla remota que se ha creado utilizando un DDL transparente. Al transferir la propiedad de una tabla remota, no se transfiere la propiedad del apodo asociado a la tabla. La propiedad de un apodo de este tipo se puede transferir explícitamente con la sentencia TRANSFER OWNERSHIP.

### **TABLE HIERARCHY** *nombre-tabla-raíz*

Identifica la tabla con tipo que es la tabla raíz de una jerarquía de tablas con tipo cuya propiedad se debe transferir. El *nombre-tabla-raíz* debe identificar una tabla con tipo que es la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR) y debe hacer referencia a una tabla con tipo existente en la base de datos (SQLSTATE 42704).

Cuando se transfiere la propiedad de la jerarquía de tablas:

- El valor de la columna OWNER de la tabla raíz y todas sus subtablas de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la tabla y todas sus subtablas de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

### **TABLESPACE** *nombre-espacio-tablas*

Identifica el espacio de tablas cuya propiedad se debe transferir. El *nombre-espacio-tablas* debe identificar un espacio de tablas descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del espacio de tablas, el valor de la columna OWNER del espacio de tablas de la vista de catálogo SYSCAT.TABLESPACES se sustituye por el ID de autorización del propietario nuevo.

### **TRIGGER** *nombre-activador*

Identifica el activador cuya propiedad se debe transferir. El *nombre-activadores* debe identificar un activador que se haya descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del activador, el valor de la columna OWNER del activador de la vista de catálogo SYSCAT.TRIGGERS se sustituye por el ID de autorización del propietario nuevo.

### **TYPE** *nombre-tipo*

Identifica el tipo definido por el usuario cuya propiedad se debe transferir. El *nombre-tipo* debe identificar un tipo 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).

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 o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados.

Cuando se transfiere la propiedad del tipo, el valor de la columna OWNER del tipo de la vista de catálogo SYSCAT.DATATYPES se sustituye por el ID de autorización del propietario nuevo.

### **TYPE MAPPING** *nombre-correlación-tipos*

Identifica la correlación de tipos de datos definida por el usuario cuya propiedad se debe transferir. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la correlación de tipos, el valor de la columna OWNER de la correlación de tipos de la vista de catálogo SYSCAT.TYPEMAPPINGS se sustituye por el ID de autorización del propietario nuevo.

**VARIABLE** *nombre-variable*

Indica que el objeto cuya propiedad se va a transferir es una variable global creada. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

Cuando se transfiere la variable global, el valor de la columna OWNER de la variable global de la vista de catálogo SYSCAT.VARIABLES se sustituye por el ID de autorización del propietario nuevo.

**VIEW** *nombre-vista*

Identifica la vista cuya propiedad se debe transferir. El *nombre-vista* debe identificar una vista existente en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la vista:

- El valor de la columna OWNER de la vista de la vista de catálogo SYSCAT.VIEWS se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la vista de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de una subvista de una jerarquía de vistas (SQLSTATE 429BT).

**VIEW HIERARCHY** *nombre-vista-raíz*

Identifica la vista con tipo que es la vista raíz de una jerarquía de vistas con tipo cuya propiedad se debe transferir. El *nombre-vista-raíz* debe identificar una vista con tipo que es la vista raíz de la jerarquía de vistas con tipo (SQLSTATE 428DR) y debe hacer referencia a una vista con tipo existente en la base de datos (SQLSTATE 42704).

Cuando se transfiere la propiedad de la jerarquía de vistas:

- El valor de la columna OWNER de la vista raíz y todas sus subvistas de la vista de catálogo SYSCAT.VIEWS se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la vista y todas sus subvistas de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

**XSROBJECT** *nombre-objetoXSR*

Identifica el objeto XSR cuya propiedad se debe transferir. El *nombre-objetoXSR* debe identificar un objeto XSR que se describe en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del objeto XSR, el valor de la columna OWNER del objeto XSR de la vista de catálogo SYSCAT.XSROBJECTS se sustituye por el ID de autorización del propietario nuevo.

**USER** *nombre-autorización*

Especifica el ID de autorización al cual se transfiere la propiedad del objeto.

**SESSION\_USER**

Especifica que el valor del registro especial SESSION\_USER se debe utilizar como el ID de autorización al cual se transfiere la propiedad del objeto.

## TRANSFER OWNERSHIP

### SYSTEM\_USER

Especifica que el valor del registro especial SYSTEM\_USER se debe utilizar como el ID de autorización al cual se transfiere la propiedad del objeto.

### PRESERVE PRIVILEGES

Especifica que el propietario actual de un objeto cuya propiedad se va a transferir seguirá teniendo los privilegios actuales para el objeto tras la transferencia. Por ejemplo, los privilegios que se otorgaron al creador de una vista cuando se creó esa vista seguirán en poder del propietario original incluso después de que se transfiera la propiedad a otro usuario.

## Normas

- No se puede transferir la propiedad de la mayoría de los objetos incorporados (en que el propietario es SYSIBM) (SQLSTATE 42832). No obstante, puede transferir la propiedad de objetos de esquema creados implícitamente que tienen SYSIBM en la columna OWNER y no tienen SYSIBM en la columna DEFINER.
- No se puede transferir la propiedad de esquemas cuyo nombre empieza por 'SYS' (SQLSTATE 42832).
- No se puede transferir explícitamente la propiedad de los objetos siguientes (SQLSTATE 429BT):
  - Subtablas de una jerarquía de tablas (se transfieren con la tabla de jerarquía raíz)
  - Subvistas de una jerarquía de vistas (se transfieren con la vista de jerarquía raíz)
  - Índices que están definidos en tablas temporales globales
  - Métodos o funciones que se generan implícitamente cuando se crea un tipo definido por el usuario
  - Alias de módulo y módulos
  - Paquetes que dependen de procedimientos de SQL (se transfieren con el procedimiento de SQL)
  - Supervisores de sucesos que están activos (se pueden transferir cuando no están activos)
- Un ID de autorización con la autorización SECADM no puede transferirse la propiedad de un objeto si no es ya el propietario del objeto (SQLSTATE 42502).

## Notas

- Todos los privilegios del propietario actual que se otorgaron como parte de la creación del objeto se transfieren al nuevo propietario. Si al propietario actual se le ha revocado un privilegio sobre el objeto y posteriormente se le ha vuelto a otorgar dicho privilegio, el privilegio no se transferirá. En el caso de los objetos de esquema creados implícitamente que todavía no se han transferido, se otorga al nuevo propietario privilegios CREATEIN, DROPIN y ALTERIN sobre el esquema con la posibilidad de otorgar estos privilegios a otros usuarios.
- Al transferir la propiedad de un objeto de base de datos, el nuevo propietario debe poseer el conjunto de privilegios sobre los objetos básicos, tal como indican las dependencias del objeto que son necesarias para mantener la existencia del objeto sin modificaciones. El nuevo propietario no necesita los privilegios para crear el objeto si dichos privilegios no son necesarios para mantener la existencia del objeto.

Por ejemplo:

- Imagine una vista con dependencias SELECT e INSERT sobre una tabla subyacente. Entre los privilegios que posee el propietario nuevo de la vista deben encontrarse como mínimo SELECT (con o sin GRANT OPTION) e

INSERT (con o sin GRANT OPTION) para que se pueda transferir la propiedad correctamente. Si las dependencias son SELECT WITH GRANT OPTION e INSERT WITH GRANT OPTION, entre los privilegios que posee el propietario nuevo de la vista deben encontrarse como mínimo SELECT WITH GRANT OPTION e INSERT WITH GRANT OPTION.

- Imagine una vista con una dependencia sobre una rutina. Entre los privilegios que posee el propietario nuevo de la vista debe encontrarse como mínimo EXECUTE para la rutina dependiente.
- Imagine un activador con una dependencia sobre una tabla. Entre los privilegios que posee el propietario nuevo del activador debe encontrarse el mismo conjunto de privilegios para la tabla que indican las dependencias del activador. No es necesario el privilegio ALTER para la tabla en que se define el activador.

En la tabla siguiente se indican las vistas de catálogo que describen los objetos de que dependen otros objetos de base de datos.

*Tabla 36. Vistas de catálogo que describen los objetos de que dependen otros objetos*

| Objeto de base de datos | Vista de catálogo                                                 |
|-------------------------|-------------------------------------------------------------------|
| CONSTRAINT              | SYSCAT.CONSTDEP                                                   |
| FUNCTION                | SYSCAT.ROUTINEDEP; SYSCAT.ROUTINES<br>(para una función derivada) |
| INDEX                   | SYSCAT.INDEXDEP                                                   |
| INDEX EXTENSION         | SYSCAT.INDEXEXTENSIONDEP                                          |
| METHOD                  | SYSCAT.ROUTINEDEP                                                 |
| PACKAGE                 | SYSCAT.PACKAGEDEP                                                 |
| PROCEDURE               | SYSCAT.ROUTINEDEP                                                 |
| TABLE                   | SYSCAT.TABDEP                                                     |
| TRIGGER                 | SYSCAT.TRIGDEP                                                    |
| VIEW                    | SYSCAT.TABDEP                                                     |
| XSROBJECT               | SYSCAT.XSROBJECTDEP                                               |

Para transferir correctamente la propiedad de un objeto de base de datos que depende de otro objeto, el nuevo propietario del objeto de base de datos debe poseer unos privilegios determinados para el objeto dependiente de dicha dependencia:

- Si el objeto dependiente es una secuencia, el nuevo propietario debe poseer el privilegio USAGE para dicha secuencia.
- Si el objeto dependiente es una función, un método o un procedimiento, el nuevo propietario debe poseer el privilegio EXECUTE para dicha función, método o procedimiento.
- Si el objeto dependiente es un paquete, el nuevo propietario debe poseer el privilegio EXECUTE para dicho paquete.
- Si el objeto dependiente es un objeto XSR, el nuevo propietario debe poseer el privilegio USAGE para dicho objeto XSR.

Para cualquier otro objeto dependiente de una dependencia, utilice la columna TABAUTH de la vista de catálogo adecuada para determinar qué privilegios debe poseer el nuevo propietario.

- Si se intenta transferir la propiedad de un objeto a su propietario, se devuelve un aviso (SQLSTATE 01676).

## TRANSFER OWNERSHIP

- No se puede transferir la propiedad de los siguientes objetos de base de datos porque dichos objetos no tienen propietario: políticas de auditoría, agrupaciones de almacenamientos intermedios, roles, etiquetas de seguridad, componentes de etiquetas de seguridad, políticas de seguridad, servidores, funciones de transformación, contextos fiables, correlaciones de usuarios y derivadores. Tenga en cuenta que no existe ninguna columna OWNER en las vistas de catálogo SYSCAT.AUDITPOLICIES, SYSCAT.BUFFERPOOLS, SYSCAT.CONTEXTS, SYSCAT.ROLES, SYSCAT.SECURITYLABELS, SYSCAT.SECURITYLABELCOMPONENTS, SYSCAT.SECURITYPOLICIES, SYSCAT.SERVERS, SYSCAT.TRANSFORMS, SYSCAT.USEROPTIONS y SYSCAT.WRAPPERS.
- El nombre de esquema de un objeto cuya propiedad se ha transferido no cambia automáticamente.
- *Alternativas de sintaxis:* para mantener la coherencia con otras sentencias de SQL:
  - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
  - SYNONYM puede especificarse en lugar de ALIAS

### Ejemplos

- *Ejemplo 1:* Transfiere la propiedad de la tabla T1 a PAUL.

```
TRANSFER OWNERSHIP OF TABLE WALID.T1
TO USER PAUL PRESERVE PRIVILEGES
```

El valor de la columna OWNER de la tabla WALID.T1 de la vista de catálogo SYSCAT.TABLES se sustituye por 'PAUL'. Se otorgan a Paul implícitamente los privilegios siguientes para la tabla WALID.T1 (suponiendo que el propietario anterior de la tabla no perdió ninguno de los privilegios que tenía para ella): CONTROL y ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE, REFERENCE (WITH GRANT OPTION).

- *Ejemplo 2:* Suponga que JOHN crea las tablas T1 y T2, y que MIKE tiene privilegio SELECT para las tablas JOHN.T1 y JOHN.T2. MIKE crea la vista V1 que depende de las tablas JOHN.T1 y JOHN.T2. Transfiere la propiedad de la vista V1 a HENRY, que tiene la autorización DBADM.

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER HENRY PRESERVE PRIVILEGES
```

El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'HENRY'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'HENRY' y TABNAME = 'V1'.

- *Ejemplo 3:* Suponga que HENRY, que tiene la autorización DBADM, crea un activador TR1 que depende de la tabla T1. Transfiere la propiedad del activador TR1 a WALID, que no tiene la autorización DBADM.

```
TRANSFER OWNERSHIP OF TRIGGER TR1
TO USER WALID PRESERVE PRIVILEGES
```

La propiedad del activador se transfiere correctamente, aunque Walid no tiene la autorización DBADM.

- *Ejemplo 4:* Suponga que JOHN crea las tablas T1 y T2, y que MIKE tiene privilegio SELECT para la tabla JOHN.T1 y privilegio CONTROL para la tabla JOHN.T2. PAUL tiene privilegio SELECT para las tablas JOHN.T1 y JOHN.T2. MIKE crea la vista V1 que depende de las tablas JOHN.T1 y JOHN.T2. La vista tiene una entrada para el privilegio SELECT en SYSCAT.TABAUTH y dos



dependencias SELECT en SYSCAT.TABDEP para las tablas JOHN.T1 y JOHN.T2. Transfiera la propiedad de la vista V1 a PAUL, que es un usuario normal.

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER PAUL PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, aunque Paul no tiene el privilegio CONTROL para la tabla JOHN.T2. Paul solo necesita el privilegio SELECT para las tablas JOHN.T1 y JOHN.T2 con objeto de mantener la existencia de la vista. (La vista solo tiene el privilegio SELECT ya que Paul no tenía el privilegio CONTROL para ambas tablas al crearse la vista, por lo que no se le otorgó el privilegio CONTROL para la vista.) El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'PAUL'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'PAUL'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'PAUL' y TABNAME = 'V1'.

- *Ejemplo 5:* Suponga que JOHN crea la tabla T1 y que PUBLIC tiene privilegio SELECT para JOHN.T1. PAUL tiene privilegio SELECT para JOHN.T1 explícitamente y crea la vista V1 que depende de la tabla JOHN.T1. Transfiera la propiedad de la vista V1 a MIKE, que no es DBADM pero tiene los privilegios necesarios para adquirir una propiedad de vista mediante el grupo especial PUBLIC.

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER MIKE PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, ya que Mike tiene el privilegio SELECT para la tabla JOHN.T1 por medio de PUBLIC. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'MIKE'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'MIKE'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' y TABNAME = 'V1'.

- *Ejemplo 6:* al igual que en el ejemplo 5, suponga que JOHN crea la tabla T1 y que el rol R1 tiene el privilegio SELECT para JOHN.T1. PAUL tiene privilegio SELECT para JOHN.T1 explícitamente y crea la vista V1 que depende de la tabla JOHN.T1. Transfiera la propiedad de la vista V1 a MIKE, que no es un usuario DBADM pero tiene los privilegios necesarios por ser miembro en el rol R1 para adquirir la propiedad de vista.

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER MIKE PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, ya que Mike tiene el privilegio SELECT para la tabla JOHN.T1 por ser miembro en el rol R1. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'MIKE'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'MIKE'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' y TABNAME = 'V1'.

## TRUNCATE

La sentencia TRUNCATE suprime todas las filas de una tabla.

### Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

### Autorización

Los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes para la tabla y todas las subtablas de una jerarquía de tablas:

- Privilegio DELETE sobre la tabla que se debe truncar
- Privilegio CONTROL sobre la tabla que se debe truncar
- Autorización DATAACCESS

Para pasar por alto los activadores DELETE definidos en la tabla, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes para la tabla y todas las subtablas de una jerarquía de tablas:

- Privilegio ALTER sobre la tabla
- Privilegio CONTROL sobre la tabla
- Autorización DBADM

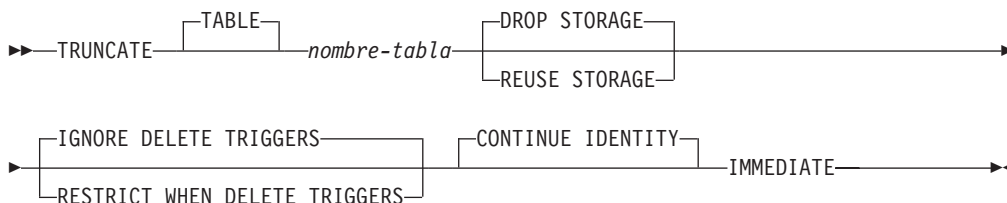
Para truncar una tabla protegida por una política de seguridad, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla
- Autorización DBADM

Para truncar una tabla que tiene activado el control de acceso a filas, el ID de autorización de la sentencia debe incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio CONTROL sobre la tabla
- Autorización DBADM

### Sintaxis



## Descripción

### *nombre-tabla*

Identifica la tabla que se debe truncar. El nombre debe identificar una tabla que exista en el servidor actual (SQLSTATE 42704), pero no puede ser una tabla de catálogo (SQLSTATE 42832), un apodo (SQLSTATE 42809), una vista, una subtabla, una tabla de etapas, una tabla de consulta materializada mantenida por el sistema, una tabla temporal de período de sistema (SQLSTATE 428HZ) ni una tabla agrupada por clústeres de rangos (SQLSTATE 42807).

Si *nombre-tabla* es la tabla raíz de una jerarquía de tablas, se truncarán todas las tablas de la jerarquía.

### **DROP STORAGE o REUSE STORAGE**

Especifica si se debe descartar o reutilizar el almacenamiento existente asignado para la tabla. El valor por omisión es DROP STORAGE.

#### **DROP STORAGE**

Todo el almacenamiento asignado a la tabla queda libre y disponible. Si se especifica esta opción (de forma implícita o explícita), se bloquearía una copia de seguridad en línea.

#### **REUSE STORAGE**

Todo el almacenamiento asignado a la tabla seguirá estando asignado a ella, pero se considerará vacío. Esta opción sólo es aplicable a las tablas que se encuentran en espacios de tablas DMS; en caso contrario se pasa por alto.

### **IGNORE DELETE TRIGGERS o RESTRICT WHEN DELETE TRIGGERS**

Especifica lo que debe llevarse a cabo cuando se definen activadores de supresión en la tabla. El valor por omisión es IGNORE DELETE TRIGGERS.

#### **IGNORE DELETE TRIGGERS**

La operación de truncamiento no activa los activadores de supresión que haya definidos para la tabla.

#### **RESTRICT WHEN DELETE TRIGGERS**

Se devuelve un error si hay activadores de supresión definidos en la tabla (SQLSTATE 428GJ).

### **CONTINUE IDENTITY**

Si existe una columna de identidad para la tabla, el siguiente valor de columna de identidad generado continúa con el valor siguiente que se habría generado si la sentencia TRUNCATE no se hubiera ejecutado.

### **IMMEDIATE**

Especifica que la operación de truncamiento se procesa inmediatamente y no se podrá deshacer. La sentencia debe ser la primera en una transacción (SQLSTATE 25001).

La tabla truncada está disponible inmediatamente para ser utilizada en la misma unidad de trabajo. Aunque está permitido que una sentencia ROLLBACK se ejecute después de una sentencia TRUNCATE, la operación de truncamiento no se puede deshacer y la tabla permanecerá truncada. Por ejemplo, si se realiza otra operación de cambio de datos en la tabla después de la sentencia TRUNCATE IMMEDIATE y a continuación se ejecuta la sentencia ROLLBACK, la operación de truncamiento no se deshazá, pero sí se deshazán las demás operaciones de cambio de datos.

# TRUNCATE

## Normas

- **Integridad referencial:** la tabla, así como todas las tablas de una jerarquía de tablas, no deben ser una tabla padre en una restricción de referencia impuesta (SQLSTATE 428GJ). Se permite el uso de una restricción de RI con referencia a sí misma.
- **Tablas particionadas:** la tabla no debe tener el estado pendiente de establecer integridad debido a que se ha alterado para añadirla a una partición de datos (SQLSTATE 55019). Es necesario comprobar la integridad de la tabla antes de ejecutar la sentencia TRUNCATE. En DB2 Versión 9.7 Fixpack 1 y releases posteriores, la tabla no debe tener ninguna partición desenlazada lógicamente (SQLSTATE 55057). La tarea de desconexión de partición asíncrona deberá haberse completado antes de ejecutar la sentencia TRUNCATE.
- **Acceso exclusivo:** ninguna otra sesión puede tener un cursor abierto ni un bloqueo en la tabla (SQLSTATE 25001).
- **Cursores WITH HOLD:** la sesión actual no puede tener un cursor WITH HOLD abierto en la tabla (SQLSTATE 25001).

## Notas

- **Estadísticas de tabla:** la sentencia TRUNCATE no modifica las estadísticas de la tabla.
- **Número de filas suprimidas:** SQLERRD(3) en la SQLCA tiene el valor -1 para la operación de truncamiento. El número de filas que se han eliminado de la tabla no se devuelve.

## Ejemplos

- *Ejemplo 1:* vaciar una tabla de inventario no utilizada independientemente de si existen activadores y devolver el espacio que tenía asignado.

