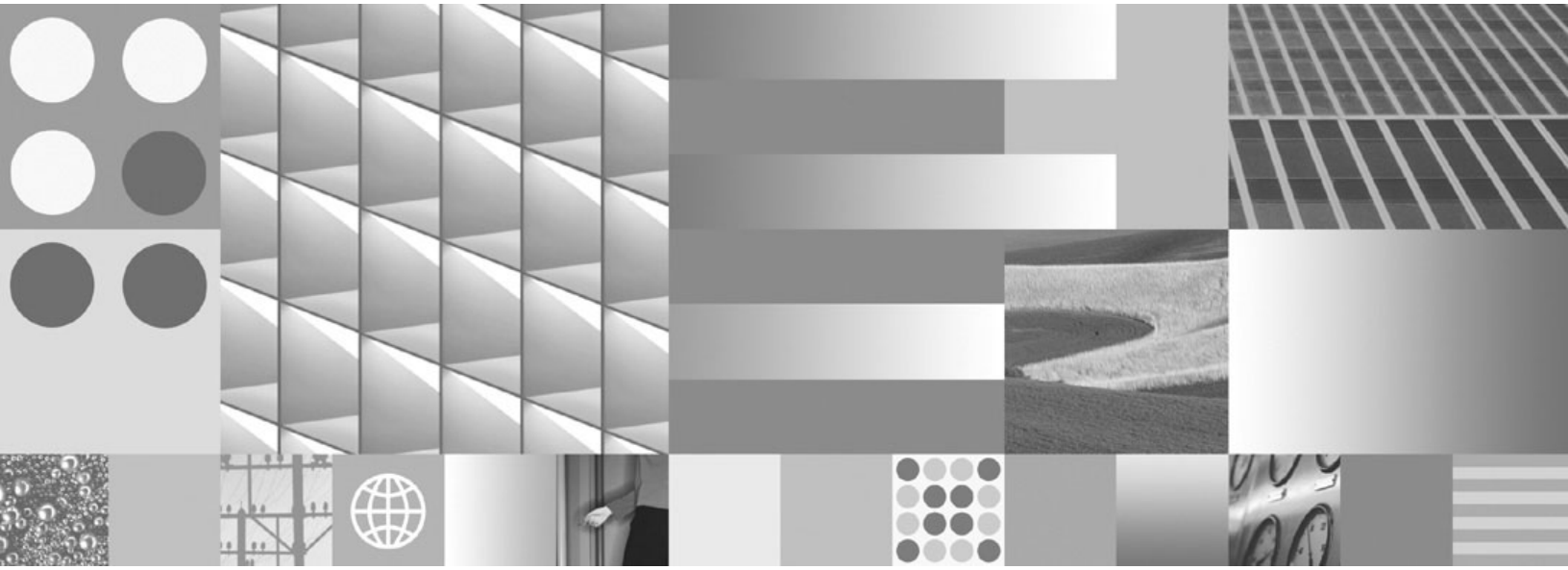
IBM DB2 9.7
para Linux, UNIX y Windows

Versión 9 Release 7

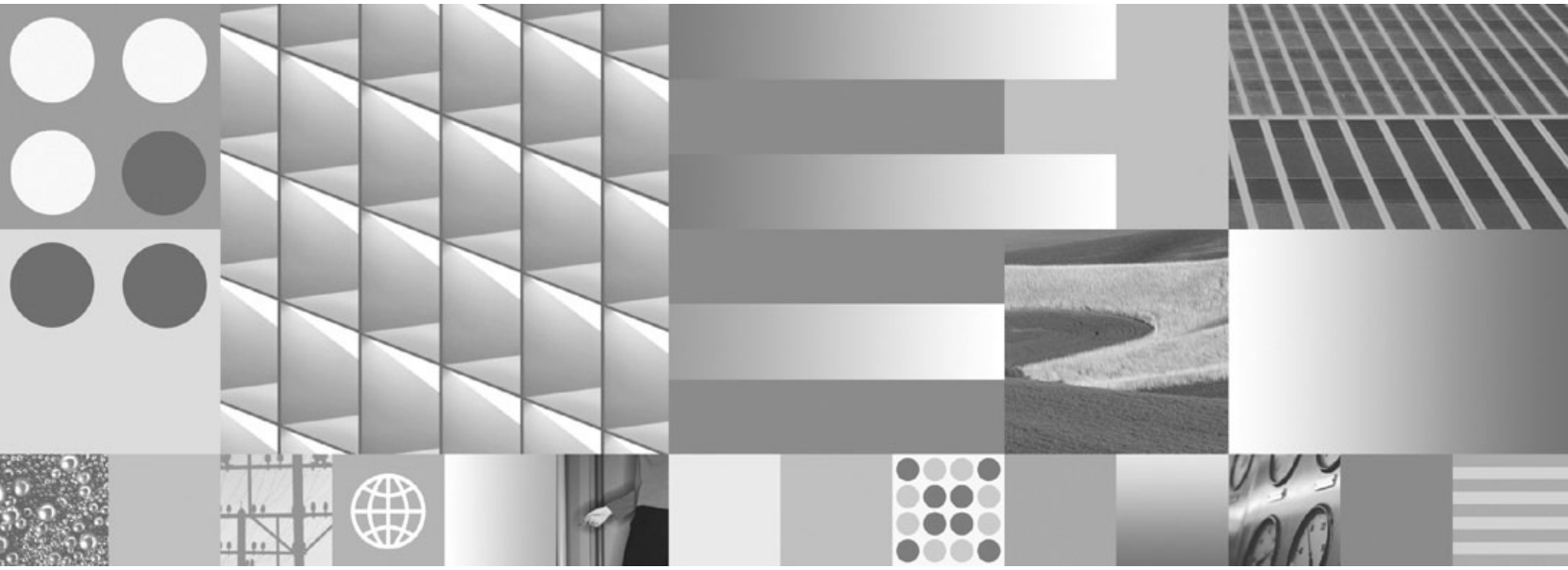


Consulta de SQL, Volumen 1
Actualizado en noviembre de 2009

SC11-3910-01

IBM DB2 9.7
para Linux, UNIX y Windows

Versión 9 Release 7



Consulta de SQL, Volumen 1
Actualizado en noviembre de 2009

SC11-3910-01

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice O, "Avisos", en la página 1175.

Nota de edición

Este manual es la traducción del manual en inglés *IBM DB2 9.7 for Linux, UNIX, and Windows Version 9 Release 7 SQL Reference, Volume 1* (SC27-2456-01).

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

- Para realizar pedidos en línea, vaya a IBM Publications Center ubicado en el sitio web www.ibm.com/shop/publications/order
- Para encontrar al representante de IBM de su localidad, vaya al IBM Directory of Worldwide Contacts en el sitio web 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, otorga a IBM un derecho no exclusivo a utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Contenido

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®. Éste 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 primer volumen del manual Consulta de SQL contiene información sobre los conceptos de las bases de datos relacionales, los elementos del lenguaje, las funciones y los formatos de las consultas. Los capítulos y los apéndices específicos de dicho volumen se describen brevemente aquí.

- El apartado "Conceptos" explica los conceptos básicos de las bases de datos relacionales y del SQL.
- "Elementos del lenguaje" describe la sintaxis básica del SQL y los elementos del lenguaje que son comunes a muchas sentencias de SQL.
- "Funciones" contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de utilización de funciones escalares y agregadas de SQL.
- "Procedimientos" contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de utilización de procedimientos.
- "Consultas de SQL" describe los distintos formatos de una consulta.
- "Límites de SQL y XML" lista las limitaciones del SQL.
- "SQLCA (área de comunicaciones de SQL" describe la estructura de SQLCA.
- "SQLDA (área de descriptor de SQL" describe la estructura de SQLDA.
- "Vistas de catálogos del sistema" describe las vistas de catálogos del sistema.
- "Sistemas federados" describe las opciones y las correlaciones de tipos para sistemas federados.
- "Vistas de catálogos del sistema" presenta la base de datos SAMPLE, que contiene las tablas que se utilizan en muchos ejemplos.
- "Nombres de esquema reservados y palabras reservadas" contiene los nombres de esquemas reservados y las palabras reservadas para los estándares SQL de IBM® y SQL99 y SQL2003 de ISO/ANSI.
- "Ejemplos de interacción entre activadores y restricciones referenciales" describe la interacción de los activadores y las restricciones de referencia.

- "Tablas Explain" describe las tablas Explain.
- "Valores de registro Explain" describe la interacción que tienen entre sí los valores de registro especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT y con los mandatos PREP y BIND.
- "Tablas de excepciones" contiene información sobre las tablas creadas por el usuario que se utilizan con la sentencia SET INTEGRITY.
- "Sentencias SQL permitidas en las rutinas" lista las sentencias de SQL que se permite ejecutar en rutinas con diferentes contextos de acceso de datos de SQL.
- "CALL invocada desde una sentencia compilada" describe la sentencia CALL que se puede invocar desde una sentencia compilada.

Cómo leer los diagramas de sintaxis

La sintaxis se describe con la estructura definida de la forma siguiente:

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.

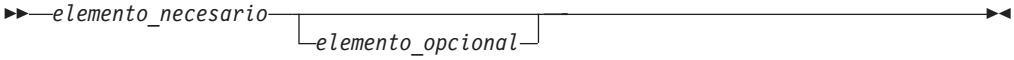
El símbolo ?--- indica que la sintaxis continúa de la línea anterior.

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

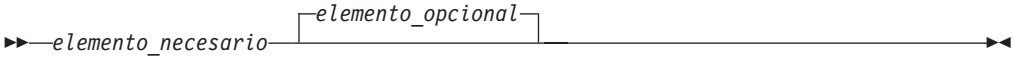
Los fragmentos de sintaxis empiezan con el símbolo +--- y finalizan con el símbolo ---¦.

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

Los elementos opcionales aparecen bajo la vía de acceso principal.

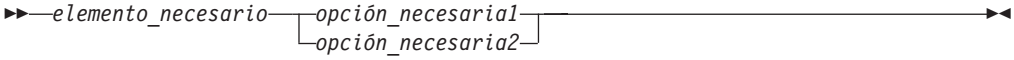


Si un elemento opcional aparece sobre la vía de acceso principal, ese elemento no tiene ningún efecto en la ejecución y sólo se utiliza para posibilitar la lectura.