```
TRUNCATE TABLE INVENTORY
IGNORE DELETE TRIGGERS
DROP STORAGE
IMMEDIATE
```

- *Ejemplo 2:* vaciar una tabla de inventario no utilizada independientemente de si existen activadores de supresión y conservar el espacio que tiene asignado para utilizarlo posteriormente.

```
TRUNCATE TABLE INVENTORY
REUSE STORAGE
IGNORE DELETE TRIGGERS
IMMEDIATE
```

---

## UPDATE

La sentencia UPDATE actualiza los valores de las columnas especificadas en las filas de una tabla, vista o apodo, o las tablas, apodos o vistas subyacentes de la selección completa especificada.

La actualización de una fila de una vista actualiza una fila de su tabla base, si no se ha definido ningún activador INSTEAD OF para la operación de actualización en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador. La actualización de una fila utilizando un apodo actualiza una fila del objeto de fuente de datos al que hace referencia el apodo.

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 incorporarse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

### Autorización

Los privilegios que posee el ID de autorización de la sentencia debe tener al menos una de las autorizaciones siguientes:

- Privilegio UPDATE para la tabla, vista o apodo de destino
- Privilegio UPDATE para cada una de las columnas que deben actualizarse, incluidas las columnas del periodo BUSINESS\_TIME si se ha especificado una cláusula de periodo.
- Privilegio CONTROL sobre la tabla, vista o apodo de destino
- Autorización DATAACCESS

Si se incluye una *selección-completa-fila* en la asignación, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, una de las autorizaciones siguientes para cada tabla, vista o apodo al que se hace referencia:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

Para cada tabla, vista o apodo al que hace referencia una subconsulta, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

Si el paquete utilizado para procesar la sentencia se precompila con normas SQL92 (opción LANGLEVEL con un valor de SQL92E o MIA) y la forma buscada de una sentencia UPDATE incluye una referencia a una columna de la tabla, vista o apodo en la parte derecha de la *cláusula-asignación*, o en cualquier parte de la

## UPDATE

*condición-búsqueda*, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las autorizaciones siguientes:

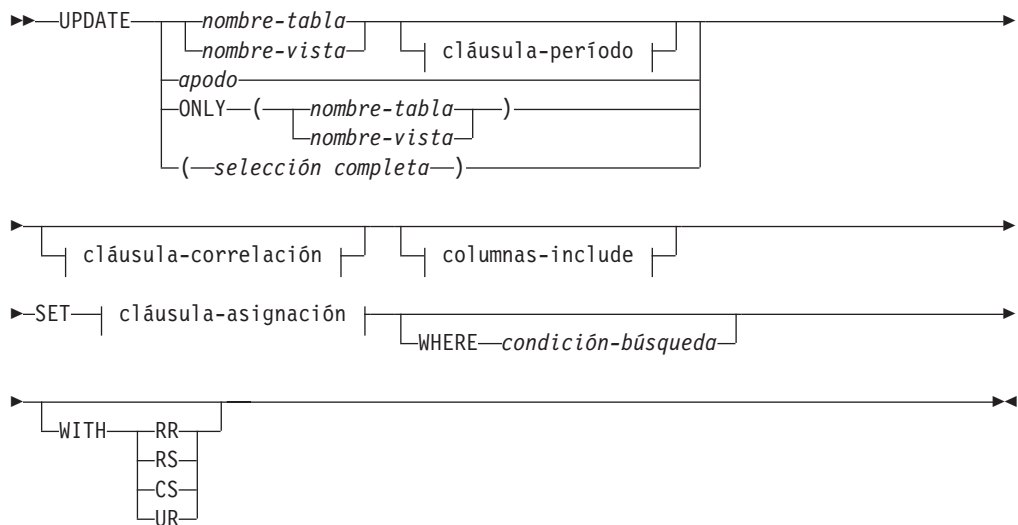
- Privilegio SELECT
- Privilegio CONTROL
- Autorización DATAACCESS

Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

No se comprueban los privilegios GROUP para las sentencias UPDATE estáticas.

Si el destino de la operación de actualización es un apodo, los privilegios para el objeto en la fuente de datos no se toman en consideración hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización utilizado para conectarse a la fuente de datos debe disponer de los privilegios necesarios a fin de realizar la operación con el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

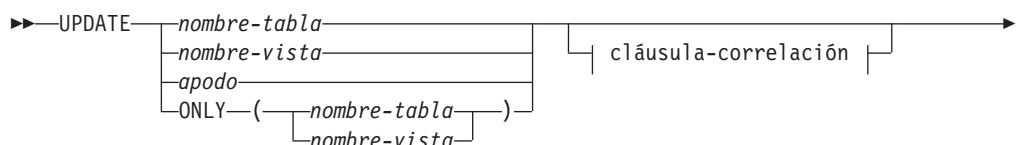
### Sintaxis (update-con-búsqueda)



### cláusula-período:

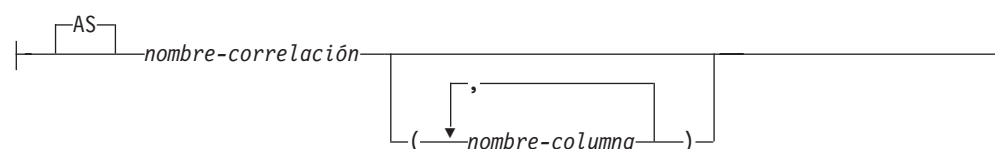
FOR PORTION OF BUSINESS\_TIME FROM valor1 TO valor2

### Sintaxis (update-con-posición)

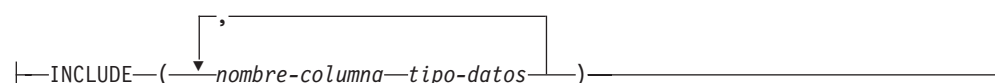


► SET | cláusula-asignación | WHERE CURRENT OF *nombre-cursor* →

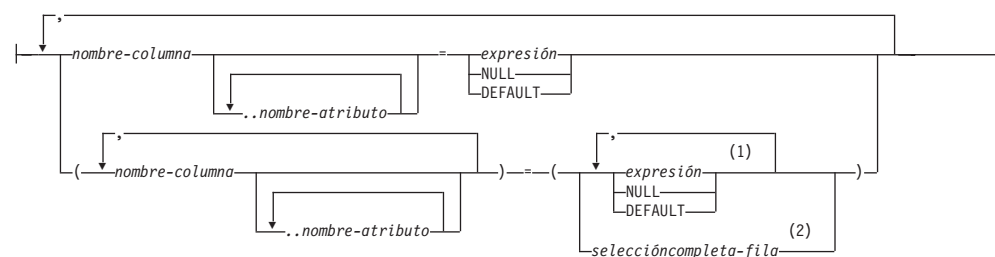
### cláusula-correlación:



### columnas-include:



### cláusula-asignación:



### Notas:

- 1 El número de expresiones, de NULL y de DEFAULT debe coincidir con el número de nombres de columna.
- 2 El número de columnas en la lista de selección debe coincidir con el número de nombres de columna.

## Descripción

*nombre-tabla*, *nombre-vista*, *apodo* o (*selección completa*)

Identifica el objeto de la operación de actualización. El nombre debe identificar uno de los objetos siguientes:

- Una tabla, vista o apodo que se describe en el catálogo en el servidor actual
- Una tabla o vista en el servidor remoto que se especifica mediante un nombre-objeto-remoto

El objeto no debe ser una tabla de catálogo, una vista de una tabla de catálogo (a menos que sea una de las vistas SYSSTAT actualizables), una tabla de consulta materializada mantenida por el sistema o una vista de sólo lectura que no tenga definido ningún activador INSTEAD OF para sus operaciones de actualización.

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 especificar la misma tabla, vista o apodo en la cláusula FROM sin utilizar ONLY.

Si el objeto de la operación de actualización es una selección completa, esta debe ser actualizable, según lo definido en el elemento “Vistas actualizables” de Notas de la descripción de la sentencia CREATE VIEW.

Si el objeto de la operación de actualización es un apodo, los valores de variable de indicador ampliado DEFAULT y UNASSIGNED no deben utilizarse (SQLSTATE 22539).

Para obtener información acerca de las restricciones adicionales relacionadas con las tablas temporales y el uso de una vista o *selección completa* como destino de la operación de actualización, consulte los apartados “Consideraciones para una tabla temporal de período de sistema:” y “Consideraciones para una tabla temporal de período de aplicación” en la sección Notas de este tema.

### **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 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.

### **cláusula-período**

Especifica que una cláusula de período se aplica al destino de la operación de actualización. Si el destino de la operación de actualización es una vista, a la vista se aplican las condiciones siguientes:

- La cláusula FROM de la selección completa externa de la definición de vista debe incluir una referencia, directa o indirectamente, a una tabla temporal de período de aplicación (SQLSTATE 42724M).
- No debe definirse un activador INSTEAD OF UPDATE para la vista (SQLSTATE 428HY).

### **FOR PORTION OF BUSINESS\_TIME**

Especifica que la actualización sólo se aplica a los valores de fila de la parte correspondiente al período en la fila que especifique la cláusula de período. El período BUSINESS\_TIME debe existir en la tabla (SQLSTATE 4274M).

### **FROM** *valor1* **TO** *valor2*

Especifica que la actualización se aplica a filas durante el período especificado desde *valor1* hasta *valor2*. No se actualiza ninguna fila si *valor1* es superior o igual a *valor2* o si *valor1* o *valor2* es el valor nulo (SQLSTATE 02000).

Para el período especificado con FROM *valor1* TO *valor2*, el período BUSINESS\_TIME en una fila del destino de la actualización tiene uno de los estados siguientes:



- **Se solapa con el principio** del período especificado si el valor de la columna de inicio es inferior a *valor1* y el valor de la columna de fin es superior a *valor1*.
- **Se solapa con el final** del período especificado si el valor de la columna de fin es superior o igual a *valor2* y el valor de la columna de inicio es inferior a *valor2*.
- **Está contenido por completo** en el período especificado si el valor de la columna de inicio para BUSINESS\_TIME es superior o igual a *valor1* y el valor para la columna de fin correspondiente es inferior o igual a *valor2*.
- Está **contenido parcialmente** en el período especificado si la fila se solapa con el principio del período especificado o el fin del período especificado, pero no con ambos.
- **Se solapa por completo** con el período especificado si el período de la fila se solapa con el principio y el fin del período especificado.
- **No está contenido** en el período si las dos columnas de BUSINESS\_TIME son inferiores o iguales a *valor1* o superiores o iguales a *valor2*.

Si el período BUSINESS\_TIME de una fila no está contenido en el período especificado, la fila no se actualiza. De lo contrario, la actualización se aplica en función de cómo los valores de las columnas del período BUSINESS\_TIME se solapan con el período especificado, de la siguiente forma:

- Si el período BUSINESS\_TIME de una fila está contenido por completo dentro del período especificado, la fila se actualiza y los valores de la columna de inicio y de la columna de fin de BUSINESS\_TIME no cambian.
- Si el período BUSINESS\_TIME de una fila está parcialmente contenido en el período especificado y se solapa con el principio del período especificado:
  - La fila se actualiza. En la fila actualizada, el valor de la columna de inicio se establece en *valor1* y el valor de la columna de fin es el valor original de la columna de fin.
  - Se inserta una fila con los valores originales de la fila, salvo que la columna final se establece en *valor1*.
- Si el período BUSINESS\_TIME de una fila está parcialmente contenido en el período especificado y se solapa con el final del período especificado:
  - La fila se actualiza. En la fila actualizada, el valor de la columna de inicio es el valor original de la columna de inicio, y la columna de fin se establece en *valor2*.
  - Se inserta una fila con los valores originales de la fila, salvo que la columna de inicio se establece en *valor2*.
- Si el período BUSINESS\_TIME de una fila se solapa por completo con el período especificado:
  - La fila se actualiza. En la fila actualizada, el valor de la columna de inicio se establece en *valor1* y el valor de la columna de fin se establece en *valor2*.
  - Se inserta una fila con los valores originales de la fila, salvo que la columna final se establece en *valor1*.

## UPDATE

- Se inserta una fila adicional con los valores originales de la fila, salvo que la columna de inicio se establece en *valor2*.

### *valor1 y valor2*

Cada expresión debe devolver un valor que tenga un tipo de datos de fecha, un tipo de datos de indicación de fecha y hora, o un tipo de datos válido para la representación en forma de serie de caracteres de una fecha o indicación de fecha y hora (SQLSTATE 428HY). El resultado de cada expresión debe ser comparable al tipo de datos de las columnas del período especificado (SQLSTATE 42884). Consulte las normas de comparación descritas en “Asignaciones y comparaciones”.

Cada expresión puede contener alguno de los operandos soportados siguientes (SQLSTATE 428HY):

- Constante
- Registro especial
- Variable
- Función escalar cuyos argumentos sean operandos soportados (aunque las funciones definidas por el usuario y las funciones no deterministas no se pueden utilizar)
- Especificación CAST donde el operando de conversión es un operando soportado
- Expresión que utiliza operadores y operandos aritméticos

### **cláusula-correlación**

Se puede utilizar dentro de la *condición-búsqueda* o *cláusula-asignación* para designar una tabla, vista, apodo o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte “referencia-tabla” en la descripción de “Subselección”.

### *columnas-include*

Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia UPDATE cuando está anidada en la cláusula FROM de una selección completa. Las *columnas-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

### **INCLUDE**

Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia UPDATE.

### *nombre-columna*

Especifica una columna de la tabla de resultados intermedia de la sentencia UPDATE. El nombre no puede coincidir con el nombre de otra columna incluye ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

### *tipo-datos*

Especifica el tipo de datos de la columna incluye. El tipo de datos debe ser uno que reciba soporte de la sentencia CREATE TABLE.

### **SET**

Especifica la asignación de valores a nombres de columna.

### *cláusula-asignación*

#### *nombre-columna*

Identifica una columna que se debe actualizar. Si las variables de indicador

ampliado no están habilitadas, el *nombre-columna* debe identificar una columna actualizable de la tabla, vista o apodo que se haya especificado o una columna INCLUDE. 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).

Si especifica una columna INCLUDE, el nombre de columna no se puede cualificar.

Para una sentencia UPDATE con posición:

- Si se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor, cada nombre de columna de la *cláusula-asignación* también debe aparecer en la *cláusula-actualización*.
- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL MIA o SQL92E al compilarse previamente la aplicación, puede especificarse el nombre de cualquier columna que pueda actualizarse.
- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL SAA1 explícitamente o por omisión al compilarse previamente la aplicación, no puede actualizarse 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). Una asignación que no incluye la cláusula *..nombre-atributo* se denomina *asignación convencional*.

#### *expresión*

Indica el nuevo valor de la columna. La expresión es cualquier expresión del tipo que se describe en el apartado "Expresiones". La expresión no puede incluir una función agregada salvo cuando aparece 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.

Una expresión no puede contener referencias a una columna INCLUDE. Si la *expresión* es una sola variable de lenguaje principal, esta puede incluir una variable de indicador que esté habilitada para las variables de indicador ampliado. Si las variables de indicador ampliado están habilitadas, no deben utilizarse los valores para la variable de indicador ampliado por omisión (-5) o no asignado (-7) (SQLSTATE 22539) si cualquiera de las afirmaciones siguientes es cierta:

- La expresión es más compleja que una sola variable del lenguaje principal con conversiones explícitas.
- La columna de destino tiene un tipo de datos de tipo estructurado.

#### 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 que se ha definido para la columna (consulte *cláusula-valor-por-omisión* en “ALTER TABLE”).
- 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).
- Si la columna se ha definido mediante la cláusula ROW CHANGE TIMESTAMP, el gestor de bases de datos genera el valor.

El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).

La palabra clave DEFAULT no se puede utilizar como valor en una asignación de atributos (SQLSTATE 429B9).

La palabra clave DEFAULT no puede utilizarse como valor en una asignación para realizar una actualización en un apodo donde la fuente de datos no da soporte a la sintaxis DEFAULT.

### *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

Especifica 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, vista o apodo.

### *condición-búsqueda*

Cada *nombre-columna* de la condición de búsqueda, a excepción de en una subconsulta, debe especificar el nombre de una columna de la tabla, vista o apodo. 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, vista o apodo y las filas actualizadas son las filas 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 debe utilizar en la operación de actualización. El *nombre-cursor* debe identificar un cursor declarado, que se explica en "DECLARE CURSOR". La sentencia DECLARE CURSOR debe preceder a la sentencia UPDATE en el programa.

La tabla, vista o apodo especificado también debe indicarse en la cláusula FROM de la sentencia SELECT del cursor y la tabla de resultados del cursor no debe ser de sólo lectura. Para ver una explicación sobre las tablas de resultados de sólo lectura, consulte "DECLARE CURSOR".

Cuando se ejecuta la sentencia UPDATE, el cursor debe posicionarse en una fila; dicha fila se actualiza.

Esta forma de UPDATE no se puede utilizar (SQLSTATE 42828) si el cursor hace referencia a:

- Una vista en la que se ha definido un activador INSTEAD OF UPDATE
- Una vista que incluye una función OLAP en la lista de la selección completa que define la vista
- Una vista que se ha definido, directa o indirectamente, utilizando la cláusula WITH ROW MOVEMENT

#### **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. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

#### **Normas**

- **Activadores:** Las sentencias UPDATE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores de la actualización. Si una operación de actualización en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se realizan en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.
- **Asignación:** Los valores de actualización se asignan a las columnas según las normas de asignación específicas.
- **Validez:** La fila actualizada se debe ajustar a cualquier restricción impuesta en la tabla (o en la tabla base de la vista) por cualquier índice exclusivo de 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 obtener información acerca de las normas que rigen esta situación, consulte "CREATE VIEW".

- **Restricción de comprobación:** El valor de actualización debe satisfacer las condiciones de comprobación 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 referencial:** El valor de las claves exclusivas padre no se puede cambiar si la norma de actualización es RESTRICT y hay una o varias filas dependientes. Sin embargo, si la norma de actualización es NO ACTION, las claves exclusivas 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.
- **Valores XML:** Cuando se actualiza un valor de la columna XML, el nuevo valor debe ser un documento XML con el formato correcto (SQLSTATE 2200M).
- **Política de seguridad:** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
  - Acceso de grabación a todas las columnas protegidas que se están actualizando (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)
  - Acceso de grabación y de lectura a todas las filas que se están actualizando (SQLSTATE 42519)

El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:

- La columna DB2SECURITYLABEL no está incluida en la lista de columnas que se van a actualizar (y, por lo tanto, se actualizará implícitamente en la etiqueta de seguridad para el acceso de grabación del ID de autorización de la sesión)
  - Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
- **Uso de variable de indicador ampliado:** Si se habilita, los valores de variable de indicador distintos de 0 (cero) hasta -7 no pueden ser una entrada (SQLSTATE

22010). Además, si se habilitan, los valores DEFAULT y UNASSIGNED de variable de indicador ampliado no deben aparecer en contextos en los que no estén soportados (SQLSTATE 22539).

- **Variables de indicador ampliado:** En la *cláusula-asignación* de una sentencia UPDATE, una *expresión* que sea una referencia a una sola variable de lenguaje principal o una variable de lenguaje principal que se vaya a convertir de forma explícita puede dar lugar a un valor de variable de indicador ampliado. La asignación de un valor UNASSIGNED basado en una variable de indicador ampliado tiene como efecto dejar que la columna de destino se establezca en su valor por omisión, como si no se hubiera especificado en la sentencia. La asignación de un valor DEFAULT basado en una variable de indicador ampliado asigna el valor por omisión de la columna. Para obtener información acerca de los valores por omisión de los tipos de datos, consulte la descripción de la cláusula DEFAULT en “CREATE TABLE” en la página 722.

Si una columna de destino no es actualizable (por ejemplo, una columna en una vista que esté definida como una expresión), se le debe asignar el valor UNASSIGNED basado en una variable de indicador ampliado (SQLSTATE 42808).

Si la columna de destino es una columna definida como GENERATED ALWAYS, se le debe asignar la palabra clave DEFAULT o los valores DEFAULT o UNASSIGNED basados en una variable de indicador ampliado (SQLSTATE 428C9).

La sentencia UPDATE no debe asignar a todas las columnas de destino un valor UNASSIGNED basado en una variable de indicador ampliado (SQLSTATE 22540).

## 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.
- Una actualización de una vista definida utilizando la cláusula WITH ROW MOVEMENT puede ocasionar una operación de supresión y una operación de inserción contra las tablas subyacentes de la vista. Para ver detalles, consulte la descripción de la sentencia CREATE VIEW.
- Cuando una sentencia UPDATE completa su ejecución, el valor de SQLERRD(3) de la SQLCA es el número de filas que se han calificado para la operación de actualización. En el contexto de una sentencia de procedimiento de SQL, el valor puede recuperarse utilizando la variable ROW\_COUNT de la sentencia GET DIAGNOSTICS. El campo SQLERRD(5) contiene el número de filas insertadas, suprimidas o actualizadas por todos los activadores activados.
- 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.
- Cuando se actualicen las estadísticas de distribución de columna para una tabla con tipo, debe especificarse la subtabla que haya incluido 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 que se ha definido como un tipo diferenciado, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior en el tipo de fuente antes de realizarse el cálculo.)
- 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.

- Para obtener más información acerca de la utilización máxima de los valores de una secuencia generada para una columna de identidad o acerca de cuándo se excede el valor máximo de una columna de identidad, consulte "INSERT".
- Con las tablas particionadas, la operación *nombre-cursor* UPDATE WHERE CURRENT OF puede mover una fila de una partición de datos a otra. Después de este proceso, el cursor ya no se encuentra sobre la fila y ya no se pueden realizar más modificaciones de *nombre-cursor* UPDATE WHERE CURRENT OF para dicha fila. No obstante, puede captarse la siguiente fila del cursor.
- Para una columna definida mediante la cláusula ROW CHANGE TIMESTAMP, el valor siempre se cambia al actualizar la fila. Si la columna no se especifica en la lista SET explícitamente, el gestor de bases de datos todavía genera un valor para dicha fila. El valor es exclusivo para cada partición de tabla de la partición de base de datos y se establece en la indicación de fecha y hora aproximada correspondiente a la actualización de fila.
- **Variables de indicador ampliado y activadores de actualización:** Si a una columna de destino se le ha asignado un valor UNASSIGNED basado en una variable de indicador ampliado, la columna no se considera actualizada. Esa columna se trata como si no se hubiera especificado en la lista OF *nombre-columna* de cualquier activador de actualización definido en la tabla de destino.
- **Variables de indicador ampliado y comprobaciones de errores diferidas:** Cuando se habilitan variables de indicador ampliado, la validación que de otro modo se realizaría en la preparación de sentencias, para reconocer una actualización de una columna no actualizable, se difiere hasta la ejecución de las sentencias, excepto para la comprobación del privilegio de actualización de nivel de columna de las sentencias UPDATE estáticas. Si ha de informarse o no de un error sólo puede determinarse durante la ejecución basada en el valor del indicador. La comprobación del privilegio de actualización de nivel de columna para las sentencias UPDATE estáticas sigue realizándose durante el proceso de vinculación, incluso cuando se han habilitado variables de indicador ampliado.
- **Consideraciones para una tabla temporal de período de sistema::** El destino de la sentencia UPDATE no debe ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para SYSTEM\_TIME si la vista se ha definido con WITH CHECK OPTION y la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):



- Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente)
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Si el registro especial CURRENT TEMPORAL SYSTEM\_TIME se establece en un valor no nulo, el destino subyacente de la sentencia UPDATE no debe ser una tabla temporal de período de sistema (SQLSTATE 51046), y el destino de la sentencia UPDATE no debe ser una vista definida con WITH CHECK OPTION si la definición de vista incluye cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente)
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Cuando una fila de una tabla temporal de período de sistema se actualiza, el gestor de bases de datos actualiza los valores de las columnas de inicio de fila y de ID de inicio de transacción de la forma siguiente:

- A una columna de inicio de fila se le asigna un valor que se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exige que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla, o que se suprima una tabla temporal de período de sistema. El gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de inicio de fila en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas insertadas en una tabla histórica asociada tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio, lo que puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema. El parámetro de configuración de base de datos **system\_time\_period\_adj** debe establecerse en Yes para que tenga lugar este ajuste en el valor de indicación de fecha y hora. Si se actualizan varias filas en una sola transacción de SQL y no es necesario un ajuste, los valores de la columna de inicio de fila son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.
- A una columna de ID de inicio de transacción se le asigna un valor de indicación de fecha y hora exclusivo o el valor nulo. El valor nulo se asigna a la columna de ID de inicio de transacción si la columna puede contener nulos y si existe una columna de inicio de fila en la tabla en la que el valor no tiene que ajustarse. Si no, el valor se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exija que se asigne un valor a la columna de inicio de fila o a la columna de ID de inicio de transacción de una tabla, o que se suprima una tabla temporal de período de sistema. Si se actualizan varias filas en una sola transacción de SQL, los valores de la columna de ID de inicio de transacción son iguales para todas las filas y son exclusivos entre los valores generados para la columna en otra transacción.

Si la sentencia UPDATE tiene una condición de búsqueda que contiene una subconsulta correlacionada que hace referencia a filas históricas (con referencia explícita al nombre de tabla histórica o implícita a través del uso de una especificación de período en la cláusula FROM), la versión anterior de las filas actualizadas que se inserten como filas históricas (en la tabla histórica, si la

hubiera) se pueden visualizar para actualizar operaciones para las filas procesadas posteriormente para la sentencia.

El destino de la sentencia UPDATE no puede ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para SYSTEM\_TIME si se cumplen las dos condiciones siguientes (SQLSTATE 51046):

- La vista está definida con WITH CHECK OPTION.
- La definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes:
  - Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente).
  - Una invocación de una rutina de SQL que tiene un paquete asociado a ella.
  - Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL.

Si el registro especial CURRENT TEMPORAL SYSTEM\_TIME se establece en un valor no nulo, el destino subyacente (directo o indirecto) de la sentencia UPDATE no puede ser una tabla temporal de período de sistema (SQLSTATE 51046).

Si el registro especial CURRENT TEMPORAL SYSTEM\_TIME se establece en un valor no nulo, el destino de la sentencia UPDATE no puede ser una vista definida con WITH CHECK OPTION si la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de sistema (directa o indirectamente).
  - Una invocación de una rutina de SQL que tiene un paquete asociado a ella.
  - Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL.
- **Consideraciones sobre una tabla histórica:** cuando se actualiza una fila de una tabla temporal de período de sistema, se inserta una copia histórica de la fila en la tabla histórica correspondiente y la indicación de fecha y hora final de la fila histórica se captura en forma de valor determinado por el sistema que se corresponde con la hora de la operación de cambio de datos. El gestor de bases de datos asigna el valor que se genera a partir de la lectura del reloj durante la ejecución de la primera sentencia de cambio de datos en la transacción que exige que se asigne un valor a la columna de inicio de fila o de ID de inicio de transacción de una tabla, o que se suprima una fila de una tabla temporal de período de sistema. El gestor de bases de datos garantiza la exclusividad de los valores generados para una columna de fin de una tabla histórica en las transacciones. El valor de indicación de fecha y hora puede ajustarse para garantizar que las filas insertadas en la tabla histórica tengan un valor de indicación de fecha y hora de fin superior al valor de indicación de fecha y hora de inicio, lo que puede suceder cuando una transacción conflictiva actualiza la misma fila en la tabla temporal de período de sistema (SQLSTATE 01695). El parámetro de configuración de base de datos **system\_time\_period\_adj** debe establecerse en Yes para que tenga lugar este ajuste en el valor de indicación de fecha y hora.

En el caso de una operación de actualización, el ajuste sólo afecta al valor de la columna de fin correspondiente a la columna de fin de fila de la tabla histórica asociada con la tabla temporal de período de sistema. Tenga en cuenta estos ajustes en las referencias posteriores a la tabla exista o no una búsqueda de la

hora de inicio de la transacción en los valores de las columnas correspondientes a las columnas de inicio de fila y de fin de fila de la tabla temporal de período de sistema asociada.

- **Consideraciones para una tabla temporal de período de aplicación:** El destino de la sentencia UPDATE no debe ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para BUSINESS\_TIME si la vista se ha definido con WITH CHECK OPTION y la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):
  - Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente)
  - Una invocación de una rutina de SQL que tiene un paquete asociado a ella
  - Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Si el registro especial CURRENT TEMPORAL BUSINESS\_TIME se establece en un valor no nulo, el destino de la sentencia UPDATE no debe ser una vista definida con WITH CHECK OPTION si la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente)
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL

Una sentencia UPDATE para una tabla temporal de período de aplicación que contiene una cláusula FOR PORTION OF BUSINESS\_TIME indica entre qué dos puntos en el tiempo están en vigor las supresiones especificadas. Cuando se especifica FOR PORTION OF BUSINESS\_TIME y el valor de período de una fila, especificado por los valores de columna de inicio de fila y columna de fin de fila, sólo está contenido parcialmente en el período especificado desde *valor1* hasta *valor2*, la fila se actualiza y una o dos filas se insertan de forma automática para representar la parte de la fila que no ha cambiado. Se generan nuevos valores para cada columna generada en una tabla temporal de período de aplicación para cada fila que se inserta de forma automática como resultado de una operación de actualización en la tabla. Si se define una columna generada como parte de una clave exclusiva o primaria, una clave principal de una restricción referencial o un índice exclusivo, es posible que una inserción automática viole una restricción o índice, en cuyo caso se devolverá un error.

Cuando se inserta una fila en una tabla temporal de período de aplicación que tiene una clave primaria o una restricción de unicidad con la cláusula BUSINESS\_TIME WITHOUT OVERLAPS definida o un índice exclusivo con la cláusula BUSINESS\_TIME WITHOUT OVERLAPS definida, si el período definido por las columnas de inicio y fin del período BUSINESS\_TIME se solapa con el período definido por las columnas de inicio y fin del período BUSINESS\_TIME de otra fila con la misma restricción de unicidad o índice exclusivo en la tabla, se devuelve un error.

El destino de la sentencia UPDATE no puede ser una selección completa que haga referencia a una vista en la cláusula FROM seguida de una especificación de período para BUSINESS\_TIME si se cumplen las dos condiciones siguientes (SQLSTATE 51046):

- La vista está definida con WITH CHECK OPTION.
- La definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes:

## UPDATE

- Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente).
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella.
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL.

Si el registro especial CURRENT TEMPORAL BUSINESS\_TIME se establece en un valor no nulo, el destino de la sentencia UPDATE no puede ser una vista definida con WITH CHECK OPTION si la definición de vista incluye una cláusula WHERE que contiene uno de los elementos de sintaxis siguientes (SQLSTATE 51046):

- Una subconsulta que hace referencia a una tabla temporal de período de aplicación (directa o indirectamente).
- Una invocación de una rutina de SQL que tiene un paquete asociado a ella.
- Una invocación de una rutina externa con una indicación de acceso a los datos que no es NO SQL.

Cuando una tabla temporal de período de aplicación es el destino de una sentencia UPDATE, el valor en vigor para el registro especial CURRENT TEMPORAL BUSINESS\_TIME no es el valor nulo y la opción de vinculación BUSTIMESENSITIVE está establecida en YES, están implícitos los predicados adicionales siguientes:

```
bt_begin <= CURRENT TEMPORAL BUSINESS_TIME
AND bt_end > CURRENT TEMPORAL BUSINESS_TIME
```

donde bt\_begin y bt\_end son las columnas inicial y final del período BUSINESS\_TIME de la tabla de destino de la sentencia UPDATE.

- **Consideraciones sobre tablas temporales de período de aplicación y activadores:** Cuando se actualiza una fila y se especifica la cláusula FOR PORTION OF BUSINESS\_TIME, se pueden insertar implícitamente filas adicionales para reflejar cualquier porción de la fila que no haya sido actualizada. Los activadores de actualización existentes se activan para las filas actualizadas y los activadores de inserción existentes se activan para las filas que se han insertado implícitamente.

## Ejemplos

- *Ejemplo 1:* Cambiar el trabajo (JOB) del empleado número (EMPNO) '000290' en la tabla EMPLOYEE por 'LABORER'.