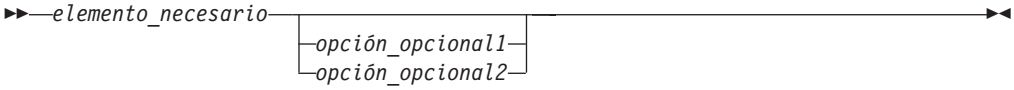


Si puede elegir entre dos o más elementos, éstos aparecen en una pila.

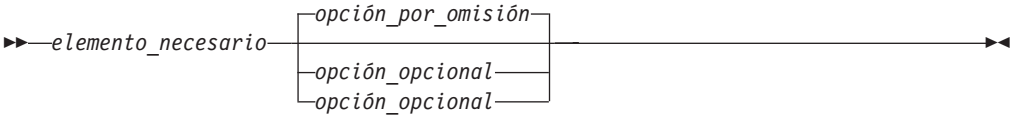
Si *debe* elegir uno de los elementos, un elemento de la pila aparece en la vía de acceso principal.



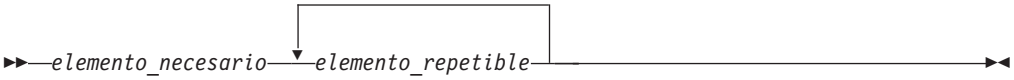
Si la elección de uno de los elementos es opcional, la pila entera aparece bajo la vía de acceso principal.



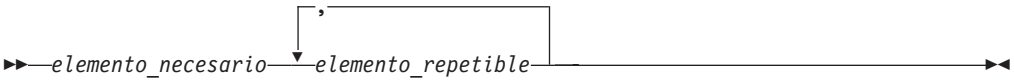
Si uno de los elementos es el valor por omisión, aparecerá sobre la vía de acceso principal y las opciones restantes se mostrarán debajo.



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.

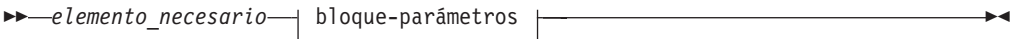


Una flecha de repetición sobre una pila indica que puede realizar más una elección en los elementos apilados o repetir una sola elección.

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
```

Convenciones utilizadas 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

Se utilizan los siguientes convenios en este manual.

Negrita	Indica mandatos, palabras clave y otros elementos cuyos nombres ha predefinido el sistema.
Cursiva	Indica uno de los 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

Convenciones para la descripción de datos Unicode

Cuando se hace referencia a un elemento de código de Unicode específico, éste se expresa como U+*n* donde *n* consta de un número entre cuatro y seis dígitos hexadecimales y donde se utilizan los dígitos 0 a 9 y las letras en mayúsculas de la A a la F. Los ceros iniciales se omiten a menos que el elemento de código tenga menos de cuatro dígitos hexadecimales. El carácter de espacio, por ejemplo, se expresa como U+0020. En la mayoría de los casos, el valor *n* es igual al de la codificación UTF-16BE.

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 los API de programación soportados.
- *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 escribir 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 SQL, tal como aparece en la norma ISO 2003.

Capítulo 1. Conceptos

Bases de datos

Una base de datos DB2 es una *base de datos relacional*. La *base de datos* almacena todos los datos en tablas que se relacionan entre ellas. Las relaciones se establecen entre tablas de modo que los datos se comparten y la duplicación se minimiza.

Una *base de datos relacional* es una base de datos que se trata como un conjunto de tablas y se manipula de acuerdo con el modelo de datos relacional. Contiene un conjunto de objetos que se utilizan para almacenar y gestionar los datos, así como para acceder a los mismos. Las tablas, vistas, índices, funciones, activadores y paquetes son ejemplos de estos objetos. Los objetos pueden ser definidos por el sistema (objetos definidos por el sistema) o por el usuario (objetos definidos por el usuario).

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

Una *base de datos relacional particionada* es una base de datos relacional cuyos datos se gestionan entre varias particiones de base de datos. Esta separación de datos en distintas particiones de base de datos es transparente para la mayoría de las sentencias de SQL. No obstante, algunas sentencias DDL (lenguaje de definición de datos) toman en consideración la información de partición de base de datos (por ejemplo, CREATE DATABASE PARTITION GROUP). El DDL es el subconjunto de sentencias de SQL que se utilizan para describir las relaciones de los datos de una base de datos.

Una *base de datos federada* es una base de datos relacional cuyos datos se almacenan en varias fuentes de datos (como, por ejemplo, bases de datos relacionales separadas). Los datos son tratados como si pertenecieran a una sola gran base de datos y se pueden acceder mediante las consultas SQL normales. Los cambios en los datos se pueden dirigir explícitamente hacia la fuente datos apropiada.

Lenguaje de consulta estructurada (SQL)

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

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

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

Consultas y expresiones de tabla

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

Una *expresión de tabla* crea una tabla de resultados temporal a partir de una consulta sencilla. Las cláusulas definen mejor la tabla de resultados. Por ejemplo, puede utilizar una expresión de tabla como una consulta para seleccionar todos los directores de varios departamentos, que deben tener una experiencia laboral de más de 15 años y que deben estar ubicados en la oficina de Nueva York.

Una *expresión de tabla común* es como una vista temporal dentro de una consulta compleja. Se puede hacer referencia a la misma en otros puntos de la consulta y se puede utilizar en lugar de una vista. Cada uso de una expresión de tabla común dentro de una consulta compleja comparte la misma vista temporal.

El uso recurrente de una expresión de tabla común dentro de una consulta se puede utilizar para dar soporte a aplicaciones como sistemas de reservar de líneas aéreas, generadores de material de envío (BOM) y planificación de red.

Introducción a DB2 Call Level Interface y ODBC

DB2 Call Level Interface (CLI de DB2) es una interfaz de SQL llamable de IBM para la familia DB2 de servidores de base de datos. Es una interfaz de programación de aplicaciones 'C' y 'C++' para el acceso a bases de datos relacionales que utiliza llamadas de función para pasar sentencias de SQL dinámico como argumentos de función.

Puede utilizar la interfaz CLI de DB2 para acceder a las bases de datos de servidor de datos de IBM siguientes:

- DB2 Versión 9 para Linux, UNIX y Windows
- DB2 Universal Database (DB2 UDB) Versión 8 (y posteriores) para Linux, UNIX y Windows
- DB2 Universal Database Versión 8 (y posteriores) para OS/390 y z/OS
- DB2 para IBM i 5.4 y posteriores

La CLI de DB2 es una alternativa al SQL dinámico incorporado pero, a diferencia del SQL incorporado, no requiere variables del lenguaje principal ni un precompilador. Las aplicaciones pueden ejecutarse en distintas bases de datos sin necesidad de volver a compilarlas en cada una de estas bases de datos. Las aplicaciones utilizan llamadas a procedimientos en tiempo de ejecución para conectarse a bases de datos, emitir sentencias de SQL y recuperar datos e información de estado.

La interfaz CLI de DB2 proporciona muchas características que no están disponibles en SQL interno. Por ejemplo:

- La CLI proporciona llamadas de función que soportan una forma de consultar catálogos de bases de datos que es coherente en toda la familia DB2. Esto reduce la necesidad de escribir consultas de catálogo que deban adaptarse a servidores de bases de datos concretos.

- La CLI proporciona la capacidad de desplazarse con un cursor de estas maneras:
  - Hacia adelante, una o más filas
  - Hacia atrás, una o más filas
  - Hacia adelante desde la primera fila, una o más filas
  - Hacia atrás desde la última fila, una o más filas
  - Desde una posición en el cursor almacenada previamente.
- Los procedimientos almacenados llamados desde programas de aplicación que se hayan escrito con la CLI pueden devolver conjuntos resultantes a esos programas.

La CLI de DB2 está basada en la especificación Microsoft® Open Database Connectivity (ODBC) y en la Norma internacional para SQL/CLI. Estas especificaciones se escogieron como la base para la DB2 Call Level Interface en un intento por seguir las normas de la industria, y para proporcionar una curva de aprendizaje más corta para aquellos programadores de aplicaciones que ya están familiarizados con cualquiera de estas interfaces de bases de datos. También se han añadido algunas extensiones específicas de DB2 para ayudar a los programadores de aplicaciones a explotar determinadas funciones de DB2.

El controlador de CLI de DB2 también actúa como un controlador ODBC cuando se carga mediante un gestor de controladores ODBC. Se ajusta a ODBC 3.51.

Información de fondo de la CLI de DB2

Para entender la CLI de DB2 o cualquier interfaz SQL llamable, es útil entender en qué se basa y compararla con las interfaces existentes.

X/Open Company y SQL Access Group desarrollaron conjuntamente una especificación para una interfaz SQL llamable denominada *X/Open Call Level Interface*. El objetivo de esta interfaz es aumentar la portabilidad de las aplicaciones, permitiéndoles ser independientes de cualquier interfaz de programación del proveedor de bases de datos. La mayor parte de la especificación X/Open Call Level Interface se ha aceptado como parte de la Norma Internacional ISO (ISO/IEC 9075-3:1995 SQL/CLI).

Microsoft desarrolló una interfaz SQL llamable denominada Open Database Connectivity (ODBC) para los sistemas operativos de Microsoft basada en un borrador preliminar de la CLI de X/Open.

La especificación ODBC también incluye un entorno operativo, en el que los controladores ODBC específicos de la base de datos se cargan dinámicamente en el tiempo de ejecución, mediante un gestor de controladores basado en la fuente de datos (nombre de la base de datos) proporcionada en la petición de conexión. La aplicación se enlaza directamente con una única biblioteca del gestor de controladores, en lugar de con la biblioteca de cada DBMS. El gestor de controladores media entre las llamadas de función de la aplicación durante el tiempo de ejecución y asegura que éstas se remitan al controlador ODBC adecuado que sea específico para el DBMS. Dado que el gestor de controladores ODBC sólo conoce las funciones específicas de ODBC, no se podrá acceder a las funciones específicas de DBMS en un entorno ODBC. Las sentencias de SQL dinámico específicas de DBMS están soportadas mediante un mecanismo denominado cláusula de escape.

ODBC no está limitado a los sistemas operativos de Microsoft; otras implementaciones están disponibles en varias plataformas.

Un gestor de controladores ODBC puede cargar la biblioteca de carga de CLI de DB2 como un controlador ODBC. Para el desarrollo de aplicaciones ODBC, debe obtener un kit de desarrollo de software de ODBC. Para la plataforma Windows, el SDK de ODBC se encuentra disponible como parte del SDK de Microsoft Data Access Components (MDAC), disponible para su descarga en http://www.microsoft.com/data/. Para las plataformas que no sean Windows, el SDK de ODBC lo proporcionan otros proveedores. Al desarrollar aplicaciones ODBC que puedan conectarse a servidores DB2, utilice la publicación Call Level Interface Guide and Reference, Volume 1 y la publicación Call Level Interface Guide and Reference, Volume 2 (para obtener información sobre información de diagnóstico y extensiones específicas de DB2), junto con la publicación ODBC Programmer's Reference and SDK Guide, disponible de Microsoft.

Las aplicaciones grabadas directamente en la CLI de DB2 vinculan directamente con la librería de carga de la CLI de DB2. La CLI de DB2 incluye soporte para muchas funciones ODBC y SQL/CLI de ISO, así como para funciones específicas de DB2.

Las siguientes funciones de DB2 están disponibles tanto para aplicaciones ODBC como para aplicaciones de la CLI de DB2:

- tipos de datos de doble byte (gráfico)
- procedimientos almacenados
- unidad de trabajo distribuida (DUOW), confirmación en dos fases
- SQL compuesto
- tipos definidos por el usuario (UDT)
- funciones definidas por el usuario (UDF)

Desarrollo de aplicaciones Java para servidores de datos de IBM

Los sistemas de base de datos DB2 e IBM Informix Dynamic Server (IDS) proporcionan soporte de controlador para aplicaciones cliente y applets escritos en Java™.

Puede acceder a datos en sistemas de base de datos DB2 e IDS utilizando JDBC, SQL o pureQuery.

JDBC

JDBC es una interfaz de programación de aplicaciones (API) que las aplicaciones de Java utilizan para acceder a las bases de datos relacionales. El soporte de IBM Data Server para JDBC le permite escribir aplicaciones Java que acceden a datos locales de DB2 o datos relacionales remotos de IDS en un servidor que admite DRDA.

SQLJ

SQLJ proporciona soporte para SQL estático incorporado en aplicaciones Java. IBM, Oracle y Tandem desarrollaron inicialmente SQLJ, para complementar al modelo JDBC de SQL dinámico con un modelo de SQL estático.

Para las conexiones a DB2, en general, las aplicaciones Java utilizan JDBC para el SQL dinámico y SQLJ para el SQL estático.

Para las conexiones con IDS, las sentencias de SQL contenidas en aplicaciones JDBC o SQLJ se ejecutan dinámicamente.

Debido a que SQLJ puede interactuar con JDBC, un programa de aplicación puede utilizar JDBC y SQLJ dentro de la misma unidad de trabajo.

pureQuery

pureQuery es una plataforma de acceso a datos de alto rendimiento que facilita el desarrollo, la optimización, la protección y la gestión del acceso a los datos. Se compone de:

- Interfaces de programación de aplicaciones creadas para facilitar el uso y simplificar el uso de las recomendaciones.
- Las herramientas de desarrollo, que se entregan en IBM Optim Development, para el desarrollo de Java y SQL.
- Un tiempo de ejecución, que se proporciona en IBM Optim pureQuery Runtime, para optimizar y proteger el acceso a la base de datos y simplificar las tareas de gestión.

Con pureQuery, puede escribir aplicaciones Java que tratan los datos relacionales como objetos, tanto si esos datos se encuentran en bases de datos como en objetos JDBC DataSource. Sus aplicaciones también pueden tratar los objetos almacenados en colecciones Java en memoria como si dichos objetos fueran datos relacionales. Para consultar o actualizar los datos relacionales o los objetos Java, utilice SQL.

Para obtener más información sobre pureQuery, consulte el Centro de información de Integrated Data Management.

Esquemas

Un *esquema* es un conjunto de objetos con nombre que proporciona una forma de agrupar los objetos lógicamente. Un esquema también es un calificador de nombres que proporciona una forma de utilizar el mismo nombre natural para varios objetos y de evitar referencias ambiguas a dichos objetos.

Por ejemplo, los nombres de esquema 'INTERNAL' y 'EXTERNAL' facilitan la distinción de dos tablas SALES diferentes (INTERNAL.SALES, EXTERNAL.SALES).

Además, los esquemas permiten que varias aplicaciones almacenen datos en una sola base de datos sin encontrarse con colisiones de espacios de nombres.

Un esquema es distinto de un *esquema XML* , y no debe confundirse con éste, que es un estándar que describe la estructura y valida el contenido de documentos XML.

Un esquema puede contener tablas, vistas, apodos, activadores, funciones, paquetes y otros objetos. Un esquema es un objeto de base de datos. Se crea explícitamente utilizando la sentencia CREATE SCHEMA con el usuario actual o un ID de autorización especificado registrado como propietario del esquema. También se puede crear implícitamente cuando se crea otro objeto, si el usuario tiene la autorización IMPLICIT_SCHEMA.

Un *nombre de esquema* se utiliza como la parte más a la izquierda de las dos partes del nombre de objeto. Si el objeto se califica específicamente con un nombre de esquema al crearse, se asigna el objeto a dicho esquema. Si no se especifica ningún nombre de esquema al crear el objeto, se utiliza el nombre de esquema por omisión (especificado en el registro especial CURRENT SCHEMA).

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