```
UPDATE EMPLOYEE
SET JOB = 'LABORER'
WHERE EMPNO = '000290'
```

- *Ejemplo 2:* Incrementar las personas que trabajan en el proyecto (PRSTAFF) en 1,5 para todos los proyectos de los que es 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 reasignado temporalmente. Indique esto cambiando su trabajo (JOB) por el valor null y los valores de su paga (SALARY, BONUS, COMM) por 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 (SELECT EMPNO, SALARY, COMM,
 AVG(SALARY) OVER (PARTITION BY WORKDEPT),
 AVG(COMM) OVER (PARTITION BY WORKDEPT)
 FROM EMPLOYEE E) AS E(EMPNO, SALARY, COMM, AVGSAL, AVGCOMM)
 SET (SALARY, COMM) = (AVGSAL, AVGCOMM)
 WHERE EMPNO = '000120'
```

La sentencia anterior es semánticamente equivalente a la sentencia siguiente, pero sólo necesita un acceso a la tabla EMPLOYEE, mientras que la sentencia siguiente especifica la tabla EMPLOYEE dos veces.

```
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

```
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
```

- *Ejemplo 7:* Actualice la columna XMLDOC de la tabla DOCUMENTS con DOCID '001' a la serie de caracteres seleccionada y analizada a partir de la tabla XMLTEXT.

```
UPDATE DOCUMENTS SET XMLDOC =
 (SELECT XMLPARSE(DOCUMENT C1 STRIP WHITESPACE)
 FROM XMLTEXT WHERE TEXTID = '001')
WHERE DOCID = '001'
```

---

**VALUES**

La sentencia VALUES es una forma de consulta.

La sentencia VALUES puede incorporarse en un programa de aplicación o emitirse interactivamente.





*expresión-fila*

Especifica la fila de valores nueva. La *expresión-fila* es cualquier expresión de fila del tipo descrito en "Expresiones de fila". La *expresión-fila* no debe incluir un nombre de columna.

**INTO** *destino-asignación*

Identifica uno o varios destinos para la asignación de los valores de salida.

El primer valor de la fila del resultado se asigna al primer destino de la lista, el segundo valor al segundo destino, etcétera. Cada asignación que se realiza para un *destino-asignación* se realiza secuencialmente y siguiendo la lista. Si se produce un error en cualquier asignación, no se asigna ningún valor a ningún *destino-asignación*.

Cuando el tipo de datos de cada *destino-asignación* no es un tipo de fila, se asigna el valor 'W' al campo SQLWARN3 del SQLCA si el número de *destinos-asignación* es menor que el número de valores de la columna de resultados.

Si el tipo de datos de un *destino-asignación* es un tipo de fila, debe haber exactamente un *destino-asignación* especificado (SQLSTATE 428HR), el número de columnas debe coincidir con el número de campos en el tipo de fila y los tipos de datos de las columnas de la fila captada deben poder asignarse a los campos correspondientes del tipo de fila (SQLSTATE 42821).

Si el tipo de datos de un *destino-asignación* es un elemento de matriz, debe haber exactamente un *destino-asignación* especificado.

*nombre-variable-global*

Identifica la variable global que es el sujeto de la asignación.

*nombre-variable-lenguaje-principal*

Identifica la variable del lenguaje principal que es el sujeto de la asignación. Para los valores de salida LOB, el destino puede ser una variable del lenguaje principal normal (si es lo suficientemente grande), una variable de localizador LOB o una variable de referencia a archivos LOB.

*nombre-parámetro-SQL*

Identifica el parámetro de nombre que es el destino de la asignación.

*nombre-variable-SQL*

Identifica la variable de SQL que es el sujeto de la asignación. Las variables de SQL se deben declarar antes de utilizarlas.

*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 activador, calificado opcionalmente por un nombre de correlación que identifica el nuevo valor.

*nombre-variable-matriz*

Identifica una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz.

*[índice-matriz]*

Expresión que especifica qué elemento de la matriz será el destino de la asignación. Para una matriz común, la expresión de *índice-matriz* se debe poder asignar a INTEGER (SQLSTATE 428H1) y no puede ser un valor nulo. Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz (SQLSTATE 2202E). Para una matriz asociativa,

## VALUES INTO

la expresión de *índice-matriz* se debe poder asignar al tipo de datos de índice de la matriz asociativa (SQLSTATE 428H1) y no puede ser un valor nulo.

### *referencia-campo*

Identifica el campo de un valor de tipo de fila que es el destino de la asignación. La *referencia-campo* debe especificarse como un *nombre-campo* calificado donde el calificador identifica el valor de fila donde está definido el campo.

## Normas

- Las variables globales no se pueden asignar dentro de activadores que no se han definido mediante una sentencia de SQL compuesto (compilado), funciones que no se han definido mediante una sentencia de SQL compuesto (compilado), métodos o sentencias de SQL compuesto (en línea) (SQLSTATE 428GX).

## 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;
```

- *Ejemplo 3:* este ejemplo C recupera el valor del registro especial SESSION\_USER en una variable global.

```
EXEC SQL VALUES (SESSION_USER)
 INTO GV_SESS_USER;
```

## WHENEVER

La sentencia WHENEVER especifica la acción que se debe tomar cuando se produce una condición de excepción especificada.

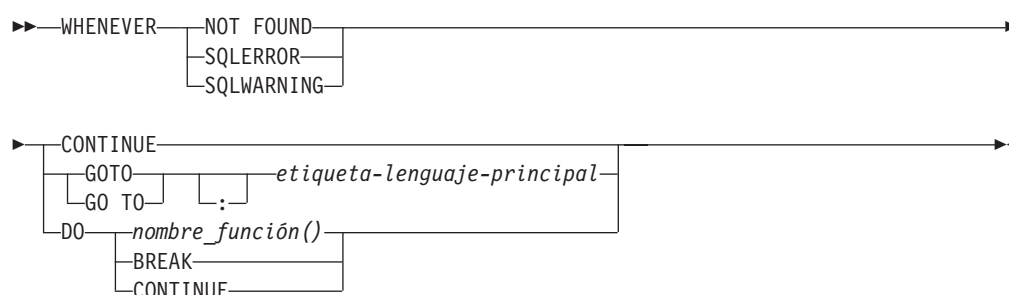
### Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. La sentencia no se soporta 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-lenguaje-principal*

Provoca que el control pase a la sentencia identificada por la etiqueta-lenguaje-principal. Para etiqueta-lenguaje-principal, utilice un símbolo individual, precedido opcionalmente por dos puntos. El formato del símbolo depende del lenguaje principal.

**DO** Da lugar a una acción adicional en forma de una llamada de función, sentencia break o sentencia continue.

## WHENEVER

### *nombre\_función()*

Especifica la función C que ha de llamarse. La función debe tener un valor de retorno nulo y no puede aceptar ningún argumento. El nombre de función debe acabar con un conjunto de paréntesis "(" y ")". El nombre de la función tiene una limitación de 255 bytes.

La resolución de nombre de función tiene lugar durante la compilación de la aplicación de SQL incorporado C y C++. El precompilador de DB2 no resuelve el nombre de función.

### **BREAK**

Especifica la sentencia break de C. La sentencia break de C sale del bloqueo de sentencia do, for, switch o while.

### **CONTINUE**

Especifica la sentencia continue de C. La sentencia continue de C pasa el control a la siguiente iteración del bloqueo de sentencia do, for, switch o while.

## 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.

Si no se utiliza la sentencia WHENEVER, la acción por omisión consiste en continuar el proceso si se produce una condición de error, de aviso o de excepción durante la ejecución.

La sentencia WHENEVER debe utilizarse antes que las sentencias de SQL que desea afectar. De otro modo, el precompilador no sabe que se necesita código de gestión de errores adicional para las sentencias de SQL ejecutables. En un momento cualquiera puede estar activa una combinación cualquiera de los tres formatos básicos. El orden en el que declare los tres formatos no es significativo.

Para evitar una situación de bucle infinito, asegúrese de deshacer el manejo WHENEVER antes de que se ejecute alguna sentencia de SQL dentro del manejador. Puede deshacer el manejo WHENEVER utilizando la sentencia WHENEVER SQLERROR CONTINUE.

El soporte de la sentencia WHENEVER para utilizar la sintaxis DO *nombre\_función()*, DO BREAK o DO CONTINUE está disponible en la Versión 9.7, Fixpack 6 y posteriores.

## 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;
```

Ejemplos de C para utilizar la sintaxis `DO nombre_función()`, `DO BREAK` o `DO CONTINUE` son:

```
/* DO function_name */
EXEC SQL WHENEVER SQLERROR DO perform_error_action();
EXEC SQL WHENEVER SQLWARNING DO perform_warning_action();
EXEC SQL WHENEVER NOT FOUND DO perform_notfound_action();

/* DO BREAK */
EXEC SQL WHENEVER SQLERROR DO BREAK;
EXEC SQL WHENEVER SQLWARNING DO BREAK;
EXEC SQL WHENEVER NOT FOUND DO BREAK;

/* DO CONTINUE */
EXEC SQL WHENEVER SQLERROR DO CONTINUE;
EXEC SQL WHENEVER SQLWARNING DO CONTINUE;
EXEC SQL WHENEVER NOT FOUND DO CONTINUE;
```

## WHILE

La sentencia WHILE repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

### Invocación

Esta sentencia se puede incluir en:

- una definición de procedimiento de SQL
- una sentencia de SQL compuesto (compilado)
- una sentencia de SQL compuesto (en línea)

Las sentencias compuestas pueden incorporarse en una definición de procedimiento, función o activador de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

### Autorización

No se requieren privilegios para invocar la sentencia WHILE. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia WHILE.

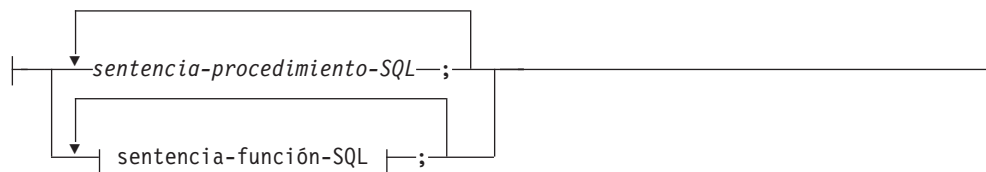
### Sintaxis

```

 ┌───────────┐ ┌───┐ ┌───────────┐ ┌───┐ ┌───────────┐ ┌───┐ ┌───────────┐ ┌───┐
 │ etiqueta: │ WHILE │ condición-búsqueda-DO │ sentencia-rutina-SQL │ END WHILE │ etiqueta │
 └───────────┘ └───┘ └───────────┘ └───┘ └───────────┘ └───┘ └───────────┘ └───┘

```

#### sentencia-rutina-SQL:



### 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 una etiqueta final, ésta deberá 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 de SQL incluidas en el bucle.

#### sentencia-procedimiento-SQL

Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL o una sentencia de SQL compuesto (compilado). Consulte *sentencia-procedimiento-SQL* en la sentencia de "SQL compuesto (compilado)".

*sentencia-función-SQL*

Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en una función de SQL o una sentencia de SQL compuesto (en línea) que se puede incorporar en un activador de SQL, una función de SQL o un método de SQL. Consulte *sentencia-función-SQL* en “FOR”.

**Ejemplo**

Este ejemplo utiliza una sentencia WHILE para ejecutar un proceso iterativo mediante sentencias FETCH y SET. Mientras el valor de la variable de SQL *v\_counter* sea menor que la mitad del número de empleados del departamento que identifica el parámetro IN *deptNumber*, la sentencia WHILE seguirá realizando 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
 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. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible en diversos formatos a los que se puede acceder de varias maneras.

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- DB2Centro de información
  - Temas (Tareas, concepto y temas de consulta)
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (desde el DVD con PDF de DB2)
  - Manuales en copia impresa
- Ayuda de la línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes

**Nota:** Los temas del Centro de información de DB2 se actualizan con más frecuencia que los manuales en PDF o impresos. Para obtener la información más actualizada, instale las actualizaciones de la documentación conforme pasen a estar disponibles, o consulte el Centro de información de DB2 en [ibm.com](http://ibm.com).

Puede acceder a información técnica adicional de DB2 como, por ejemplo, notas técnicas, documentos técnicos y publicaciones IBM Redbooks en línea, en el sitio [ibm.com](http://ibm.com). Acceda al sitio de la biblioteca de software de gestión de información de DB2 en <http://www.ibm.com/software/data/sw-library/>.

### Comentarios sobre la documentación

Agradecemos los comentarios sobre la documentación de DB2. Si tiene sugerencias sobre cómo podemos mejorar la documentación de DB2, envíe un correo electrónico a [db2docs@ca.ibm.com](mailto:db2docs@ca.ibm.com). El personal encargado de la documentación de DB2 lee todos los comentarios de los usuarios, pero no puede responderlos directamente. Proporcione ejemplos específicos siempre que sea posible de manera que podamos comprender mejor sus problemas. Si realiza comentarios sobre un tema o archivo de ayuda determinado, incluya el título del tema y el URL.

No utilice esta dirección de correo electrónico para contactar con el Soporte al cliente de DB2. Si tiene un problema técnico de DB2 que no está tratado por la documentación, consulte al centro local de servicio técnico de IBM para obtener ayuda.

## Biblioteca técnica de DB2 en copia impresa o en formato PDF

Las tablas siguientes describen la biblioteca de DB2 que está disponible en el Centro de publicaciones de IBM en [www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss). Los manuales de DB2 Versión 10.1 en inglés y las versiones traducidas en formato PDF se pueden descargar del sitio web [www.ibm.com/support/docview.wss?rs=71&uid=swg27009474](http://www.ibm.com/support/docview.wss?rs=71&uid=swg27009474).

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

El número de documento se incrementa cada vez que se actualiza un manual. Asegúrese de que lee la versión más reciente de los manuales, tal como aparece a continuación:

**Nota:** El Centro de información de DB2 se actualiza con más frecuencia que los manuales en PDF o impresos.

Tabla 37. Información técnica de DB2

| Nombre                                                              | Número de documento | Copia impresa disponible | Última actualización |
|---------------------------------------------------------------------|---------------------|--------------------------|----------------------|
| <i>Consulta de las API administrativas</i>                          | SC11-8067-00        | Sí                       | Abril de 2012        |
| <i>Rutinas y vistas administrativas</i>                             | SC11-8068-01        | No                       | Enero de 2013        |
| <i>Call Level Interface Guide and Reference Volume 1</i>            | SC27-3866-01        | Sí                       | Enero de 2013        |
| <i>Call Level Interface Guide and Reference Volume 2</i>            | SC27-3867-01        | Sí                       | Enero de 2013        |
| <i>Consulta de mandatos</i>                                         | SC11-8069-01        | Sí                       | Enero de 2013        |
| <i>Database Administration Concepts and Configuration Reference</i> | SC27-3871-01        | Sí                       | Enero de 2013        |
| <i>Data Movement Utilities Guide and Reference</i>                  | SC27-3869-01        | Sí                       | Enero de 2013        |
| <i>Database Monitoring Guide and Reference</i>                      | SC27-3887-01        | Sí                       | Enero de 2013        |
| <i>Data Recovery and High Availability Guide and Reference</i>      | SC27-3870-01        | Sí                       | Enero de 2013        |
| <i>Database Security Guide</i>                                      | SC27-3872-01        | Sí                       | Enero de 2013        |
| <i>Guía y consulta de DB2 Workload Management</i>                   | SC11-8079-01        | Sí                       | Enero de 2013        |
| <i>Developing ADO.NET and OLE DB Applications</i>                   | SC27-3873-01        | Sí                       | Enero de 2013        |
| <i>Developing Embedded SQL Applications</i>                         | SC27-3874-01        | Sí                       | Enero de 2013        |
| <i>Desarrollo de aplicaciones Java</i>                              | SC11-8065-01        | Sí                       | Enero de 2013        |

Tabla 37. Información técnica de DB2 (continuación)

| Nombre                                                                          | Número de documento | Copia impresa disponible | Última actualización |
|---------------------------------------------------------------------------------|---------------------|--------------------------|----------------------|
| <i>Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails</i>             | SC11-8066-00        | No                       | Abril de 2012        |
| <i>Desarrollo de aplicaciones RDF para servidores de datos IBM</i>              | SC11-8315-00        | Sí                       | Enero de 2013        |
| <i>Developing User-defined Routines (SQL and External)</i>                      | SC27-3877-01        | Sí                       | Enero de 2013        |
| <i>Getting Started with Database Application Development</i>                    | GI13-2046-01        | Sí                       | Enero de 2013        |
| <i>Iniciación a la instalación y administración de DB2 en Linux y Windows</i>   | GI13-1946-00        | Sí                       | Abril de 2012        |
| <i>Globalization Guide</i>                                                      | SC27-3878-00        | Sí                       | Abril de 2012        |
| <i>Instalación de servidores DB2</i>                                            | GC11-8073-01        | Sí                       | Enero de 2013        |
| <i>Instalación de clientes de IBM Data Server</i>                               | GC11-8074-00        | No                       | Abril de 2012        |
| <i>Consulta de mensajes Volumen 1</i>                                           | SC11-8079-01        | No                       | Enero de 2013        |
| <i>Consulta de mensajes Volumen 2</i>                                           | SC11-8080-01        | No                       | Enero de 2013        |
| <i>Net Search Extender Guía de administración y del usuario</i>                 | SC11-8082-01        | No                       | Enero de 2013        |
| <i>Partitioning and Clustering Guide</i>                                        | SC27-3882-01        | Sí                       | Enero de 2013        |
| <i>Preparation Guide for DB2 10.1 Fundamentals Exam 610</i>                     | SC27-4540-00        | No                       | Enero de 2013        |
| <i>Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611</i> | SC27-4541-00        | No                       | Enero de 2013        |
| <i>pureXML Guide</i>                                                            | SC27-3892-01        | Sí                       | Enero de 2013        |
| <i>Spatial Extender Guía del usuario y manual de consulta</i>                   | SC11-8081-00        | No                       | Abril de 2012        |
| <i>SQL Procedural Languages: Application Enablement and Support</i>             | SC27-3896-01        | Sí                       | Enero de 2013        |
| <i>Consulta de SQL - Volumen 1</i>                                              | SC11-8070-01        | Sí                       | Enero de 2013        |
| <i>Consulta de SQL - Volumen 2</i>                                              | SC11-8071-01        | Sí                       | Enero de 2013        |
| <i>Guía de Text Search</i>                                                      | SC11-3888-01        | Sí                       | Enero de 2013        |

Tabla 37. Información técnica de DB2 (continuación)

| Nombre                                                 | Número de documento | Copia impresa disponible | Última actualización |
|--------------------------------------------------------|---------------------|--------------------------|----------------------|
| <i>Troubleshooting and Tuning Database Performance</i> | SC27-3889-01        | Sí                       | Enero de 2013        |
| <i>Actualización a DB2 Versión 10.1</i>                | SC11-8072-01        | Sí                       | Enero de 2013        |
| <i>Novedades en DB2 Versión 10.1</i>                   | SC11-8078-01        | Sí                       | Enero de 2013        |
| <i>XQuery Reference</i>                                | SC27-3893-01        | No                       | Enero de 2013        |

Tabla 38. Información técnica específica de DB2 Connect

| Nombre                                                                         | Número de documento | Copia impresa disponible | Última actualización |
|--------------------------------------------------------------------------------|---------------------|--------------------------|----------------------|
| <i>DB2 Connect Instalación y configuración de DB2 Connect Personal Edition</i> | SC11-8075-00        | Sí                       | Abril de 2012        |
| <i>DB2 Connect Instalación y configuración de servidores DB2 Connect</i>       | SC11-8076-01        | Sí                       | Enero de 2013        |
| <i>Guía del usuario de DB2 Connect</i>                                         | SC11-8077-01        | Sí                       | Enero de 2013        |

## Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

Los productos DB2 devuelven un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

### Procedimiento

Para iniciar la ayuda para estados de SQL, abra el procesador de línea de mandatos y entre:

```
? sqlstate o ? código de clase
```

donde *sqlstate* representa un estado de SQL válido de cinco dígitos y *código de clase* representa los dos primeros dígitos del estado de SQL.

Por ejemplo, ? 08003 visualiza la ayuda para el estado de SQL 08003, y ? 08 visualiza la ayuda para el código de clase 08.

## Acceso a diferentes versiones del Centro de información de DB2

La documentación correspondiente a otras versiones de los productos DB2 se encuentra en otros centros de información en [ibm.com](http://ibm.com).

### Acerca de esta tarea

Para los temas de DB2 Versión 10.1, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1>.

## Acceso a diferentes versiones del Centro de información de DB2

Para los temas de DB2 Versión 9.8, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/>.

Para los temas de DB2 Versión 9.7, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/>.

Para los temas de DB2 Versión 9.5, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>.

Para los temas de DB2 Versión 9.1, el URL del *Centro de información de DB2* es <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>.

Para los temas de DB2 Versión 8, vaya al URL del *Centro de información de DB2* en el sitio: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

---

## Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

El Centro de información de DB2 instalado en local se debe actualizar periódicamente.

### Antes de empezar

Ya debe haber un Centro de información de DB2 Versión 10.1 instalado. Para obtener información adicional, consulte el tema “Instalación del Centro de información de DB2 utilizando el Asistente de instalación de DB2” en la publicación *Instalación de servidores DB2*. Todos los requisitos previos y las restricciones aplicables a la instalación del Centro de información se aplican también a la actualización del Centro de información.

### Acerca de esta tarea

Un Centro de información de DB2 existente se puede actualizar automática o manualmente:

- Las actualizaciones automáticas actualizan las funciones y los idiomas del Centro de información existentes. Una ventaja de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo más breve a cuando se realiza la actualización manual. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Las actualizaciones manuales se pueden utilizar para actualizar las funciones y los idiomas existentes del Centro de información. Las actualizaciones automáticas reducen el tiempo de inactividad durante el proceso de actualización. Sin embargo, debe utilizar el proceso manual cuando desee añadir funciones o idiomas. Por ejemplo, un Centro de información en local se instaló inicialmente tanto en inglés como en francés, y ahora se desea instalar el idioma alemán. Con la actualización manual, se instalará el alemán y se actualizarán además las funciones y los idiomas existentes del Centro de información. No obstante, la actualización manual requiere que el usuario detenga, actualice y reinicie manualmente el Centro de información. El Centro de información no está disponible durante todo el proceso de actualización. En el proceso de actualización automática, el Centro de información incurre en una interrupción de servicio para reiniciar el Centro de información solo después de la actualización.

## Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Este tema detalla el proceso de las actualizaciones automáticas. Para conocer las instrucciones para la actualización manual, consulte el tema “Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet”.

### Procedimiento

Para actualizar automáticamente el Centro de información de DB2 instalado en el sistema o en el servidor de Intranet:

1. En sistemas operativos Linux,
  - a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `/opt/ibm/db2ic/V10.1`.
  - b. Navegue desde el directorio de instalación al directorio `doc/bin`.
  - c. Ejecute el script `update-ic`:  
`update-ic`
2. En sistemas operativos Windows,
  - a. Abra una ventana de mandatos.
  - b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `<Archivos de programa>\IBM\DB2 Information Center\Versión 10.1`, siendo `<Archivos de programa>` la ubicación del directorio Archivos de programa.
  - c. Navegue desde el directorio de instalación al directorio `doc\bin`.
  - d. Ejecute el archivo `update-ic.bat`:  
`update-ic.bat`

### Resultados

El Centro de información de DB2 se reinicia automáticamente. Si hay actualizaciones disponibles, el Centro de información muestra los temas nuevos y actualizados. Si no había actualizaciones del Centro de información disponibles, se añade un mensaje al archivo de anotaciones cronológicas. El archivo de anotaciones cronológicas está ubicado en el directorio `doc\eclipse\configuration`. El nombre del archivo de anotaciones cronológicas es un número generado aleatoriamente. Por ejemplo, `1239053440785.log`.

---

## Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Si ha instalado localmente el Centro de información de DB2 localmente, puede obtener e instalar actualizaciones de la documentación de IBM.

### Acerca de esta tarea

Para actualizar manualmente el *Centro de información de DB2* instalado localmente es preciso que:

1. Detenga el *Centro de información de DB2* en el sistema, y reinicie el Centro de información en modalidad autónoma. La ejecución del Centro de información en modalidad autónoma impide que otros usuarios de la red accedan al Centro de información y permite al usuario aplicar las actualizaciones. La versión de estación de trabajo del Centro de información de DB2 siempre se ejecuta en modalidad autónoma.

2. Utilice la función Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que debe instalar, puede utilizar la función Actualizar para obtenerlas y actualizarlas.

**Nota:** Si su entorno requiere la instalación de actualizaciones del *Centro de información de DB2* en una máquina no conectada a Internet, duplique el sitio de actualizaciones en un sistema de archivos local utilizando una máquina que esté conectada a Internet y tenga instalado el *Centro de información de DB2*. Si muchos usuarios en la red van a instalar las actualizaciones de la documentación, puede reducir el tiempo necesario para realizar las actualizaciones duplicando también el sitio de actualizaciones localmente y creando un proxy para el sitio de actualizaciones.

Si hay paquetes de actualización disponibles, utilice la característica Actualizar para obtener los paquetes. Sin embargo, la característica Actualizar sólo está disponible en modalidad autónoma.

3. Detenga el Centro de información autónomo y reinicie el *Centro de información de DB2* en su equipo.

**Nota:** En Windows 2008 y Windows Vista (y posterior), los mandatos listados más abajo deben ejecutarse como administrador. Para abrir un indicador de mandatos o una herramienta gráfica con privilegios de administrador completos, pulse con el botón derecho del ratón el atajo y, a continuación, seleccione **Ejecutar como administrador**.

## Procedimiento

Para actualizar el *Centro de información de DB2* instalado en el sistema o en el servidor de Intranet:

1. Detenga el *Centro de información de DB2*.
  - En Windows, pulse **Inicio > Panel de control > Herramientas administrativas > Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Detener**.
  - En Linux, especifique el mandato siguiente:  

```
/etc/init.d/db2icdv10 stop
```
2. Inicie el Centro de información en modalidad autónoma.
  - En Windows:
    - a. Abra una ventana de mandatos.
    - b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el *Centro de información de DB2* se instala en el directorio *Archivos\_de\_programa\IBM\DB2 Information Center\Versión 10.1*, siendo *Archivos\_de\_programa* la ubicación del directorio Archivos de programa.
    - c. Navegue desde el directorio de instalación al directorio `doc\bin`.
    - d. Ejecute el archivo `help_start.bat`:  

```
help_start.bat
```
  - En Linux:
    - a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el *Centro de información de DB2* se instala en el directorio `/opt/ibm/db2ic/V10.1`.
    - b. Navegue desde el directorio de instalación al directorio `doc/bin`.
    - c. Ejecute el script `help_start`:  

```
help_start
```

## Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Se abre el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.

3. Pulse en el botón **Actualizar** (🔄). (JavaScript debe estar habilitado en el navegador.) En la derecha del panel del Centro de información, pulse en **Buscar actualizaciones**. Se visualiza una lista de actualizaciones para la documentación existente.
4. Para iniciar el proceso de instalación, compruebe las selecciones que desee instalar y, a continuación, pulse **Instalar actualizaciones**.
5. Cuando finalice el proceso de instalación, pulse **Finalizar**.
6. Detenga el Centro de información autónomo:

- En Windows, navegue hasta el directorio `doc\bin` del directorio de instalación y ejecute el archivo `help_end.bat`:

```
help_end.bat
```

**Nota:** El archivo `help_end` de proceso por lotes contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el archivo `help_start` de proceso por lotes. No utilice `Control-C` ni ningún otro método para detener `help_start.bat`.

- En Linux, navegue hasta el directorio `doc/bin` del directorio de instalación y ejecute el script `help_end`:

```
help_end
```

**Nota:** El script `help_end` contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el script `help_start`. No utilice ningún otro método para detener el script `help_start`.

7. Reinicie el *Centro de información de DB2*.
  - En Windows, pulse **Inicio > Panel de control > Herramientas administrativas > Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Iniciar**.
  - En Linux, especifique el mandato siguiente:

```
/etc/init.d/db2icdv10 start
```

### Resultados

El *Centro de información de DB2* actualizado muestra los temas nuevos y actualizados.

---

## Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos de base de datos DB2. Se proporcionan instrucciones paso a paso a través de lecciones.

### Antes de comenzar

Puede ver la versión XHTML de la guía de aprendizaje desde el Centro de información en el sitio <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/>.

Algunas lecciones utilizan datos o código de ejemplo. Consulte la guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.



## Guías de aprendizaje de DB2

Para ver la guía de aprendizaje, pulse el título.

### “pureXML” en *pureXML Guide*

Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

## Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución y determinación de problemas para ayudarle en la utilización de productos de base de datos DB2.

### Documentación de DB2

Puede encontrar información sobre la resolución de problemas en la publicación *Troubleshooting and Tuning Database Performance* o en la sección sobre conceptos fundamentales sobre bases de datos del *Centro de información de DB2*, que contiene:

- Información sobre cómo aislar e identificar problemas con programas de utilidad y herramientas de diagnóstico de DB2.
- Soluciones a algunos de los problemas más comunes.
- Consejo para ayudarle a resolver problemas que podría encontrar en los productos de base de datos DB2

### Portal de Soporte de IBM

Consulte el portal de soporte de IBM si tiene problemas y desea obtener ayuda para encontrar las causas y soluciones posibles. El sitio de soporte técnico tiene enlaces a las publicaciones más recientes de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR o arreglos de defectos), fixpacks y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al portal de Soporte de IBM en el sitio [http://www.ibm.com/support/entry/portal/Overview/Software/Information\\_Management/DB2\\_for\\_Linux,\\_UNIX\\_and\\_Windows](http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows)

## Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

**Aplicación:** Además de las condiciones de uso del sitio web de IBM, se aplican estos términos y condiciones.

**Uso personal:** Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se mantengan los avisos sobre la propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

**Uso comercial:** Puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando se mantengan todos los avisos sobre la propiedad. No puede realizar trabajos derivados de estas publicaciones, ni reproducirlas, distribuir las o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

## Términos y condiciones

**Derechos:** Excepto lo expresamente concedido en este permiso, no se conceden otros permisos, licencias ni derechos, explícitos o implícitos, sobre las publicaciones ni sobre ninguna información, datos, software u otra propiedad intelectual contenida en el mismo.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando, a su discreción, el uso de las publicaciones sea en detrimento de su interés o cuando, según determine IBM, las instrucciones anteriores no se cumplan correctamente.

No puede descargar, exportar ni volver a exportar esta información excepto en el caso de cumplimiento total con todas las leyes y regulaciones vigentes, incluyendo todas las leyes y regulaciones sobre exportación de los Estados Unidos.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN "TAL CUAL" Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

**Marcas registradas de IBM:** IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Puede consultarse en línea una lista actualizada de las marcas registradas de IBM en la web en [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

---

## Apéndice B. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América. La información acerca de productos que no son IBM se basa en la información disponible cuando se publicó este documento por primera vez y está sujeta a cambio.

Es posible que IBM no comercialice en otros 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. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere 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 juegos de caracteres de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país o escribir a:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japón

**El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde 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, COMERCIALIZACIÓN 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

## Avisos

efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y 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 de esos sitios web no forma parte de la información del presente producto de IBM y la utilización de esos sitios web se realiza bajo la responsabilidad del usuario.

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 éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited  
U59/3600  
3600 Steeles Avenue East  
Markham, Ontario L3R 9Z7  
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 documento 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 lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado 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 ninguna 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.

Este manual 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 contiene 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 como 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 lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas. Los programas de ejemplo se proporcionan "TAL CUAL", sin ningún tipo de garantía. IBM no se hará responsable de los daños derivados de la utilización que haga el usuario de los programas de ejemplo.

Cada copia o parte 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 proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *\_entre el o los años\_*. Reservados todos los derechos.

#### **Marcas registradas**

IBM, el logotipo de IBM e [ibm.com](http://ibm.com) son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. La lista actual de marcas registradas de IBM está disponible en la web, en "Copyright and trademark information", en la dirección [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Los siguientes términos son marcas registradas de otras empresas.

- Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.
- Java y todos los logotipos y marcas registradas basadas en Java son marcas registradas de Oracle, sus filiales o ambos.
- UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.
- Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Celeron, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y en otros países.
- Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.



---

# Índice

## A

- acceso remoto
  - conexiones no satisfactorias 353
  - conexiones satisfactorias 353
  - sentencia CONNECT 353
- activadores
  - adición de comentarios al catálogo 312
  - CREATE TRIGGER, sentencia 849
  - descartar 1042
  - INSERT, sentencia 1173
  - mensajes de error 849
  - no operativos 225, 849
  - sentencia ALTER TRIGGER 225
  - tipo, tablas 849
  - UPDATE, sentencia 1415
- activadores no operativos 225, 849
- actualización posicional de columnas por fila 1415
- actualizaciones
  - Centro de información de DB2 1447, 1448
  - vistas actualizables 945
- agrupaciones de almacenamientos intermedios
  - creación 377
  - descartar 1042
  - establecimiento de tamaño 33, 377
  - tamaño de página 377
- alias
  - adición de comentarios al catálogo 312
  - CREATE ALIAS, sentencia 369
  - descartar 1042
- ALLOCATE CURSOR, sentencia 27
- almacenamiento en antememoria
  - EXECUTE, sentencia 1079
- ALTER AUDIT POLICY, sentencia 29
- ALTER BUFFERPOOL, sentencia 33
- ALTER DATABASE, sentencia
  - detalles 40
- ALTER DATABASE PARTITION GROUP, sentencia 36
- ALTER EVENT MONITOR, sentencia
  - detalles 45
- ALTER FUNCTION, sentencia 50
- ALTER HISTOGRAM TEMPLATE, sentencia 54
- ALTER INDEX, sentencia 56
- ALTER METHOD, sentencia 58
- ALTER NICKNAME, sentencia 69
- ALTER NODEGROUP, sentencia
  - véase ALTER DATABASE PARTITION GROUP, sentencia 36
- ALTER PACKAGE, sentencia 78
- ALTER PROCEDURE (derivado), sentencia 86
- ALTER PROCEDURE (externo), sentencia 83
- ALTER PROCEDURE (SQL), sentencia 88
- ALTER SCHEMA, sentencia 90
- ALTER SECURITY LABEL COMPONENT, sentencia 92
- ALTER SECURITY POLICY, sentencia 95
- ALTER SEQUENCE, sentencia 99
- ALTER SERVICE CLASS, sentencia 106
- ALTER TABLE, sentencia
  - detalles 122
- ALTER TABLESPACE, sentencia
  - detalles 194
- ALTER THRESHOLD, sentencia 211
- ALTER TRUSTED CONTEXT, sentencia 226
- ALTER TYPE (Estructurado), sentencia 235
- ALTER USAGE LIST, sentencia 242
- ALTER USER MAPPING, sentencia 244
- ALTER VIEW, sentencia
  - detalles 247
- ALTER WORK ACTION SET, sentencia 249
- ALTER WORK CLASS SET, sentencia 264
- ALTER WRAPPER, sentencia 287
- ALTER XSROBJECT, sentencia 289
- ámbito
  - adición con sentencia ALTER TABLE 122
  - adición con sentencia ALTER VIEW 247
  - CREATE VIEW, sentencia 945
  - definición con columnas añadidas 122
  - definición con la sentencia CREATE TABLE 722
- anotaciones cronológicas
  - creación de tablas sin anotaciones cronológicas iniciales 722
- aplicaciones de SQL incorporado
  - EXECUTE IMMEDIATE, sentencia 1087
  - sentencias de formato de serie de caracteres 1087
  - visión general 11
- apodos
  - detalles 631
  - privilegios
    - otorgar 1159
    - revocar 1276
- aritméticos
  - marcadores de parámetro 1214
- ASSOCIATE LOCATORS, sentencia 290
- ASUTIME
  - CREATE FUNCTION (escalar externa), sentencia 471
  - CREATE FUNCTION (tabla externa), sentencia 501
  - CREATE PROCEDURE (externo), sentencia 650
  - CREATE PROCEDURE (SQL), sentencia 673
- AUDIT, sentencia 292
- autorización SECADM (administrador de seguridad)
  - otorgar 1117
  - revocar 1240
- avisos 1453
- ayuda
  - sentencias SQL 1446

## B

- base de datos, autorizaciones
  - otorgar
    - GRANT (autorizaciones de bases de datos), sentencia 1117
- bases de datos
  - acceso
    - otorgamiento de autoridad 1117
    - CREATE TABLESPACE, sentencia 810
  - BEGIN DECLARE SECTION, sentencia 296
- BIGINT, tipo de datos
  - CREATE TABLE, sentencia 722
- BLOB, tipo de datos
  - CREATE TABLE, sentencia 722
- bloqueos
  - INSERT, sentencia 1173

bloqueos (*continuación*)  
  LOCK TABLE, sentencia 1189  
  restricción del acceso 1189  
  Sentencia COMMIT 323  
  terminación para unidad de trabajo 1285  
  UPDATE, sentencia 1415

## C

CALL, sentencia  
  detalles 298  
cargas  
  otorgar autorización de base de datos 1117  
CASCADE, norma de supresión 722  
catálogos  
  COMMENT, sentencia 312  
Centro de información de DB2  
  actualización 1447, 1448  
  versiones 1446  
CHAR VARYING, tipo de datos 722  
CHARACTER VARYING, tipo de datos 722  
cláusula SQLCA de la sentencia INCLUDE 1171  
cláusula SQLERROR de la sentencia WHENEVER 1437  
cláusula SQLWARNING de sentencia WHENEVER 1437  
claves de particionamiento  
  añadir 122  
  definición al crear tablas 722  
  descartar 122  
claves exclusivas  
  ALTER TABLE, sentencia 122  
  CREATE TABLE, sentencia 722  
claves foráneas  
  añadir 122  
  descartar 122  
  nombres de restricción 722  
claves primarias  
  añadir  
    ALTER TABLE, sentencia 122  
    CREATE TABLE, sentencia 722  
  descartar  
    ALTER TABLE, sentencia 122  
  privilegios necesarios 1159  
CLOB, tipo de datos  
  columnas 722  
CLOSE, sentencia  
  detalles 310  
códigos de retorno  
  sentencias de SQL ejecutables 11, 14  
  sentencias incorporadas 11, 14  
códigos de retorno de SQL 14  
COLLID  
  CREATE FUNCTION (escalar externa), sentencia 471  
  CREATE FUNCTION (tabla externa), sentencia 501  
  CREATE PROCEDURE (externo), sentencia 650  
  CREATE PROCEDURE (SQL), sentencia 673  
columnas  
  actualización 1415  
  adiciones de comentarios al catálogo 312  
  añadir  
    ALTER TABLE, sentencia 122  
  claves de índice 587  
  nombres  
    INSERT, sentencia 1173  
  otorgación de privilegios de adición 1159  
  restricciones  
    nombres 722  
  columnas (*continuación*)  
    valores  
      insertar 1173  
    valores nulos  
      ALTER TABLE, sentencia 122  
  columnas de identidad  
    CREATE TABLE, sentencia 722  
  columnas generadas  
    CREATE TABLE, sentencia 722  
  comentarios  
    estáticas, sentencias de SQL 15  
    SQL  
      sentencias estáticas 11  
      tabla de catálogo 312  
  Comentarios de SQL  
    compuestos 15  
    simples 15  
  COMMENT, sentencia 312  
  COMMIT, sentencia  
    detalles 323  
  compilación  
    condicional (SQL) 15  
  compilación condicional  
    SQL 15  
  condiciones de búsqueda  
    DELETE, sentencia 1018  
    UPDATE, sentencia 1415  
  condiciones de error x  
  conexiones implícitas  
    sentencia CONNECT 353  
  conjuntos de resultados  
    devolución  
      procedimientos SQL 335  
  CONNECT, sentencia  
    tipo 2 361  
  contenedores  
    CREATE TABLESPACE, sentencia 810  
  convenios  
    resaltar x  
  conversión  
    serie de caracteres a SQL ejecutable 1087  
  correlaciones de particiones  
    creación de grupos de particiones de base de datos 381  
  crear sentencia de SQL sentencia-de-selección  
    definición 13  
    invocación 11  
    invocación dinámica 12  
    invocación estática 13  
  CREATE ALIAS, sentencia 369  
  CREATE AUDIT POLICY, sentencia 373  
  CREATE BUFFERPOOL, sentencia 377  
  CREATE DATABASE PARTITION GROUP, sentencias 381  
  CREATE DISTINCT TYPE, sentencia  
    véase sentencia CREATE TYPE, tipo diferenciado 883  
  CREATE EVENT MONITOR (actividades), sentencia 406  
  CREATE EVENT MONITOR (bloqueo), sentencia 425  
  CREATE EVENT MONITOR (estadísticas), sentencia 438  
  CREATE EVENT MONITOR, sentencia 384  
  CREATE EVENT MONITOR (violaciones de umbral),  
    sentencia 451  
  CREATE FUNCTION, sentencia  
    derivadas 534  
    escalares de SQL 549  
    escalares externas 471  
    fila de SQL 549  
    OLE, tabla externa 523  
    plantilla 534



CREATE FUNCTION, sentencia (*continuación*)  
  tabla de SQL 549  
  tabla externa 501  
  visión general 470

CREATE FUNCTION MAPPING, sentencia 567

CREATE HISTOGRAM TEMPLATE, sentencia 585

CREATE INDEX, sentencia  
  detalles 587

CREATE INDEX EXTENSION, sentencia 609

CREATE METHOD, sentencia  
  detalles 623

CREATE MODULE, sentencia 629

CREATE NICKNAME, sentencia  
  detalles 631

CREATE PROCEDURE, sentencia  
  DECLARE, sentencia 335  
  derivadas 667  
  externas 650  
  FOR, sentencia 1107  
  GET DIAGNOSTICS, sentencia 1111  
  ITERATE, sentencia 1185  
  manejadores de condiciones 335  
  Sentencia CASE 307  
  sentencia de manejador de condiciones 335  
  Sentencia GOTO 1115  
  Sentencia IF 1169  
  Sentencia LEAVE 1187  
  Sentencia LOOP 1191  
  Sentencia REPEAT 1233  
  Sentencia RETURN 1238  
  Sentencia WHILE 1440  
  SIGNAL, sentencia 1397  
  SQL 673  
  SQL compuesto 335  
  una sentencia de SQL compuesto (en línea) 326  
  variables 335  
  visión general 649

CREATE ROLE, sentencia  
  detalles 684

CREATE SCHEMA, sentencia 685

CREATE SECURITY LABEL, sentencia 691

CREATE SECURITY LABEL COMPONENT, sentencia 688

CREATE SECURITY POLICY, sentencia 693

CREATE SEQUENCE, sentencia 695

CREATE SERVER, sentencia 713

CREATE SERVICE CLASS, sentencia 701

CREATE STOGROUP, sentencia  
  detalles 717

CREATE SYNONYM, sentencia 721

CREATE TABLE, sentencia  
  detalles 722

CREATE TABLESPACE, sentencia  
  detalles 810

CREATE THRESHOLD, sentencia  
  detalles 827

CREATE TRANSFORM, sentencia  
  detalles 845

CREATE TRIGGER, sentencia 849

CREATE TRUSTED CONTEXT, sentencia  
  detalles 865

CREATE TYPE, sentencia  
  detalles 872  
  tipo de fila 891  
  tipo de matriz 873  
  tipo diferenciado 883  
  tipo estructurado 896

CREATE USAGE LIST, sentencia 928

CREATE USER MAPPING, sentencia  
  detalles 932

CREATE VARIABLE, sentencia 935

CREATE VIEW, sentencia 945

CREATE WORK ACTION SET, sentencia 961

CREATE WORK CLASS SET, sentencia 971

CREATE WORKLOAD, sentencia  
  detalles 977

CURRENT DECFLOAT ROUNDING MODE, registro especial  
  SET CURRENT DECFLOAT ROUNDING MODE,  
  sentencia 1299

CURRENT DEGREE, registro especial  
  SET CURRENT DEGREE, sentencia 1303

CURRENT EXPLAIN MODE, registro especial  
  SET CURRENT EXPLAIN MODE, sentencia 1305

CURRENT EXPLAIN SNAPSHOT, registro especial  
  SET CURRENT EXPLAIN SNAPSHOT, sentencia 1308