```
CREATE SCHEMA C AUTHORIZATION A
```

El usuario A puede emitir la siguiente sentencia para crear una tabla llamada X en el esquema C (siempre que el usuario A cuente con la autorización CREATETAB sobre la base de datos:

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

Algunos nombres de esquema están reservados. Por ejemplo, las funciones incorporadas pertenecen al esquema SYSIBM y las funciones preinstaladas definidas por el usuario pertenecen al esquema SYSFUN.

Cuando se crea una base de datos, si no se crea con la opción RESTRICTIVE, todos los usuarios tienen la autorización IMPLICIT_SCHEMA. Con esta autorización, los usuarios crean implícitamente un esquema cada vez que crean un objeto con un nombre de esquema que todavía no existe. Cuando se crean esquemas implícitamente, se otorgan privilegios CREATEIN que permiten a cualquier usuario crear otros objetos en este esquema. La posibilidad de crear objetos tales como alias, tipos diferenciados, funciones y activadores se amplía a los esquemas creados implícitamente. Los privilegios por omisión de un esquema creado implícitamente proporcionan compatibilidad con las versiones anteriores.

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

Los esquemas también tienen privilegios, los cuales permiten al propietario del esquema controlar qué usuarios tienen el privilegio de crear, modificar, copiar y eliminar objetos en el esquema. Esto proporciona una forma de controlar la manipulación de un subconjunto de objetos de la base de datos. A un propietario de esquema se le dan inicialmente todos estos privilegios en el esquema, con la posibilidad de otorgarlos a otros usuarios. Un esquema creado implícitamente es de propiedad del sistema y a todos los usuarios se les proporciona inicialmente el privilegio de crear objetos en dicho esquema. Un usuario con autorización ACCESSCTRL o SECADM puede cambiar los privilegios que poseen los usuarios en cualquier esquema. Por lo tanto, se puede controlar el acceso para crear, modificar, copiar y eliminar objetos en cualquier esquema (incluso uno creado implícitamente).

Tablas

Las tablas son estructuras lógicas mantenidas por el gestor de bases de datos. Las tablas están formadas por columnas y filas.

En la intersección de cada columna con una fila hay un elemento de datos específico denominado *valor*. Una *columna* es un conjunto de valores del mismo tipo o de uno de sus subtipos. Una *fila* es una secuencia de valores ordenados de forma que el valor *n* sea el valor de la columna *n* de la tabla.

Un programa de aplicación puede determinar el orden en que se llenan las filas de la tabla, pero el orden real de las filas lo determina el gestor de bases de datos y, por lo general, no se puede controlar. La agrupación en clúster de múltiples dimensiones (MDC) proporciona una visión de la agrupación en clúster, pero no la ordenación real entre las filas.

Tipos de tablas

Las bases de datos DB2 almacenan los datos en tablas. Además de las tablas utilizadas para almacenar datos persistentes, existen también algunas tablas que se utilizan para presentar resultados, tablas de resumen y tablas temporales; las tablas de clúster multidimensional ofrecen ventajas específicas para entornos de depósito, mientras que las tablas particionadas permiten distribuir datos en más de una partición de base de datos.

Tablas base

Las tablas de este tipo contienen datos persistentes. Existen diferentes clases de tablas base, algunas de las cuales se especifican a continuación

Tablas normales

Las tablas normales con índices son la opción de tablas ″de finalidad general″.

Tablas de clúster multidimensional (MDC)

Las tablas de este tipo se implementan como tablas que están agrupadas físicamente en clústeres en más de una clave, o dimensión, al mismo tiempo. Las tablas MDC se utilizan en entornos de base de datos grandes y depósito de datos. Los índices de clústeres en tablas normales dan soporte a la agrupación de los datos en clústeres de una única dimensión. Las tablas MDC ofrecen las ventajas de agrupar en clústeres de datos en más de una dimensión. Las tablas MDC permiten realizar la *agrupación en clústeres garantizada* dentro de las dimensiones compuestas. En contraposición, aunque se puede disponer de un índice en clúster con tablas normales, en este caso es el gestor de bases de datos quien intenta la agrupación, que no está garantizada y que generalmente se degrada con el paso del tiempo. Las MDC pueden coexistir con tablas particionadas y pueden serlo ellas mismas.

Tablas agrupadas por rangos de clústeres (RCT)

Las tablas de este tipo se implementan como clústeres secuenciales de datos que permiten el acceso rápido y directo. Cada registro de la tabla cuenta con un ID de registro predeterminado (RID) que es un identificador interno utilizado para ubicar un registro en una tabla. Las tablas RCT se utilizan cuando los datos están estrechamente agrupados en clústeres en una o varias columnas de la tabla. Los valores mayor y menor de las columnas definen el rango de valores posibles. Puede utilizar estas columnas para acceder a los registros de la tabla; este método es el más adecuado para utilizar el aspecto de identificador de registro predeterminado (RID) de las tablas RCT.

Tablas temporales

Las tablas de este tipo se utilizan como tablas de trabajo temporales para

diversas operaciones de base de datos. Las *tablas temporales declaradas* (DGTT) no aparecen en el catálogo del sistema; por este motivo, no pueden compartirse con otras aplicaciones y se convierten en no persistentes si las utilizan otras aplicaciones. Cuando la aplicación que utiliza esta tabla termina o se desconecta de la base de datos, se borran los datos de la tabla y ésta se descarta. En cambio, las *tablas temporales creadas* (CGTT) aparecen en el catálogo del sistema y no es necesario definirlas en cada sesión en la que se utilizan. Como resultado, son persistentes y pueden compartirse con otras aplicaciones en diferentes conexiones.

Ninguno de los dos tipos de tabla temporal da soporte a lo siguiente

- Columnas de tipo estructurado definidas por el usuario o de referencia definidas por el usuario
- Columnas LONG VARCHAR

Además no se pueden utilizar columnas XML en tablas temporales creadas.

Tablas de consulta materializada
Las tablas de este tipo se definen mediante una consulta que se utiliza también para determinar los datos de la tabla. Las tablas de consulta materializadas pueden utilizarse para mejorar el rendimiento de las consultas. Si el gestor de bases de datos determina que una parte de la consulta puede resolverse mediante una tabla de resumen, el gestor de bases de datos puede reescribrir la consulta para que se utilice la tabla de resumen. Esta decisión se basa en los valores de configuración de la base de datos, como los registros especiales CURRENT REFRESH AGE y CURRENT QUERY OPTIMIZATION. Una tabla de resumen es un tipo especializado de tabla de consulta materializada.

Puede crear todos los tipos de tabla anteriores mediante la sentencia CREATE TABLE.

Según el aspecto que vayan a tener sus datos, es posible que observe que un tipo de tabla ofrece capacidades específicas que pueden optimizar el rendimiento del almacenamiento y las consultas. Por ejemplo, si dispone de registros de datos que estarán débilmente agrupados en clústeres (que no aumentan monotónicamente), puede que convenga utilizar una tabla normal e índices. Si dispone de registros de datos que tendrán valores duplicados (pero no exclusivos) en la clave, no debe utilizar una tabla agrupada por rango de clústeres. Además, si no puede permitirse asignar previamente una cantidad de espacio en disco fija para las tablas agrupadas por rango de clústeres que desee, no debe utilizar este tipo de tabla. Si cuenta con datos que pueden potencialmente agruparse en clústeres en múltiples dimensiones, como una tabla para el seguimiento de las ventas minoristas por región geográfica, división y proveedor, es posible que una tabla de clúster multidimensional sea la opción más adecuada para sus necesidades.

Además de los diversos tipos de tabla descritos anteriormente, puede utilizar también características del tipo *particionada*, que pueden mejorar el rendimiento en diversas tareas, como el despliegue de los datos de tabla. Las tablas particionadas pueden también retener mucha más información que una tabla normal no particionada. Asimismo, puede explotar otras funciones, como la *compresión*, que puede ayudarle a reducir considerablemente sus costes de almacenamiento de datos.

Restricciones

En cualquier empresa, con frecuencia los datos se ajustan algunas restricciones o normas. Por ejemplo, un número de empleado debe ser exclusivo. El gestor de bases de datos proporciona *restricciones* como una forma de imponer estas normas.

Están disponibles los siguientes tipos de restricciones:

- Restricciones NOT NULL
- Restricciones exclusivas (o de clave exclusiva)
- Restricciones de clave primaria
- Restricciones de clave foránea (o de integridad referencial)
- Restricciones de comprobación (de tabla)
- Restricciones informativas

Las restricciones sólo se asocian a tablas y se definen como parte del proceso de creación de tablas (utilizando la sentencia CREATE TABLE) o se añaden a la definición de tabla después de haber creado la tabla (utilizando la sentencia ALTER TABLE). Puede utilizar la sentencia ALTER TABLE para modificar restricciones. En la mayoría de casos, las restricciones existentes se pueden eliminar en cualquier momento; esta acción no afecta a la estructura de la tabla o a los datos que se almacenan en ella.

Nota: Las restricciones primarias y exclusivas sólo se asocian a objetos de tabla y con frecuencia se imponen mediante el uso de uno o más de los índices de clave exclusiva o primaria.

Índices

Un *índice* es un conjunto de punteros ordenados lógicamente por los valores de una o varias claves. Los punteros pueden hacer referencia a filas de una tabla, bloqueos de una tabla MDC, datos XML de un objeto de almacenamiento XML, etc.

Los índices se utilizan para:

- Mejorar el rendimiento. En la mayoría de los casos, el acceso a los datos es más rápido con un índice. Aunque no puede crearse un índice para una vista, un índice creado para la tabla en la que se basa una vista puede mejorar a veces el rendimiento de las operaciones en esta vista.
- Asegurar la exclusividad. Una tabla con un índice de unicidad no puede tener filas con claves idénticas.

Cuando se agregan datos a una tabla, se añaden al final (salvo que se hayan ejecutado otras acciones en la tabla o en los datos que se están agregando). No existe un orden inherente para los datos. Cuando se busca una fila de datos determinada, debe buscarse en cada fila de la tabla desde la primera hasta la última. Los índices se utilizan como un medio para acceder a los datos de la tabla según un orden que, de otra manera, podría no estar disponible.

Generalmente, cuando se buscan datos en una tabla, se buscan filas con columnas que tengan valores específicos. Se puede utilizar un valor de columna de una fila de datos para identificar toda la fila. Por ejemplo, un número de empleado definiría probablemente de forma específica a un empleado individual concreto. O bien podría necesitarse más de una columna para identificar la fila. Por ejemplo,

una combinación de nombre y teléfono de cliente. Las columnas de un índice que se utilizan para identificar las filas de datos se denominan *claves*. Se puede utilizar una columna en más de una clave.

Un índice se ordena según los valores que contiene una clave. Las claves pueden ser exclusivas o no exclusivas. Cada tabla debe tener como mínimo una clave exclusiva; pero también puede tener otras claves no exclusivas. Cada índice tiene exactamente una clave. Por ejemplo, puede utilizar el número de ID de empleado (exclusivo) como clave exclusiva para un índice y el número de departamento (no exclusivo) como clave para un índice diferente.

No todos los índices apuntan a filas de una tabla. Los índices de bloque MDC apuntan a extensiones (o bloques) de los datos. Los índices XML para datos XML utilizan expresiones de patrones XML determinados para vías y valores de índices en documentos XML almacenados en una única columna. El tipo de datos de dicha columna debe ser XML. Tanto los índices de bloque MDC como los índices XML son índices generados por el sistema.

Ejemplo

La tabla A de la Figura 1 contiene un índice basado en los números de empleado de la tabla. Este valor de clave proporciona un puntero hacia las filas de la tabla. Por ejemplo, el número de empleado 19 apunta al empleado KMP. Un índice permite un acceso eficaz a las filas de una tabla creando una vía de acceso a los datos mediante punteros.

Se pueden crear índices exclusivos para asegurar la exclusividad de la clave de índice. Una *clave de índice* es una columna o una colección ordenada de columnas en la que se define un índice. La utilización de un índice exclusivo asegura que el valor de cada clave de índice de la columna o columnas indexadas es exclusivo.

La Figura 1 muestra la relación entre un índice y una tabla.

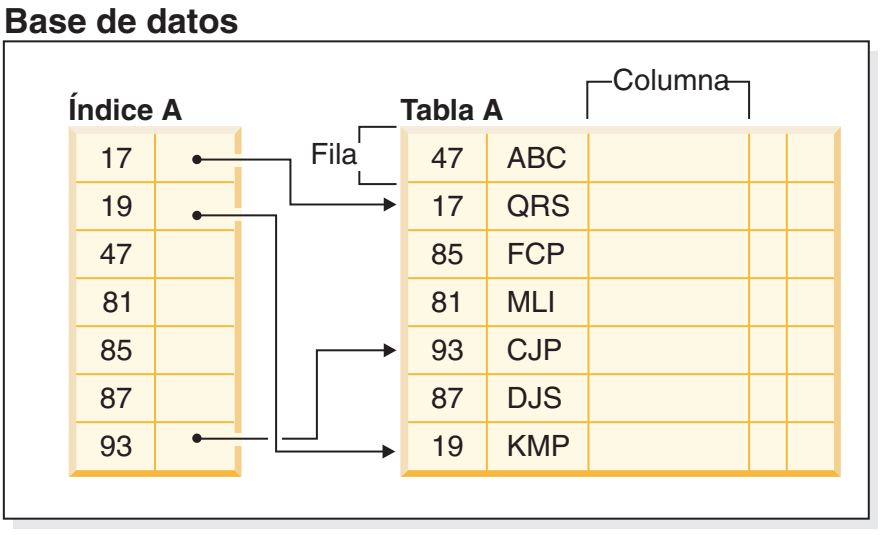


Figura 1. Relación entre un índice y una tabla

La Figura 2 en la página 11 ilustra las relaciones entre algunos objetos de base de datos. También muestra que las tablas, índices y datos extensos se almacenan en

espacios de tabla.

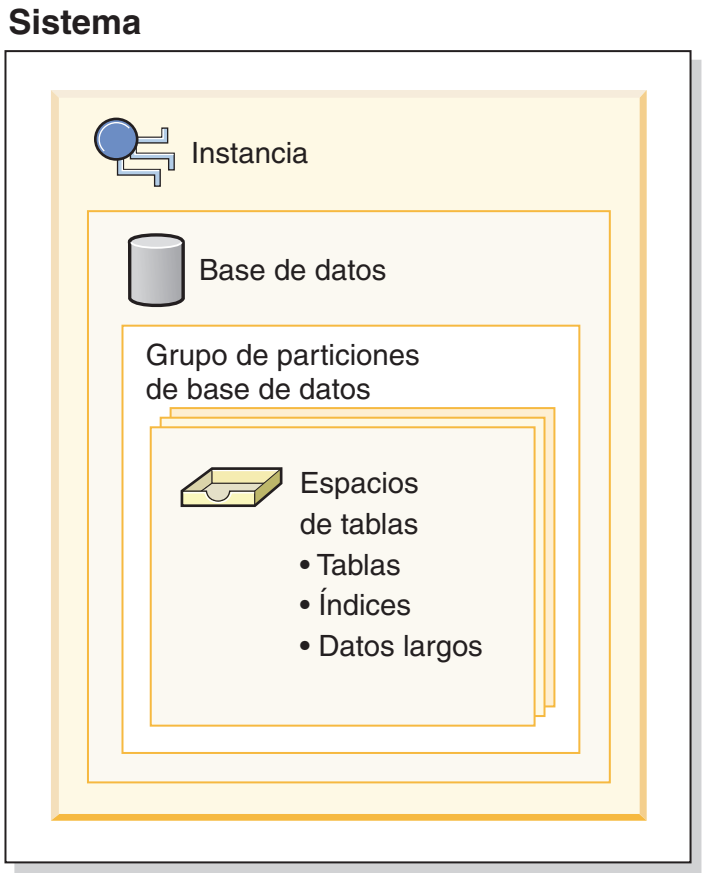


Figura 2. Relaciones entre objetos de base de datos seleccionados

Activadores

Un *activador* define un conjunto de acciones que se ejecutan en respuesta a una operación de inserción, actualización o supresión en una tabla determinada. Cuando se ejecuta una de estas operaciones de SQL, se dice que el activador se ha *activado*. Los activadores son opcionales y se definen mediante la sentencia CREATE TRIGGER.

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

Los activadores son un mecanismo útil para definir e imponer reglas empresariales *transicionales*, que son reglas que incluyen diferentes estados de los datos (por ejemplo, un salario que no se puede aumentar más del 10 por ciento).

La utilización de activadores proporciona la lógica que impone las reglas empresariales en la base de datos. Esto significa que las aplicaciones no son responsables de imponer dichas reglas. La lógica centralizada que se imponen en todas las tablas facilita el mantenimiento, porque no se necesitan realizar cambios en los programas de aplicación cuando cambia la lógica.

Los elementos siguientes se especifican al crear un activador:

- La *tabla sujeto* especifica la tabla para la que se define el activador.
- El *suceso activador* define una operación de SQL específica que modifica la tabla sujeto. El suceso puede ser una operación de inserción, actualización o supresión.
- La *hora de activación del activador* especifica si el activador debería activarse antes o después de que se produzca el suceso activador.

La sentencia que hace que se active un activador incluye un *conjunto de filas afectadas*. Éstas son las filas de la tabla sujeto que se están insertando, actualizando o suprimiendo. La *granularidad del activador* especifica si las acciones del activador se realizan una vez para la sentencia o una vez para cada una de las filas afectadas.

La *acción activada* está formada por una condición de búsqueda opcional y un conjunto de sentencias que se ejecutan siempre que se activa el activador. Las sentencias sólo se ejecutan si la condición de búsqueda se evalúa como verdadera. Si la hora de activación del activador es anterior al suceso activador, las acciones activadas pueden incluir sentencias que seleccionen, establezcan variables de transición y señalen estados de SQL. Si la hora de activación del activador es posterior al suceso activador, las acciones activadas pueden incluir sentencias que seleccionen, inserten, actualicen, supriman o señalen estados de SQL.

La acción activada puede hacer referencia a los valores del conjunto de filas afectadas utilizando *variables de transición*. Las variables de transición utilizan los nombres de las columnas de la tabla sujeto, calificados por un nombre especificado que identifica si la referencia es al valor anterior (antes de la actualización) o al valor nuevo (después de la actualización). El nuevo valor también se puede cambiar utilizando la sentencia variable SET en activadores anteriores, de inserción o actualización.

Otra forma de hacer referencia a los valores del conjunto de filas afectadas consisten en utilizar *tablas de transición*. Las tablas de transición también utilizan los nombres de las columnas de la tabla sujeto pero especifican un nombre para permitir que el conjunto completo de filas afectadas se trate como una tabla. Las tablas de transición sólo se pueden utilizar en activadores AFTER (es decir, no con activadores BEFORE e INSTEAD OF) y es posible definir tablas de transición independientes para los valores anteriores y nuevos.

Se pueden especificar varios activadores para una combinación de tabla, suceso (INSERT, UPDATE, DELETE) o momento de activación (BEFORE, AFTER, INSTEAD OF). Cuando existe más de un activador para una tabla, un suceso o un momento de activación determinados, el orden en que se activan los activadores es igual que el orden en que se han creado. Por consiguiente, el activador creado más recientemente es el último activador que se activa.

La activación de un activador puede provocar una *cascada de activadores*, que es el resultado de la activación de un activador que ejecuta sentencias SQL que provocan la activación de otros activadores o incluso del mismo activador otra vez. Las acciones activadas también pueden causar actualizaciones como resultado de la aplicación de las normas de integridad de referencia para las supresiones que, a su vez, pueden provocar la activación de activadores adicionales. Con una cascada de activadores, se puede activar una cadena de activadores y reglas de supresión de integridad de referencia, lo que puede producir un cambio significativo en la base de datos como resultado de una sola sentencia INSERT, UPDATE o DELETE.

Cuando varios activadores tienen acciones de inserción, actualización o supresión para el mismo objeto, se utiliza un mecanismo de resolución de conflictos, tales como tablas temporales a fin de evitar conflictos de acceso, lo que puede repercutir notablemente en el rendimiento, sobre todo en los entornos de bases de datos particionadas.

Vistas

Una *vista* es un modo eficaz de representar datos sin necesidad de mantenerlos. Una vista no es una tabla real y no necesita un almacenamiento permanente. Se crea y se utiliza un "tabla virtual".

Una *vista* proporciona una manera distinta de ver los datos de una o varias tablas; es una especificación con nombre de una tabla resultante. La especificación es una sentencia SELECT que se ejecuta siempre que se hace referencia a la vista en una sentencia de SQL. Una vista tiene columnas y filas como en una tabla. Todas las vistas se pueden utilizar como si fueran tablas para efectuar una recuperación de datos. Si una vista pueda utilizarse o no en una operación de inserción, actualización o supresión dependerá de su definición.

Una vista puede incluir todas o algunas de las columnas o filas contenidas en las tablas en las que se basa. Por ejemplo, puede unir una tabla de departamentos y una tabla de empleados en una vista, a fin de poder listar todos los empleados de un determinado departamento.

La Figura 3 muestra la relación entre tablas y vistas.

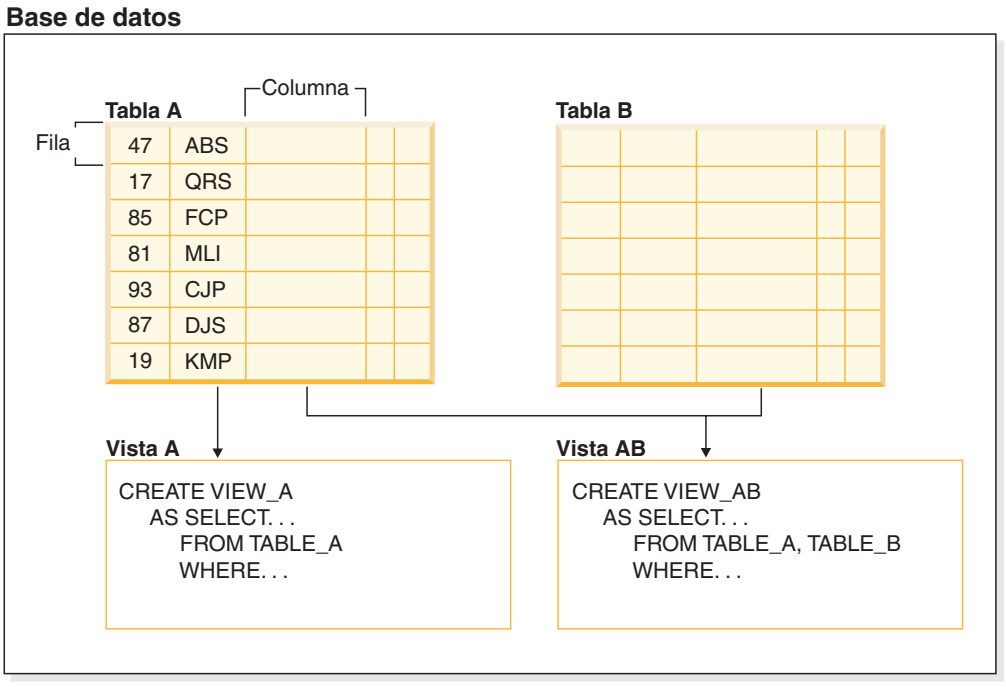


Figura 3. Relaciones entre tablas y vistas

Es posible utilizar las vistas para controlar el acceso a datos sensibles, porque las vistas permiten que muchos usuarios vean presentaciones distintas de los mismos datos. Por ejemplo, es posible que varios usuarios accedan a una tabla de datos sobre empleados. Un director ve los datos sobre sus empleados pero no de los empleados de otro departamento. Un oficial de reclutamiento ve las fechas de

contratación de todos los empleados, pero no sus salarios; un oficial de finanzas ve los salarios, pero no las fechas de contratación. Cada uno de estos usuarios trabaja con una vista derivada de la tabla. Cada vista se parece a una tabla y tiene nombre propio.