CURRENT FUNCTION PATH, registro especial  
  SET CURRENT FUNCTION PATH, sentencia 1372  
  SET CURRENT PATH, sentencia 1372  
  SET PATH, sentencia 1372

CURRENT IMPLICIT XMLPARSE OPTION, registro especial  
  SET CURRENT IMPLICIT XMLPARSE OPTION,  
  sentencia 1313

CURRENT ISOLATION, registro especial  
  SET CURRENT ISOLATION, sentencia 1314

CURRENT OPTIMIZATION PROFILE, registro especial  
  sentencia SET CURRENT TEMPORAL  
  BUSINESS\_TIME 1341  
  sentencia SET CURRENT TEMPORAL  
  SYSTEM\_TIME 1343  
  SET CURRENT OPTIMIZATION PROFILE,  
  sentencia 1325

CURRENT PATH, registro especial  
  SET CURRENT FUNCTION PATH, sentencia 1372  
  SET CURRENT PATH, sentencia 1372  
  SET PATH, sentencia 1372

CURRENT QUERY OPTIMIZATION, registro especial  
  SET CURRENT QUERY OPTIMIZATION, sentencia 1334

CURRENT REFRESH AGE, registro especial  
  SET CURRENT REFRESH AGE, sentencia 1337

cursores  
  actualizables  
  determinación 998  
  ambiguo 998  
  apertura 1206  
  asociación de conjunto activo 1206  
  declarar  
  sintaxis de sentencia de SQL 998

DECLARE CURSOR, sentencia 998

estado cerrado 1206

fila actual 1095

movimiento de la posición mediante FETCH 1095

nombres  
  asignación 27

preparación para que lo utilice la aplicación 1206

relación con tabla de resultados 998

sólo lectura  
  condiciones 998  
  suprimir 1018

ubicación en la tabla como resultado de la sentencia  
  FETCH 1095

unidades de trabajo  
  estados condicionales 998  
  terminación para 1285

WITH HOLD  
  cláusula de bloqueo de la sentencia COMMIT 323

cursores ambiguos 998  
cursores de sólo lectura  
ambiguo 998

## D

datos  
integridad  
bloqueos 1189  
datos XML  
CREATE INDEX, sentencia 587  
db2nodes.cfg, archivo  
ALTER DATABASE PARTITION GROUP, sentencia 36  
CONNECT (tipo 1), sentencia 353  
CREATE DATABASE PARTITION GROUP, sentencias 381  
DBADM (administrador de la base de datos), autorización  
otorgar 1117  
DBCLOB, tipo de datos  
CREATE TABLE, sentencia 722  
declaraciones  
inserción en programa 1171  
DECLARE, sentencias  
BEGIN DECLARE SECTION, sentencia 296  
END DECLARE SECTION, sentencia 1078  
SQL compuesto 335  
DECLARE CURSOR, sentencia  
detalles 998  
DECLARE GLOBAL TEMPORARY TABLE, sentencia  
detalles 1004  
DELETE, sentencia  
detalles 1018  
DESCRIBE, sentencia  
detalles 1029  
sentencias preparadas  
DESCRIBE INPUT, sentencia 1030  
DESCRIBE OUTPUT, sentencia 1034  
DESCRIBE INPUT, sentencia 1030  
DESCRIBE OUTPUT, sentencia 1034  
determinación de problemas  
guías de aprendizaje 1451  
información disponible 1451  
devolución de conjuntos de resultados  
procedimientos SQL 335  
diagramas de sintaxis  
lectura viii  
DISCONNECT, sentencia 1039  
documentación  
archivos PDF 1444  
copia impresa 1444  
términos y condiciones de uso 1451  
visión general 1443  
DROP, sentencia  
detalles 1042  
transformaciones 1042

## E

elemento de sintaxis designador de función 22  
elemento de sintaxis designador de método 22  
elemento de sintaxis designador de procedimiento 22  
en desuso, funcionalidad  
sentencias de SQL  
ALTER DATABASE 40  
END DECLARE SECTION, sentencia 1078  
errores  
cursores 1206

errores (continuación)  
FETCH, sentencia 1095  
UPDATE, sentencia 1415  
espacio gestionado por base de datos (DMS)  
espacios de tablas  
CREATE TABLESPACE, sentencia 810  
espacio gestionado por el sistema (SMS)  
espacios de tablas  
CREATE TABLESPACE, sentencia 810  
espacios de tablas  
agrupaciones de almacenamientos intermedios 377  
añadir  
comentarios al catálogo 312  
creación  
CREATE TABLESPACE, sentencia 810  
descartar  
DROP, sentencia 1042  
identificación  
CREATE TABLE, sentencia 722  
índices  
CREATE TABLE, sentencia 722  
otorgar privilegios 1157  
renombrar 1231  
revocar privilegios 1274  
suprimir utilizando la sentencia DROP 1042  
tamaños de página 810  
esquemas  
adición de comentarios al catálogo 312  
CREATE SCHEMA, sentencia 685  
implícitos  
autorización, revocar 1240  
otorgamiento de autoridad 1117  
esquemas implícitos  
GRANT (autorizaciones de bases de datos), sentencia 1117  
sentencia REVOKE (autorizaciones de base de datos) 1240  
estado cerrado  
cursores 1206  
estado pendiente de establecer integridad  
SET INTEGRITY, sentencia 1350  
estándares  
establecer normas para SQL dinámico 1375  
estructura SQLCA  
visión general 14  
estructuras de almacenamiento  
ALTER BUFFERPOOL, sentencia 33  
ALTER TABLESPACE, sentencia 194  
CREATE BUFFERPOOL, sentencia 377  
CREATE TABLESPACE, sentencia 810  
etiquetas  
procedimientos SQL 20  
Sentencia GOTO 1115  
etiquetas de seguridad (LBAC)  
ALTER SECURITY LABEL COMPONENT, sentencia 92  
CREATE SECURITY LABEL, sentencia 691  
CREATE SECURITY LABEL COMPONENT, sentencia 688  
políticas  
ALTER SECURITY POLICY, sentencia 95  
CREATE SECURITY POLICY, sentencia 693  
sentencia GRANT (etiqueta de seguridad) 1147  
sentencia REVOKE (etiqueta de seguridad) 1265  
EXCLUSIVE MODE, conexión 353  
EXECUTE, sentencia  
detalles 1079  
incorporado 11, 12  
EXECUTE IMMEDIATE, sentencia  
detalles 1087  
incorporado 11, 12

EXPLAIN, sentencia  
detalles 1090

## F

FETCH, sentencia  
detalles 1095  
requisitos previos del cursor para la ejecución 1095  
filas  
actualización  
valores de columna utilizando la sentencia  
UPDATE 1415  
asignación de valores a variables del lenguaje principal  
SELECT INTO, sentencia 1292  
VALUES INTO, sentencia 1434  
bloqueos  
efecto en el cursor de WITH HOLD 998  
INSERT, sentencia 1173  
claves de índice con cláusula UNIQUE 587  
cursores  
efecto de cierre en una sentencia FETCH 310  
FETCH, sentencia 1206  
ubicación en las tablas de resultados 998  
índices 587  
insertar  
INSERT, sentencia 1173  
otorgar privilegios 1159  
petición FETCH 998  
restricciones que conducen a anomalías 1173  
suprimir  
DELETE, sentencia 1018  
FLOAT, tipo de datos  
CREATE TABLE, sentencia 722  
FLUSH BUFFERPOOLS, sentencia 1100  
FLUSH EVENT MONITOR, sentencia 1101  
FLUSH FEDERATED CACHE, sentencia 1102  
FLUSH PACKAGE CACHE, sentencia 1106  
FOR, sentencia 1107  
FREE LOCATOR, sentencia 1110  
FROM, cláusula  
DELETE, sentencia 1018  
funciones  
adición de comentarios al catálogo 312  
plantillas  
detalles 567  
transformación 845  
funciones de transformación  
CREATE TRANSFORM 845

## G

generación aleatoria en claves de particionamiento 722  
GET DIAGNOSTICS, sentencia 1111  
GRANT, sentencia  
apodo, privilegios 1159  
base de datos, autorizaciones 1117  
etiquetas de seguridad 1147  
exenciones 1123  
privilegio SETSESSIONUSER 1155  
privilegios de carga de trabajo 1166  
privilegios de espacio de tablas 1157  
privilegios de esquema 1144  
privilegios de índice 1129  
privilegios de objeto XSR 1168  
privilegios de paquete 1133  
privilegios de rutina 1139

GRANT, sentencia (*continuación*)  
privilegios de secuencia 1150  
privilegios de servidor 1153  
privilegios de tabla 1159  
privilegios de variable global 1126  
roles 1136  
vista, privilegios 1159  
GRAPHIC, tipo de datos  
CREATE TABLE, sentencia 722  
grupos de particiones de base de datos  
adición de comentarios al catálogo 312  
adición de particiones 36  
creación 381  
creación de correlación de distribución 381  
descartar particiones 36  
guías de aprendizaje  
determinación de problemas 1451  
lista 1450  
pureXML 1450  
resolución de problemas 1451

## I

ID de autorización  
control público en índices 1129  
otorgar control  
índices 1129  
operaciones de base de datos 1117  
otorgar privilegios de esquema 1144  
revocar autorizaciones 1240  
identificador de juego de caracteres codificados (CCSID)  
CREATE TABLE, sentencia 722  
DECLARE GLOBAL TEMPORARY TABLE,  
sentencia 1004  
identificadores de objeto (OID)  
columnas  
visión general 722  
CREATE TABLE, sentencia 722  
CREATE VIEW, sentencia 945  
INCLUDE, sentencia  
detalles 1171  
índice sobre datos XML  
CREATE INDEX, sentencia  
detalles 587  
índices  
clave exclusiva 122  
clave primaria 122  
comentarios de especificación de catálogo 312  
correspondencia con los valores de fila insertados 1173  
descartar 1042  
nombres  
restricción de clave primaria 722  
restricción de unicidad 722  
otorgar control 1129, 1159  
privilegios  
revocar 1249  
renombrar 1228  
índices de tipo 2 587  
INSERT, sentencia 1173  
INTEGER, tipo de datos  
CREATE TABLE, sentencia 722  
ITERATE, sentencia  
detalles 1185

## L

- LBAC
  - ALTER SECURITY LABEL COMPONENT, sentencia 92
  - ALTER SECURITY POLICY, sentencia 95
  - componentes de etiqueta de seguridad
    - ALTER SECURITY LABEL COMPONENT, sentencia 92
    - CREATE SECURITY LABEL COMPONENT, sentencia 688
  - CREATE SECURITY LABEL, sentencia 691
  - CREATE SECURITY LABEL COMPONENT, sentencia 688
  - CREATE SECURITY POLICY, sentencia 693
  - etiquetas de seguridad
    - ALTER SECURITY LABEL COMPONENT, sentencia 92
    - CREATE SECURITY LABEL, sentencia 691
    - CREATE SECURITY LABEL COMPONENT, sentencia 688
    - sentencia GRANT (etiqueta de seguridad) 1147
    - sentencia REVOKE (etiqueta de seguridad) 1265
  - exenciones de normas
    - sentencia GRANT (exención) 1123
    - sentencia REVOKE (exención) 1244
  - políticas de seguridad
    - ALTER SECURITY POLICY, sentencia 95
    - CREATE SECURITY POLICY, sentencia 693
    - sentencia GRANT (etiqueta de seguridad) 1147
    - sentencia GRANT (exención) 1123
    - sentencia REVOKE (etiqueta de seguridad) 1265
    - sentencia REVOKE (exención) 1244
- lenguaje REXX
  - prohibición END DECLARE SECTION 1078
- listas de uso
  - suprimir utilizando la sentencia DROP 1042
- localizadores
  - ASSOCIATE LOCATORS, sentencia 290
  - FREE LOCATOR, sentencia 1110
- LOCK TABLE, sentencia
  - detalles 1189

## M

- manejadores de condiciones
  - declarar 335
- marcadores de parámetro
  - con tipo 1214
  - EXECUTE, sentencia 1079
  - normas de contraseña 1214
  - OPEN, sentencia 1206
  - PREPARE, sentencia 1214
  - sin tipo 1214
- máscaras
  - sentencia ALTER MASK 57
  - sentencia CREATE MASK 616
- mensajes de error
  - activadores
    - ejecución 849
    - tipo, tablas 225
  - códigos de retorno 11, 14
  - máscaras de columna 57, 616
  - permisos de fila 82, 645
- MERGE, sentencia 1193
- MODE, palabra clave 1189
- módulos
  - creación 629
  - modificar 60

## MQT

- definición 722
- REFRESH TABLE, sentencia 1221

## N

- niveles de aislamiento
  - DELETE, sentencia 1018
  - INSERT, sentencia 1173
  - SELECT, sentencia 1292
  - UPDATE, sentencia 1415
- NO ACTION, norma de supresión 722
- nombres de condición de SQL
  - referencias 20
- nombres de cursor de SQL
  - referencias 21
- nombres de sentencias de SQL
  - referencias 21
- norma de supresión RESTRICT 722
- NOT FOUND, cláusula
  - WHENEVER, sentencia 1437

## O

- objetos dependientes
  - DROP, sentencia 1042
- objetos grandes binarios (BLOB)
  - tablas 722
- OID
  - véase identificadores de objeto (OID) 722
- opción PROGRAM para DB2 para compatibilidad con z/OS
  - DROP, sentencia 1042
- OPEN, sentencia
  - detalles 1206

## P

- paquetes
  - ALTER TABLE, sentencia 122
  - autorización para crear 1117
  - comentarios de catálogo 312
  - COMMIT, efecto de la sentencia en los cursores 323
  - privilegios
    - otorgar 1133
    - revocación utilizando la sentencia REVOKE (privilegios de paquete) 1253
    - revocación utilizando la sentencia REVOKE (privilegios de tabla, vista o apodo) 1276
  - suprimir 1042
- permisos
  - sentencia ALTER PERMISSION 82
  - sentencia CREATE PERMISSION 645
- PIPE, sentencia 1212
- precisión simple, tipo de datos de coma flotante 722
- precompilación
  - archivos de texto externos 1171
  - INCLUDE, sentencia 1171
  - sentencias de SQL no ejecutables 11
  - SQLCA 1171
  - SQLDA 1171
- PREPARE, sentencia
  - declarar dinámicamente 1214
  - detalles 1214
  - incorporado 11, 12
  - sustitución de variable en sentencia OPEN 1206

- privilegio SETSESSIONUSER
  - necesario para la sentencia SET SESSION
    - AUTHORIZATION 1379
  - sentencia GRANT (privilegio SETSESSIONUSER) 1155
  - sentencia REVOKE (privilegio SETSESSIONUSER) 1272
- privilegios
  - bases de datos
    - revocar 1263
  - índices
    - revocar 1249
  - paquetes
    - revocar 1253, 1276
  - revocar
    - REVOKE, sentencia 1276
- procedimientos
  - autorización para crear
    - CREATE PROCEDURE (externo), sentencia 650
    - CREATE PROCEDURE (SQL), sentencia 673
  - CALL, sentencia 298
  - creación 650, 673
  - CREATE PROCEDURE, sentencia 649
- procedimientos SQL
  - DECLARE, sentencia 326, 335
  - FOR, sentencia 1107
  - GET DIAGNOSTICS, sentencia 1111
  - ITERATE, sentencia 1185
  - manejadores de condiciones
    - declarar 335
  - Sentencia CASE 307
  - sentencia compuesta compilada 335
  - sentencia de SQL compuesto (en línea) 326
  - Sentencia GOTO 1115
  - Sentencia IF 1169
  - Sentencia LEAVE 1187
  - Sentencia LOOP 1191
  - Sentencia REPEAT 1233
  - Sentencia RETURN 1238
  - Sentencia WHILE 1440
  - SIGNAL, sentencia 1397
  - variables 326, 335
- PROGRAM TYPE
  - CREATE FUNCTION (escalar externa), sentencia 471
  - CREATE FUNCTION (tabla externa), sentencia 501
- PUBLIC AT ALL LOCATIONS 1159
- puntos de salvaguarda
  - liberar 1227
  - sentencia ROLLBACK con cláusula TO SAVEPOINT 1285

## R

- REAL SQL, tipo de datos
  - CREATE TABLE, sentencia 722
- Referencias
  - etiquetas 20
  - nombres de condición de SQL 20
  - nombres de cursor de SQL 21
  - nombres de sentencias de SQL 21
- REFRESH TABLE, sentencia 1221
- registros
  - bloqueos en datos de fila 1173
- RELEASE (conexión), sentencia 1225
- RELEASE SAVEPOINT, sentencia 1227
- RENAME, sentencia 1228
- RENAME STOGROUP, sentencia
  - detalles 1230
- RENAME TABLESPACE, sentencia 1231

- rendimiento
  - recomendación de clave de particionamiento 722
- resolución de problemas
  - guías de aprendizaje 1451
  - información en línea 1451
- restricciones
  - adición con sentencia ALTER TABLE 122
  - adición de comentarios al catálogo 312
  - descartar 122
- restricciones de comprobación
  - ALTER TABLE, sentencia 122
  - CREATE TABLE, sentencia 722
  - INSERT, sentencia 1173
- restricciones de integridad 312
- restricciones de referencia
  - adición de comentarios al catálogo 312
- restricciones de unicidad
  - ALTER TABLE, sentencia 122
- añadir
  - ALTER TABLE, sentencia 122
  - CREATE TABLE, sentencia 722
  - descartar con ALTER TABLE 122
- RESULTSTATUS, parámetro 1111
- REVOKE, sentencia
  - apodo, privilegios 1276
  - base de datos, autorizaciones 1240
  - etiquetas de seguridad 1265
  - exenciones 1244
  - privilegio SETSESSIONUSER 1272
  - privilegios de carga de trabajo 1282
  - privilegios de espacio de tablas 1274
  - privilegios de esquema 1263
  - privilegios de índice 1249
  - privilegios de módulo 1251
  - privilegios de objeto XSR 1284
  - privilegios de paquete 1253
  - privilegios de rutina 1259
  - privilegios de secuencia 1267
  - privilegios de servidor 1270
  - privilegios de tabla 1276
  - privilegios de variable global 1246
  - roles 1256
  - vista, privilegios 1276
- ROLLBACK, sentencia
  - detalles 1285
- ROLLBACK TO SAVEPOINT, sentencia 1285

## S

- SAVEPOINT, sentencia 1288
- secuencias
  - DROP, sentencia 1042
- seguridad
  - sentencia CONNECT 353
- selección completa
  - CREATE VIEW, sentencia 945
- selección completa de fila
  - UPDATE, sentencia 1415
- SELECT, sentencia
  - cursores 998
  - detalles 1291
  - evaluación para la tabla de resultados del cursor de
    - sentencia OPEN 1206
- SELECT INTO, sentencia
  - detalles 1292
  - sentencia ALTER MASK 57
  - sentencia ALTER PERMISSION 82

- sentencia ALTER SERVER 103
- sentencia ALTER STOGROUP
  - detalles 117
- sentencia ALTER TRIGGER 225
- sentencia ALTER WORKLOAD
  - detalles 270
- Sentencia CASE
  - detalles 307
- sentencia compuesta compilada
  - detalles 335
- sentencia CONNECT
  - tipo 1 353
- Sentencia CREATE EVENT MONITOR (antememoria de paquete) 431
- sentencia CREATE EVENT MONITOR (historial de cambios) 418
- Sentencia CREATE EVENT MONITOR (unidad de trabajo) 464
- sentencia CREATE GLOBAL TEMPORARY TABLE
  - detalles 572
- sentencia CREATE MASK 616
- sentencia CREATE NODEGROUP 381
- sentencia CREATE PERMISSION 645
- sentencia CREATE TYPE MAPPING
  - detalles 921
- sentencia CREATE WRAPPER
  - detalles 996
- Sentencia FLUSH OPTIMIZATION PROFILE CACHE 1104
- Sentencia GOTO
  - detalles 1115
- Sentencia IF
  - SQL 1169
- Sentencia LEAVE
  - detalles 1187
- Sentencia LOOP
  - SQL 1191
- Sentencia REPEAT
  - detalles 1233
- Sentencia RESIGNAL 1235