Cuando la columna de una vista se deriva directamente de la columna de una tabla base, esa columna de vista hereda las restricciones aplicables a la columna de la tabla. Por ejemplo, si una vista incluye una clave foránea de su tabla, las operaciones de inserción y actualización que utilicen dicha vista están sujetas a las mismas restricciones de referencia a las que lo está la tabla. Asimismo, si la tabla de una vista es una tabla padre, las operaciones de supresión y actualización que utilicen dicha vista estarán sujetas a las mismas reglas que las operaciones de supresión y actualización de la tabla.

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

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

Alias

Un *alias* es un nombre alternativo para un objeto como un módulo, una tabla u otro alias. Se puede utilizar para hacer referencia a un objeto siempre que se pueda hacer referencia directamente a dicho objeto.

Un seudónimo no puede utilizarse en todos los contextos; por ejemplo, no puede utilizarse en la condición de comprobación de una restricción de comprobación. Un alias no puede hacer referencia a una tabla temporal declarada pero puede hacer referencia a una tabla temporal creada.

Como sucede con otros objetos, un alias se puede crear, descartar y tener comentarios asociados. Los alias pueden hacer referencia a otros alias en un proceso denominado *encadenamiento* siempre y cuando no haya referencias circulares. No se requiere ninguna autorización o privilegio especial para utilizar los alias. No obstante, para acceder al objeto al que hace referencia un alias es necesario tener la autorización asociada con ese objeto.

Si un alias se define como un alias *público*, se puede hacer referencia a él mediante su nombre no calificado sin que influya en ello el nombre de esquema por omisión actual. También se puede hacer referencia a él mediante el calificador SYSPUBLIC.

Un *sinónimo* es un nombre alternativo para un alias.

Para obtener más información, consulte la sección sobre alias en identificadores en la publicación *Consulta de SQL, Volumen 1*.

Paquete

Un *paquete* es un objeto almacenado en la base de datos que incluye la información necesaria para procesar las sentencias de SQL asociadas con un archivo fuente de un programa de aplicación.

Se genera mediante una de las siguientes:

- La precompilación de un archivo fuente con el mandato PREP
- La vinculación de un archivo de vinculación que generó el precompilador con el mandato BIND.

Autorización, privilegios y propiedad de objetos

Los usuarios (identificados mediante un ID de autorización) pueden ejecutar satisfactoriamente operaciones únicamente en el caso de que tengan la autorización para realizar la función especificada. Para crear una tabla, un usuario debe tener autorización para crear tablas; para modificar una tabla, un usuario debe tener autorización para modificar la tabla y así sucesivamente.

El gestor de bases de datos necesita que todos los usuarios estén específicamente autorizados para utilizar cada una de las funciones de la base de datos necesarias para realizar una tarea específica. Un usuario puede obtener la autorización necesaria si se le concede dicha autorización a su ID de usuario o mediante pertenencia a un rol o grupo que ostenta dicha autorización.

Existen tres formas de autorización, *autorización administrativa*, *privilegios* y *credenciales LBAC*. Además, la propiedad de los objetos conlleva un grado de autorización sobre los objetos creados. Más abajo se tratan con detalle estas formas de autorización.

Autorización administrativa

La persona o personas con autorización de administrador se encargan de la tarea de controlar el gestor de bases de datos y son responsables de la seguridad e integridad de los datos.

Autorización en el nivel del sistema

Las autorizaciones en el nivel del sistema ofrecen diversos grados de control sobre funciones del nivel de la instancia:

- Autorización SYSADM (administrador del sistema)

  La autorización SYSADM (administrador del sistema) proporciona control sobre todos los recursos que el gestor de bases de datos crea y mantiene. El administrador del sistema posee todas las autorizaciones SYSCTRL, SYSMAINT y SYSMON. El usuario que tiene la autorización SYSADM es responsable de controlar el gestor de bases de datos y de garantizar la seguridad y la integridad de los datos.
- Autorización SYSCTRL

  La autorización SYSCTRL proporciona control sobre las operaciones que afectan a los recursos del sistema. Por ejemplo, un usuario con autorización SYSCTRL puede crear, actualizar, iniciar, detener o descartar una base de datos. Este usuario también puede iniciar o detener una instancia pero no acceder a los datos de las tablas. Los usuarios con autorización SYSCTRL también poseen autorización SYSMON.

- Autorización SYSMAINT

  La autorización SYSMAINT proporciona la autorización necesaria para realizar operaciones de mantenimiento en todas las bases de datos asociadas a una instancia. Un usuario con autorización SYSMAINT puede actualizar la configuración de las bases de datos, realizar una copia de seguridad de una base de datos o de un espacio de tablas, restaurar una bases de datos existente y supervisar una base de datos. Al igual que SYSCTRL, SYSMAINT no proporciona acceso a los datos de las tablas. Los usuarios con autorización SYSMAINT también poseen autorización SYSMON.
- Autorización SYSMON (supervisor del sistema)

  La autorización SYSMON (supervisor del sistema) proporciona la autorización necesaria para utilizar el supervisor del sistema de bases de datos.

Autorización en el nivel de la base de datos

Las autorizaciones del nivel de la base de datos proporcionan control dentro de la base de datos:

- DBADM (administrador de la base de datos)

  El nivel de autorización DBADM proporciona autorización administrativa sobre una sola base de datos. Este administrador de la base de datos posee los privilegios necesarios para crear objetos y emitir mandatos de la base de datos.

  La autorización DBADM sólo puede otorgarla un usuario que tenga la autorización SECADM. La autorización DBADM no puede otorgarse a PUBLIC.
- SECADM (administrador de seguridad)

  El nivel de autorización SECADM proporciona autorización administrativa sobre una sola base de datos. La autorización de administrador de seguridad puede gestionar los objetos de seguridad de la base de datos (roles de base de datos, políticas de control, contextos fiables, componentes de etiqueta de seguridad y etiquetas de seguridad), así como otorgar y revocar todas las autorizaciones y los privilegios de base de datos. Un usuario con autorización SECADM puede transferir la propiedad de los objetos que no posea. También pueden utilizar la sentencia AUDIT para asociar una política de auditoría con una base de datos o un objeto de base de datos determinados en el servidor.

  La autorización SECADM no posee el privilegio inherente de acceder a los datos almacenados en tablas. Sólo puede otorgarla un usuario que tenga la autorización SECADM. La autorización SECADM no puede otorgarse a PUBLIC.
- SQLADM (administrador de SQL)

  El nivel de autorización SQLADM proporciona autorización administrativa para supervisar y ajustar las sentencias de SQL dentro de una única base de datos. Puede otorgarla un usuario que tenga la autorización ACCESSCTROL o SECADM.
- WLMADM (administrador de gestión de carga de trabajo)

  La autorización WLMADM proporciona autorización administrativa para gestionar objetos de gestión de carga de trabajo, como clases de servicio, conjuntos de acciones de trabajo, conjuntos de clases de trabajo y cargas de trabajo. Puede otorgarla un usuario que tenga la autorización ACCESSCTROL o SECADM.
- EXPLAIN (autorización Explain)

  El nivel de autorización EXPLAIN proporciona autorización administrativa para explicar planes de consulta sin obtener acceso a los datos. Sólo puede otorgarla un usuario que tenga la autorización ACCESSCTROL o SECADM.
- ACCESSCTRL (autorización de control de acceso)

El nivel de autorización ACCESSCTRL proporciona autorización administrativa para emitir las sentencias GRANT (y REVOKE) siguientes. La autorización ACCESSCTROL sólo puede otorgarla un usuario que tenga la autorización SECADM. La autorización ACCESSCTRL no puede otorgarse a PUBLIC.

  - GRANT (autorizaciones de bases de datos)

    La autorización ACCESSCTRL no permite a quien la posee conceder autorizaciones ACCESSCTRL, DATAACCESS, DBADM o SECADM. Sólo puede conceder estas autorizaciones un usuario que tiene autorización SECADM.
  - GRANT (privilegios de variable global)
  - GRANT (privilegios de índice)
  - GRANT (privilegios de módulo)
  - GRANT (privilegios de paquete)
  - GRANT (privilegios de rutina)
  - GRANT (privilegios de esquema)
  - GRANT (privilegios de secuencia)
  - GRANT (privilegios de servidor)
  - GRANT (privilegios de tabla, vista o apodo)
  - GRANT (privilegios de espacio de tablas)
  - GRANT (privilegios de carga de trabajo)
  - GRANT (privilegios de objeto XSR)
- DATAACCESS (autorización de acceso a datos)

  El nivel de autorización DATAACCESS proporciona los privilegios y las autorizaciones siguientes. Sólo puede concederla un usuario que tiene la autorización SECADM. La autorización DATAACCESS no puede otorgarse a PUBLIC.

  - Autorización LOAD
  - Privilegio SELECT, INSERT, UPDATE, DELETE sobre tablas, vistas, apodos y tablas de consulta materializada
  - Privilegio EXECUTE sobre paquetes
  - Privilegio EXECUTE sobre módulos
  - Privilegio EXECUTE sobre rutinas

    Salvo en las rutinas de auditoría: AUDIT_ARCHIVE, AUDIT_LIST_LOGS, AUDIT_DELIM_EXTRACT.
- Autorizaciones de base de datos (no administrativas)

  Para realizar actividades como, por ejemplo, crear una tabla o una rutina, o para cargar datos en una tabla, se necesitan autorizaciones específicas sobre las bases de datos. Por ejemplo, la autorización de base de datos LOAD se necesita para utilizar el programa de utilidad load para cargar datos en tablas (un usuario debe tener asimismo el privilegio INSERT sobre la tabla).

Privilegios

Un privilegio es un permiso para realizar una acción o una tarea. Los usuarios autorizados pueden crear objetos, tener acceso a los objetos de los que son propietarios y pueden transmitir privilegios sobre sus propios objetos a otros usuarios mediante la sentencia GRANT.

Los privilegios pueden otorgarse a usuarios individuales, a grupos o a PUBLIC. PUBLIC es un grupo especial que consta de todos los usuarios, incluyendo los

futuros usuarios. Los usuarios que son miembros de un grupo indirectamente sacarán partido a los privilegios otorgados al grupo, en los lugares en los que los grupos están soportados:

El privilegio CONTROL: La posesión del privilegio CONTROL sobre un objeto permite a un usuario acceder a este objeto de la base de datos y otorgar privilegios a otros usuarios sobre dicho objeto o revocarlos.

Nota: El privilegio CONTROL sólo es aplicable a tablas, vistas, apodos, índices y paquetes.

Si un usuario distinto necesita el privilegio CONTROL sobre este objeto, un usuario con autorización SECADM o ACCESSCTRL podrá otorgarle el privilegio CONTROL sobre dicho objeto. Aunque el privilegio CONTROL no puede revocarse del propietario del objeto, el propietario del objeto puede cambiarse utilizando la sentencia TRANSFER OWNERSHIP.

Privilegios individuales: Es posible otorgar privilegios individuales que permitan a un usuario llevar a cabo determinadas tareas sobre objetos concretos. Los usuarios con autorizaciones administrativas ACCESSCTROL SECADM, o con el privilegio CONTROL, pueden otorgar privilegios a los usuarios y revocarlos.

Los privilegios individuales y las autorizaciones de bases de datos permiten una función específica pero no incluyen el derecho a otorgar los mismos privilegios o autorizaciones a otros usuarios. El derecho a otorgar privilegios de tabla, vista, esquema, paquete, rutina y secuencia a otros puede ampliarse a otros usuarios mediante la opción WITH GRANT en la sentencia GRANT. Sin embargo, la opción WITH GRANT no permite a la persona que otorga el privilegio revocar el privilegio después de otorgarlo. Es necesario poseer autorización SECADM, autorización ACCESSCTRL o el privilegio CONTROL para revocar el privilegio.

Privilegios sobre objetos de un paquete o rutina: Cuando un usuario tenga el privilegio de ejecutar un paquete o rutina, éstos no necesitarán necesariamente privilegios específicos sobre los objetos utilizados en el paquete o rutina. Si el paquete o rutina contiene sentencias XQuery o SQL estáticas, los privilegios del propietario del paquete se utilizarán para dichas sentencias. Si el paquete o la rutina contiene sentencias de XQuery o SQL dinámico, el ID de autorización utilizado para la comprobación de privilegios dependerá del valor de la opción BIND de **DYNAMICRULES** del paquete que emita las sentencias de consulta dinámica y de si las sentencias se han emitido cuando el paquete se estaba utilizando en el contexto de una rutina.

Un usuario o grupo puede autorizarse para cualquier combinación de autorizaciones o privilegios individuales. Cuando se asocia un privilegio a un objeto, este objeto debe existir. Por ejemplo, no es posible otorgar a un usuario el privilegio SELECT sobre una tabla a menos que esta tabla se haya creado anteriormente.

Nota: Debe tenerse cuidado cuando se otorguen autorizaciones y privilegios a un nombre de autorización que represente a un usuario o grupo y no se haya creado ningún usuario o grupo con dicho nombre. En el futuro podría crearse un usuario o grupo con dicho nombre de autorización que recibiría automáticamente todas las autorizaciones y privilegios asociados con este nombre de autorización.

La sentencia REVOKE se utiliza para revocar los privilegios otorgados con anterioridad. Revocar un privilegio de un nombre de autorización revoca el privilegio otorgado por todos los nombres de autorización.

Al revocar un privilegio de un nombre de autorización no se revoca este mismo privilegio de ningún otro nombre de autorización al que este nombre de autorización haya otorgado el privilegio. Por ejemplo, supongamos que CLAIRE otorga la opción SELECT WITH GRANT a RICK y más tarde RICK otorga SELECT a BOBBY y a CHRIS. Si CLAIRE revoca el privilegio SELECT de RICK, BOBBY y CHRIS seguirán reteniendo el privilegio SELECT.

Credenciales LBAC

El control de acceso basado en etiquetas (LBAC) permite que el administrador de seguridad decida quién tiene exactamente acceso de grabación y quién tiene acceso de lectura en filas individuales y columnas individuales. El administrador de seguridad configura el sistema LBAC mediante la creación de políticas de seguridad. Una política de seguridad describe los criterios que se utilizan para decidir quién tiene acceso a unos datos determinados. Tan solo se puede utilizar una política de seguridad para proteger una misma tabla pero tablas diferentes pueden estar protegidas por diferentes políticas de seguridad.

Después de crear una política de seguridad, el administrador de seguridad crea objetos de base de datos, llamados etiquetas de seguridad y exenciones que forman parte de dicha política. Una etiqueta de seguridad describe un determinado conjunto de criterios de seguridad. Una exención permite no aplicar una norma para comparar etiquetas de seguridad al usuario que posee la exención, cuando acceden a datos protegidos por dicha política de seguridad.

Una vez creada, una etiqueta de seguridad se puede asociar con columnas y filas individuales de una tabla para proteger los datos que contienen. Los datos protegidos mediante una etiqueta de seguridad se denominan datos protegidos. Un administrador de seguridad permite a los usuarios acceder a datos protegidos otorgándoles etiquetas de seguridad. Cuando un usuario intenta acceder a datos protegidos, la etiqueta de seguridad del usuario se compara con la etiqueta de seguridad que protege los datos. La etiqueta de protección bloquea algunas etiquetas de seguridad y no bloquea otras.

Propiedad del objeto

Cuando se cree un objeto, a un ID de autorización se le asignará la *propiedad* de dicho objeto. Por propiedad se entiende que un usuario está autorizado a referencias el objeto en cualquier sentencia SQL o XQuery que sea de aplicación.

Cuando se crea un objeto en un esquema, el ID de autorización de la sentencia deberá tener el privilegio necesario para crear objetos en el esquema implícita o explícitamente especificado. Es decir, el nombre de autorización deberá ser el propietario del esquema o poseer el privilegio CREATEIN sobre el esquema.

Nota: Este requisito no es aplicable al crear espacios de tabla, agrupaciones de almacenamientos intermedios o grupos de particiones de base de datos. Estos objetos no se crean en los esquemas.

Cuando se crea un objeto, el ID de autorización de la sentencia es el definidor de dicho objeto y se convierte por omisión en el propietario del objeto tras su creación.

Nota: Hay una excepción. Si se ha especificado la opción AUTHORIZATION para la sentencia CREATE SCHEMA, los demás objetos que se creen como parte de la operación CREATE SCHEMA serán propiedad del ID de autorización especificado por la opción AUTHORIZATION. Sin embargo, los objetos creados en el esquema después de la operación de CREATE SCHEMA inicial serán propiedad del ID de autorización asociado a la sentencia CREATE específica.

Por ejemplo, la sentencia `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT)` crea el esquema `SCOTTSTUFF` y la tabla `SCOTTSTUFF.T1`, los cuales son ambos propiedad de SCOTT. Suponga que al usuario BOBBY se le otorga el privilegio CREATEIN sobre el esquema `SCOTTSTUFF` y se crea un índice en la tabla `SCOTTSTUFF.T1`. Puesto que el índice se crea después que el esquema, BOBBY poseerá el índice sobre `SCOTTSTUFF.T1`.

Los privilegios se asignarán al propietario del objeto basándose en el tipo de objeto que está creándose:

- El privilegio CONTROL se otorga implícitamente a las tablas, índices y paquetes recién creados. Este privilegio permite al creador de un objeto acceder al objeto de la base de datos y otorgar privilegios para o desde otros usuarios sobre dicho objeto o revocarlos. Si un usuario distinto necesita el privilegio CONTROL sobre este objeto, un usuario con autorización ACCESSCTRL o SECADM debe otorgarle el privilegio CONTROL sobre dicho objeto. El propietario del objeto no puede revocar el privilegio CONTROL.
- Al privilegio CONTROL se le otorgan implícitamente las vistas recién creadas en el caso de que el propietario del objeto posea el privilegio CONTROL en todas las tablas, vistas y apodos a los que se hace referencia mediante la definición de vista.
- Otros objetos como por ejemplo, los activadores, rutinas, secuencias, espacios de tabla y agrupaciones de almacenamientos intermedios no tienen el privilegio CONTROL asociado a los mismos. Sin embargo, el propietario del objeto recibe automáticamente cada uno de los privilegios asociados al objeto, los cuales incluyen WITH GRANT, cuando esté soportado. Por consiguiente, el propietario del objeto puede conceder estos privilegios a otros usuarios mediante la sentencia GRANT. Por ejemplo, si USER1 crea un espacio de tablas, USER1 tiene automáticamente el privilegio USEAUTH con WITH GRANT OPTION sobre dicho espacio de tablas y puede conceder el privilegio USEAUTH a otros usuarios. Además, el propietario del objeto puede modificar, añadir un comentario o descartar el objeto. Estas autorizaciones son implícitas para el propietario del objeto y no pueden revocarse.

El propietario puede otorgar ciertos privilegios sobre el objeto, como por ejemplo modificar una tabla, privilegios que un usuario que tenga autorización ACCESSCTRL o SECADM puede revocar al propietario. El propietario no puede otorgar ni revocar determinados privilegios sobre el objeto, como por ejemplo el de comentar una tabla. Utilice la sentencia TRANSFER OWNERSHIP para mover dichos privilegios a otro usuario. Cuando se crea un objeto, el ID de autorización de la sentencia es el definidor de dicho objeto y se convierte por omisión en el propietario del objeto tras su creación. Sin embargo, cuando se utiliza el mandato BIND para crear un paquete y se especifica la opción **OWNER** en *id de autorización*, el propietario de estos objetos creado por las sentencias de SQL estático del paquete es el valor de *id de autorización*. Además, si se especifica la cláusula AUTHORIZATION en una sentencia CREATE SCHEMA, el nombre de autorización especificado detrás de la palabra clave AUTHORIZATION será el propietario del esquema.