- Sentencia RETURN
  - detalles 1238
- sentencia SET 1385
- sentencia SET CURRENT LOCALE LC\_MESSAGES 1315
- sentencia SET CURRENT TEMPORAL BUSINESS\_TIME 1341
- sentencia SET CURRENT TEMPORAL SYSTEM\_TIME 1343
- Sentencia SET SESSION AUTHORIZATION 1379
- Sentencia WHILE
  - detalles 1440
- sentencias
  - LEAVE 1187
- sentencias de SQL
  - ALLOCATE CURSOR 27
  - ALTER AUDIT POLICY 29
  - ALTER BUFFERPOOL 33
  - ALTER DATABASE 40
  - ALTER DATABASE PARTITION GROUP 36
  - ALTER FUNCTION 50
  - ALTER HISTOGRAM TEMPLATE 54
  - ALTER INDEX 56
  - ALTER MASK 57
  - ALTER METHOD 58
  - ALTER MODULE 60
  - ALTER NICKNAME 69
  - ALTER NODEGROUP (véase sentencias de SQL, ALTER DATABASE PARTITION GROUP) 36
  - ALTER PACKAGE 78
  - ALTER PERMISSION 82
- sentencias de SQL (*continuación*)
  - ALTER PROCEDURE (derivado) 86
  - ALTER PROCEDURE (externo) 83
  - ALTER PROCEDURE (SQL) 88
  - ALTER SCHEMA 90
  - ALTER SECURITY LABEL COMPONENT 92
  - ALTER SECURITY POLICY 95
  - ALTER SEQUENCE 99
  - ALTER SERVER 103
  - ALTER SERVICE CLASS 106
  - ALTER STOGROUP 117
  - ALTER TABLE 122
  - ALTER TABLESPACE 194
  - ALTER THRESHOLD 211
  - ALTER TRIGGER 225
  - ALTER TRUSTED CONTEXT 226
  - ALTER TYPE (estructurado) 235
  - ALTER USER MAPPING 244
  - ALTER VIEW 247
  - ALTER WORK ACTION SET 249
  - ALTER WORK CLASS SET 264
  - ALTER WORKLOAD 270
  - ALTER WRAPPER 287
  - ALTER XSROBJECT 289
  - ASSOCIATE LOCATORS 290
  - atributo de cursor WITH HOLD 998
  - AUDIT 292
  - BEGIN DECLARE SECTION 296
  - CALL 298
  - CLOSE 310
  - COMMENT 312
  - COMMIT 323
  - compuesto (incorporado) 331
  - CONNECT
    - tipo 1 353
    - tipo 2 361
  - CONTINUE 1437
  - control 18
  - CREATE ALIAS 369
  - CREATE AUDIT POLICY 373
  - CREATE BUFFERPOOL 377
  - CREATE DATABASE PARTITION GROUP 381
  - CREATE EVENT MONITOR 384
  - CREATE EVENT MONITOR (actividades) 406
  - CREATE EVENT MONITOR (antememoria de paquete) 431
  - CREATE EVENT MONITOR (estadísticas) 438
  - CREATE EVENT MONITOR (historial de cambios) 418
  - CREATE EVENT MONITOR (violaciones de umbral) 451
  - CREATE FUNCTION
    - derivadas 534
    - escalares de SQL 549
    - escalares externas 471
    - fila de SQL 549
    - plantilla 534
    - tabla de SQL 549
    - tabla externa 501
    - tabla externa OLE DB 523
    - visión general 470
  - CREATE FUNCTION MAPPING 567
  - CREATE GLOBAL TEMPORARY TABLE 572
  - CREATE HISTOGRAM TEMPLATE 585
  - CREATE INDEX 587
  - CREATE INDEX EXTENSION 609
  - CREATE MASK 616
  - CREATE METHOD 623
  - CREATE MODULE 629

sentencias de SQL (continuación)

CREATE NICKNAME 631  
 CREATE NODEGROUP (véase sentencias de SQL, CREATE DATABASE PARTITION GROUP) 381  
 CREATE PERMISSION 645  
 CREATE PROCEDURE  
   derivadas 667  
   externas 650  
   SQL 673  
   visión general 649  
 CREATE ROLE 684  
 CREATE SCHEMA 685  
 CREATE SECURITY LABEL 691  
 CREATE SECURITY LABEL COMPONENT 688  
 CREATE SECURITY POLICY 693  
 CREATE SEQUENCE 695  
 CREATE SERVER 713  
 CREATE SERVICE CLASS 701  
 CREATE STOGROUP 717  
 CREATE TABLE 722  
 CREATE TABLESPACE 810  
 CREATE THRESHOLD 827  
 CREATE TRANSFORM 845  
 CREATE TRIGGER 849  
 CREATE TRUSTED CONTEXT 865  
 CREATE TYPE 872  
   diferenciado 883  
   estructurado 896  
   fila 891  
   matriz 873  
 CREATE TYPE MAPPING 921  
 CREATE USER MAPPING 932  
 CREATE VARIABLE 935  
 CREATE VIEW 945  
 CREATE WORK ACTION SET 961  
 CREATE WORK CLASS SET 971  
 CREATE WORKLOAD 977  
 CREATE WRAPPER 996  
 DECLARE CURSOR 998  
 DECLARE GLOBAL TEMPORARY TABLE 1004  
 DELETE 1018  
 DESCRIBE 1029  
 DESCRIBE INPUT 1030  
 DESCRIBE OUTPUT 1034  
 DISCONNECT 1039  
 DROP 1042  
 DROP TRANSFORM 1042  
 END DECLARE SECTION 1078  
 entrada interactiva 11, 13  
 EXECUTE 1079  
 EXECUTE IMMEDIATE 1087  
 EXPLAIN 1090  
 FETCH 1095  
 FLUSH BUFFERPOOLS 1100  
 FLUSH EVENT MONITOR 1101  
 FLUSH OPTIMIZATION PROFILE CACHE 1104  
 FLUSH PACKAGE CACHE 1106  
 FREE LOCATOR 1110  
 GRANT  
   apodo, privilegios 1159  
   base de datos, autorizaciones 1117  
   etiqueta de seguridad 1147  
   exención 1123  
   privilegio SETSESSIONUSER 1155  
   privilegios de carga de trabajo 1166  
   privilegios de espacio de tablas 1157  
   privilegios de esquema 1144

sentencias de SQL (continuación)

GRANT (continuación)  
   privilegios de índice 1129  
   privilegios de módulo 1131  
   privilegios de objeto XSR 1168  
   privilegios de paquete 1133  
   privilegios de rutina 1139  
   privilegios de secuencia 1150  
   privilegios de servidor 1153  
   privilegios de tabla 1159  
   privilegios de variable global 1126  
   role 1136  
   vista, privilegios 1159  
 INCLUDE 1171  
 incorporado 11  
 INSERT 1173  
 invocación 11  
 LOCK TABLE 1189  
 MERGE 1193  
 OPEN 1206  
 PREPARE 1214  
 REFRESH TABLE 1221  
 RELEASE (conexión) 1225  
 RELEASE SAVEPOINT 1227  
 RENAME 1228  
 RENAME STOGROUP 1230  
 RENAME TABLESPACE 1231  
 RESIGNAL 1235  
 REVOKE  
   apodo, privilegios 1276  
   base de datos, autorizaciones 1240  
   etiqueta de seguridad 1265  
   exención 1244  
   privilegio SETSESSIONUSER 1272  
   privilegios de carga de trabajo 1282  
   privilegios de espacio de tablas 1274  
   privilegios de esquema 1263  
   privilegios de índice 1249  
   privilegios de objeto XSR 1284  
   privilegios de paquete 1253  
   privilegios de rutina 1259  
   privilegios de secuencia 1267  
   privilegios de servidor 1270  
   privilegios de tabla 1276  
   privilegios de variable global 1246  
   role 1256  
   vista, privilegios 1276  
 ROLLBACK 1285  
 ROLLBACK TO SAVEPOINT 1285  
 SAVEPOINT 1288  
 SELECT 1291  
 SELECT INTO 1292  
 series  
   creación 1087  
   PREPARE, sentencia 1214  
 SET COMPILATION ENVIRONMENT 1296  
 SET CONNECTION 1297  
 SET CONSTRAINTS 1350  
 SET CURRENT DECFLOAT ROUNDING MODE 1299  
 SET CURRENT DEFAULT TRANSFORM GROUP 1301  
 SET CURRENT DEGREE 1303  
 SET CURRENT EXPLAIN MODE 1305  
 SET CURRENT EXPLAIN SNAPSHOT 1308  
 SET CURRENT FEDERATED ASYNCHRONY 1311  
 SET CURRENT FUNCTION PATH 1372  
 SET CURRENT IMPLICIT XMLPARSE OPTION 1313  
 SET CURRENT ISOLATION 1314

sentencias de SQL (*continuación*)

- SET CURRENT LOCALE LC\_MESSAGES 1315
- SET CURRENT LOCK TIMEOUT 1319
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 1321
- SET CURRENT MDC ROLLOUT MODE 1323
- SET CURRENT OPTIMIZATION PROFILE 1325
- SET CURRENT PACKAGE PATH 1328
- SET CURRENT PACKAGESET 1332
- SET CURRENT PATH 1372
- SET CURRENT QUERY OPTIMIZATION 1334
- SET CURRENT REFRESH AGE 1337
- SET CURRENT SQL\_CCFLAGS 1339
- SET ENCRYPTION PASSWORD 1345
- SET EVENT MONITOR STATE 1347
- SET INTEGRITY 1350
- SET PASSTHRU 1370
- SET PATH 1372
- SET ROLE 1374
- SET SCHEMA 1375
- SET SERVER OPTION 1377
- SET SESSION AUTHORIZATION 1379
- SET variable 1385
- SQL compuesto 325
- TRANSFER OWNERSHIP 1400
- TRUNCATE 1412
- UPDATE 1415
- VALUES 1433
- VALUES INTO 1434
- WHENEVER 1437

Sentencias de SQL

- visión general 1

sentencias de SQL activadas

- SET variable 1385

sentencias de SQL compuesto

- en línea 326
- incorporado 331
- visión general 325

sentencias de SQL ejecutables 11, 12, 13

sentencias de SQL no ejecutables

- invocación 11
- requisitos del precompilador 11

sentencias de SQL preparadas

- ejecución 1079
- obtención de información
  - DESCRIBE INPUT, sentencia 1030
  - DESCRIBE OUTPUT, sentencia 1034
- sustitución de variable del lenguaje principal 1079

sentencias SQL

- ayuda
  - visualización 1446

Sentencias SQL

- ALTER EVENT MONITOR 45
- ALTER USAGE LIST 242
- CREATE USAGE LIST 928
- FLUSH FEDERATED CACHE 1102
- PIPE 1212
- SET CURRENT TEMPORAL BUSINESS\_TIME 1341
- SET CURRENT TEMPORAL SYSTEM\_TIME 1343
- SET USAGE LIST STATE 1382

series de caracteres

- creación de sentencias de SQL 1087

servidores

- otorgar privilegios 1153

SET COMPILATION ENVIRONMENT, sentencia 1296

SET CONNECTION, sentencia 1297

SET CONSTRAINTS, sentencia 1350

SET CURRENT DECFLOAT ROUNDING MODE, sentencia 1299

SET CURRENT DEFAULT TRANSFORM GROUP, sentencia 1301

SET CURRENT DEGREE, sentencia 1303

SET CURRENT EXPLAIN MODE, sentencia 1305

SET CURRENT EXPLAIN SNAPSHOT, sentencia 1308

SET CURRENT FEDERATED ASYNCHRONY, sentencia 1311

SET CURRENT FUNCTION PATH, sentencia 1372

SET CURRENT IMPLICIT XMLPARSE OPTION, sentencia 1313

SET CURRENT ISOLATION, sentencia 1314

SET CURRENT LOCALE LC\_TIME, sentencia 1317

SET CURRENT LOCK TIMEOUT, sentencia 1319

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION, sentencia 1321

SET CURRENT MDC ROLLOUT MODE, sentencia 1323

SET CURRENT OPTIMIZATION PROFILE, sentencia 1325

SET CURRENT PACKAGE PATH, sentencia 1328

SET CURRENT PACKAGESET, sentencia 1332

SET CURRENT PATH, sentencia 1372

SET CURRENT QUERY OPTIMIZATION, sentencia detalles 1334

SET CURRENT REFRESH AGE, sentencia 1337

SET CURRENT SQL\_CCFLAGS, sentencia 1339

SET CURRENT SQLID, sentencia 1375

SET ENCRYPTION PASSWORD, sentencia detalles 1345

SET EVENT MONITOR STATE, sentencia 1347

SET INTEGRITY, sentencia detalles 1350

SET NULL, norma de supresión 722

SET PASSTHRU, sentencia detalles 1370

- independencia de la sentencia COMMIT 323
- independencia de la sentencia ROLLBACK 1285

SET PATH, sentencia 1372

SET ROLE, sentencia 1374

SET SCHEMA, sentencia 1375

SET SERVER OPTION, sentencia detalles 1377

- independencia de la sentencia COMMIT 323
- independencia de la sentencia ROLLBACK 1285

SET USAGE LIST STATE, sentencia 1382

SHARE MODE, conexión 353

SIGNAL, sentencia 1397

signo de interrogación

- como marcador de parámetro EXECUTE 1079

simultaneidad

- LOCK TABLE, sentencia 1189

sinónimos

- CREATE ALIAS, sentencia 369
- DROP ALIAS, sentencia 1042

SMALLINT, tipo de datos

- SQL estático 722

SQL

- códigos de retorno 11
- objetos
  - suprimir 1042
- parámetros 19
- variables
  - referencias 19
  - sentencia de SQL compuesto (compilado) 335
  - sentencia de SQL compuesto (en línea) 326

SQL dinámico

- cursores
  - DECLARE CURSOR, sentencia 11, 12



SQL dinámico (*continuación*)  
 DESCRIBE INPUT, sentencia 1030  
 DESCRIBE OUTPUT, sentencia 1034  
 EXECUTE, sentencia  
   detalles 1079  
   invocación de sentencias de SQL 11, 12  
 EXECUTE IMMEDIATE, sentencia  
   detalles 1087  
 FETCH, sentencia  
   detalles 1095  
   invocación de sentencias de SQL 11, 12  
 invocación de sentencias 11, 12  
 OPEN, sentencia 11, 12  
 PREPARE, sentencia  
   detalles 1214  
   invocación de sentencias de SQL 11, 12  
   utilizando DESCRIBE 1030, 1034  
 sentencias compuestas 326

SQL estático  
 DECLARE CURSOR, sentencia 11, 13  
 FETCH, sentencia 11  
 invocación 11, 13  
 OPEN, sentencia 11  
 seleccionar 13  
 sentencia-select 11  
 sentencias 11, 13

SQL92, estándar  
 SQL dinámico 1375

SQLCA  
 UPDATE, sentencia 1415  
 visión general 11

SQLCODE  
 descripción 14  
 detalles 11

SQLDA  
 detalles de variable del lenguaje principal 1206  
 FETCH, sentencia 1095  
 OPEN, sentencia 1206  
 variables obligatorias para la sentencia DESCRIBE  
   INPUT 1030  
 variables obligatorias para la sentencia DESCRIBE  
   OUTPUT 1034

SQLSTATE  
 descripción 14  
 detalles 11

STAY RESIDENT  
 CREATE FUNCTION (escalar externa), sentencia 471  
 CREATE FUNCTION (tabla externa), sentencia 501  
 CREATE PROCEDURE, sentencia 650, 673

supervisores de sucesos  
 CREATE EVENT MONITOR, sentencia 384  
 DROP, sentencia 1042  
 FLUSH EVENT MONITOR, sentencia 1101  
 SET EVENT MONITOR STATE, sentencia 1347

## T

### tablas

actualización por fila y columna 1415  
 adición de columnas 122  
 adición de comentarios al catálogo 312  
 alias 369, 1042  
 autorización para crear 722  
 columnas generadas 122  
 con tipo  
   activadores 849

### tablas (*continuación*)

creación  
   otorgamiento de autoridad 1117  
   sentencia de SQL, instrucciones 722  
 esquemas 685  
 excepciones 1350  
 índices 587  
 insertar filas 1173  
 modificar  
   ALTER TABLE, sentencia 122  
 nombres  
   ALTER TABLE, sentencia 122  
   CREATE TABLE, sentencia 722  
   LOCK TABLE, sentencia 1189  
 otorgar privilegios 1159  
 renombrar 1228  
 restricción del acceso compartido 1189  
 revocar privilegios 1276  
 suprimir utilizando la sentencia DROP 1042  
 temporales  
   OPEN, sentencia 1206  
 uniones  
   CREATE TABLE, sentencia 722

tablas de excepciones  
 SET INTEGRITY, sentencia 1350

tablas de resumen  
 visión general 722

tablas temporales  
 OPEN, sentencia 1206

terminación  
 unidades de trabajo 323, 1285

términos y condiciones  
 publicaciones 1451

TIME, tipos de datos  
 CREATE TABLE, sentencia 722

TIMESTAMP, tipo de datos  
 CREATE TABLE, sentencia 722

tipo de datos CHARACTER 722

tipo de datos DB2SECURITYLABEL  
 CREATE TABLE, sentencia 722

tipos de datos  
 abstracto 235, 896  
 ALTER TYPE, sentencia 235  
 CREATE TYPE (estructurado), sentencia 896  
 declarados 335  
 definidas por el usuario  
   CREATE TYPE (diferenciado), sentencia 883  
 diferenciado  
   CREATE TYPE (diferenciado), sentencia 883  
 estructurado  
   ALTER TYPE (estructurado), sentencia 235  
   CREATE TYPE (estructurado), sentencia 896

tipos de datos de fila  
 sentencia CREATE TYPE (cursor) 880

tipos diferenciados  
 CREATE TYPE (diferenciado), sentencia 883  
 DROP, sentencia 1042

tipos estructurados  
 CREATE TRANSFORM, sentencia 845  
 DROP, sentencia 1042

TRANSFER OWNERSHIP, sentencia 1400

transformaciones  
 DROP, sentencia 1042

TRUNCATE, sentencia  
 detalles 1412

## U

### UDF

- CREATE FUNCTION, sentencia
  - derivadas 534
  - escalar, tabla o fila de SQL 549
  - escalares externas 471
  - plantilla 534
  - tabla externa 501
  - tabla externa OLE DB 523
  - visión general 470
- DROP, sentencia 1042
- sentencia REVOKE (autorizaciones de base de datos) 1240

### UDT

- adición de comentarios al catálogo 312
- CREATE TRANSFORM, sentencia 845
- CREATE TYPE (diferenciado), sentencia 883
- tipos diferenciados
  - CREATE TABLE, sentencia 722
- tipos estructurados 722

### unidades de trabajo

- cancelar 1285
- COMMIT, sentencia 323
- destrucción de sentencias preparadas 1214
- iniciación cierra cursores 1206
- referencia a sentencias preparadas 1214
- ROLLBACK, sentencia 1285
- terminación
  - confirmaciones 323
  - destrucción de sentencias preparadas 1214
  - sin guardar cambios 1285

### uniones

- CREATE TABLE, sentencia 722

### UPDATE, sentencia

- detalles 1415

## V

valores de clave de detención 609

valores de clave de inicio 609

### VALUES, cláusula

- carga de una fila 1173
- normas para número de valores 1173

VALUES, sentencia 1433

VALUES INTO, sentencia 1434

### VARCHAR, tipo de datos

- CREATE TABLE, sentencia 722

### variables del lenguaje principal

- aplicaciones REXX 296
- asignación de valores de una fila
  - SELECT INTO, sentencia 1292
  - VALUES INTO, sentencia 1434
- BEGIN DECLARE SECTION, sentencia 296
- declarar
  - BEGIN DECLARE SECTION, sentencia 296
  - cursores 998
  - END DECLARE SECTION, sentencia 1078
- END DECLARE SECTION, sentencia 1078
- enlazar conjunto activo con cursor 1206
- EXECUTE IMMEDIATE, sentencia 1087
- FETCH, sentencia 1095
- inserción en filas 1173
- sentencias de SQL incorporadas 11, 14
- series de sentencia 1214
- sustitución de marcador de parámetro 1079

### variables del lenguaje principal de la aplicación

- Assembler 1087

variables globales

- referencias 19

### VARIANT

- CREATE TYPE (estructurado), sentencia 896

### vinculación

- GRANT, sentencia 1133
- revocar privilegio BIND 1253

### vistas

- actualizables 945
  - actualización de filas por columnas 1415
  - adición de comentarios al catálogo 312
  - alias 369, 1042
  - creación 945
  - esquemas 685
  - insertable 945
  - insertar filas 1173
  - no operativos 945
  - nombres 247
  - nombres de columna 945
  - otorgar privilegios 1159
  - prevención de la pérdida de definición de vista con WITH
    - CHECK OPTION 1415
    - Privilegio CONTROL 1159
  - revocar privilegios 1276
  - sólo lectura 945
  - suprimible 945
  - suprimir utilizando la sentencia DROP 1042
  - WITH CHECK OPTION 1415
- ### vistas con tipo
- definir subvistas 945
- ### vistas de sólo lectura
- creación 945
- ### vistas insertables
- creación 945
- ### vistas no operativas
- 945
- ### vistas suprimibles
- visión general 945

## W

### WHENEVER, sentencia

- cambio del flujo de control 11
- detalles 1437

### WHERE, cláusula

- DELETE, sentencia 1018
- UPDATE, sentencia 1415

## X

### XML

- CREATE INDEX, sentencia 587





SC11-8071-01



Spine information:

IBM DB2 10.1 para Linux, UNIX y Windows

Consulta de SQL - Volumen 2