Un administrador de seguridad o el propietario del objeto podrán utilizar la sentencia TRANSFER OWNERSHIP para cambiar la propiedad de un objeto de base de datos. Por tanto, un administrador puede crear un objeto en nombre de un ID de autorización, creando el objeto que utiliza el ID de autorización como calificador y utilizando a continuación la sentencia TRANSFER OWNERSHIP para transferir la propiedad que el administrador posee sobre el objeto al ID de autorización.

Vistas de catálogo del sistema

El gestor de bases de datos mantiene un conjunto de vistas tablas que contienen información sobre los datos que se encuentran bajo su control. Estas vistas y tablas se conocen en su conjunto como el *catálogo del sistema*.

El catálogo del sistema contiene información acerca de la estructura lógica y física de los objetos de la base de datos como, por ejemplo, tablas, vistas, índices, paquetes y funciones. También contiene información estadística. El gestor de bases de datos garantiza que las descripciones del catálogo del sistema siempre sean precisas.

Las vistas de catálogo del sistema son como cualquier otra vista de la base de datos. Se pueden utilizar sentencias de SQL para consultar los datos de las vistas de catálogo del sistema. Para modificar ciertos valores del catálogo del sistema puede utilizarse un conjunto de vistas actualizables del catálogo del sistema.

Procesos, simultaneidad y recuperación de aplicaciones

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

Más de un proceso de aplicación puede solicitar acceso a los mismos datos al mismo tiempo. El *bloqueo* es el mecanismo utilizado para conservar la integridad de los datos en estas condiciones, de forma que se impide, por ejemplo, que dos procesos de aplicación actualicen la misma fila de datos simultáneamente.

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

Una *unidad de trabajo* (UOW) es una secuencia recuperable de operaciones dentro de un proceso de aplicación. Se inicia una unidad de trabajo cuando comienza un proceso de aplicación, o cuando la UOW anterior finaliza y no se debe a la finalización del proceso de aplicación. Una unidad de trabajo finaliza con una operación de confirmación, una operación de retrotracción o la finalización de un proceso de aplicación. Las operaciones de confirmación o retrotracción solo afectan a los cambios de la base de datos realizados dentro de la UOW que está terminando.

El gestor de la base de datos ofrece una forma de restituir los cambios sin confirmar ejecutados por el proceso de aplicación. Esto puede ser necesario en caso de anomalía en la parte del proceso de aplicación o en caso de producirse una situación de punto muerto o de tiempo de espera excedido de bloqueo. Un proceso de aplicación puede solicitar explícitamente que se cancelen los cambios de su base de datos. Esto se hace utilizando una operación de *retrotracción*.

Mientras estos cambios continúen sin confirmar, los otros procesos de aplicación no podrán visualizarlos y los cambios pueden retrotraerse. No obstante, lo anterior no es cierto si el nivel de aislamiento que prevalece es de lectura no confirmada (UR). Una vez confirmados, otros procesos de aplicación pueden acceder a estos cambios de la base de datos, que ya no pueden retrotraerse.

Tanto la CLI (call level interface) de DB2 como SQL incorporado permiten una modalidad de conexión llamada *transacciones simultáneas*, que admite varias conexiones, cada una de las cuales es una transacción independiente. Una aplicación puede tener múltiples conexiones simultáneas con la misma base de datos.

Los bloqueos que obtiene el gestor de la base de datos en nombre del proceso de aplicación se retienen hasta el final de una UOW, salvo cuando el nivel de aislamiento es estabilidad del cursor (CS, donde se libera el bloqueo a medida que el cursor se mueve de una fila a otra) o lectura no confirmada (UR).

No se puede impedir nunca que un proceso de aplicación realice operaciones debido a sus propios bloqueos. No obstante, si una aplicación utiliza transacciones simultáneas, los bloqueos de una transacción podrían afectar al funcionamiento de una transacción simultánea.

El inicio y la finalización de una UOW definen los *puntos de coherencia* en un proceso de aplicación. Por ejemplo, una transacción bancaria podría suponer la transferencia de fondos de una cuenta a otra. Una transacción de este tipo necesitaría que dichos fondos se restaran de la primera cuenta y se sumaran más tarde a la segunda cuenta. Después de esta sustracción, los datos son incoherentes. La coherencia sólo queda restablecida cuando los fondos se han sumado a la segunda cuenta. Cuando se han finalizado ambos pasos, se puede utilizar la operación de confirmación para finalizar la UOW, poniendo así los cambios a disposición de otros procesos de aplicación. Si se produce una anomalía antes de que finalice la UOW, el gestor de la base de datos retrotraerá todos los cambios sin confirmar para restaurar la coherencia de los datos.

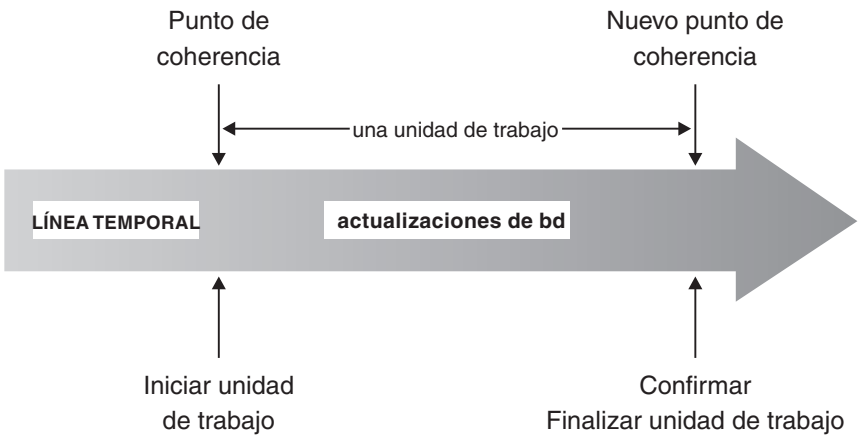


Figura 4. Unidad de trabajo con una sentencia COMMIT

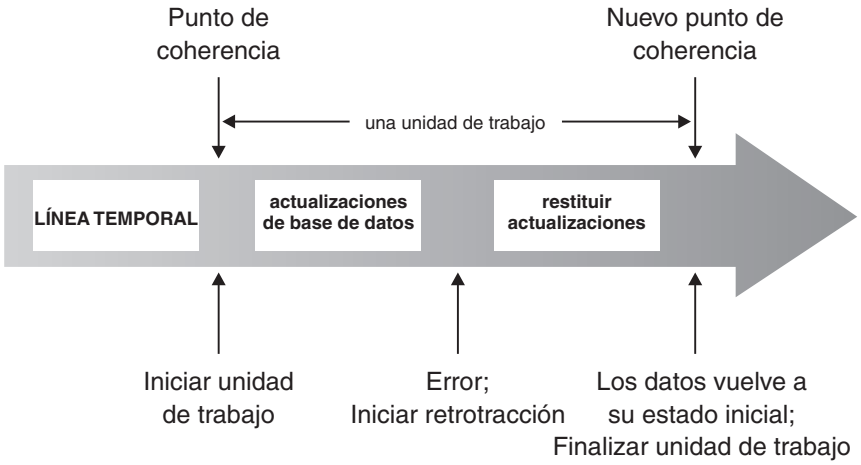


Figura 5. Unidad de trabajo con una sentencia ROLLBACK

Niveles de aislamiento

El *nivel de aislamiento* que está relacionado con un proceso de aplicación determina el grado en el que los datos a los que accede este proceso se bloquean o aíslan de otros procesos que se ejecutan simultáneamente. El nivel de aislamiento estará en vigor durante la duración de una unidad de trabajo.

Por consiguiente, el nivel de aislamiento de un proceso de aplicación especifica:

- El grado en el que las filas leídas o actualizadas por la aplicación están disponibles para otros procesos de aplicación que se ejecutan simultáneamente
- El grado en el que puede afectar a la aplicación la actividad de actualización de otros procesos de aplicación que se ejecutan simultáneamente

El nivel de aislamiento correspondiente a sentencias de SQL estático se especifica como un atributo de un paquete y se aplica a los procesos de aplicación que utilizan ese paquete. El nivel de aislamiento se especifica durante el proceso de preparación del proceso estableciendo la opción de enlace o precompilación ISOLATION. Para sentencias de SQL dinámico, el nivel de aislamiento por omisión es el nivel de aislamiento especificado para el paquete que prepara la sentencia. Utilice la sentencia SET CURRENT ISOLATION para especificar un nivel de aislamiento distinto para las sentencias de SQL dinámico que se emiten dentro de

una sesión. Para obtener más información, consulte "Registro especial CURRENT ISOLATION". Tanto para las sentencias de SQL estático como para las sentencias de SQL dinámico, la *cláusula-aislamiento* de una *sentencia-select* altera temporalmente el registro especial (si se ha establecido) y el valor de la opción de enlace. Para obtener más información, consulte "Sentencia Select".

Los bloqueos imponen los niveles de aislamiento, y el tipo de bloqueo utilizado limita o impide el acceso a los datos por parte de procesos de aplicación simultáneos. Las tablas temporales declaradas y las filas de las mismas no pueden bloquearse, pues sólo la aplicación que las declaró puede acceder a ellas.

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

Compartidos (S)
Con un bloqueo S, se limitan los procesos de aplicación simultáneos a operaciones de sólo lectura de los datos.

De actualización (U)
Con un bloqueo U, se limitan los procesos de aplicación simultáneos a operaciones de sólo lectura de los datos, si dichos procesos no han declarado que podrían actualizar una fila. El gestor de bases de datos asume que el proceso que actualmente mira a una fila podría actualizarla.

Exclusivos (X)
Con un bloqueo X, se evita que los procesos de aplicación simultáneos accedan a los datos de cualquier forma. No se aplica a los procesos de aplicación con un nivel de aislamiento de lectura no confirmada (UR), que pueden leer los datos pero no modificarlos.

Independientemente del nivel de aislamiento, el gestor de bases de datos coloca bloqueos de exclusividad en cada fila que se inserta, actualiza o suprime. Por lo tanto, los niveles de aislamiento aseguran que las filas que cambia un proceso de aplicación durante una unidad de trabajo no las pueda modificar ningún otro proceso de aplicación hasta que la unidad de trabajo haya finalizado.

El gestor de bases de datos permite cuatro niveles de aislamiento.

- "Lectura repetible (RR)"
- "Estabilidad de lectura (RS)" en la página 25
- "Estabilidad del cursor (CS)" en la página 26
- "Lectura no confirmada (UR)" en la página 26

Nota: Algunos servidores de bases de datos de sistema principal soportan el nivel de aislamiento *sin confirmación (NC)*. En otros servidores de bases de datos, este nivel de aislamiento tiene un comportamiento similar al nivel de aislamiento de lectura no confirmada.

A continuación se ofrece una descripción detallada de cada nivel de aislamiento, en orden descendente para el impacto del rendimiento, pero en orden ascendente para la atención necesaria al acceder a los datos o actualizarlos.

Lectura repetible (RR)

El nivel de aislamiento de *lectura repetible* bloquea todas las filas a las que hace referencia una aplicación durante una unidad de trabajo (UOW). Si una aplicación emite dos veces una sentencia SELECT en la misma unidad de trabajo, se devuelve

el mismo resultado cada vez. En RR, no se pueden producir actualizaciones perdidas, accesos a datos no confirmados, lecturas no repetibles ni lecturas fantasmas.

En RR, una aplicación puede recuperar filas y trabajar en ellas tantas veces como sea necesario hasta que finalice la UOW. Sin embargo, ninguna otra aplicación puede actualizar, suprimir o insertar una fila que afectaría al conjunto de resultados hasta que finalice la UOW. Las aplicaciones que se ejecutan en el nivel de aislamiento de RR no pueden ver los cambios no confirmados de otras aplicaciones. Este nivel de aislamiento garantiza que todos los datos devueltos permanezcan sin cambios hasta el momento en el que la aplicación vea los datos, aunque se utilicen tablas temporales o el bloqueo de filas.

Se bloquean todas las filas referenciales, no sólo las filas recuperadas. Por ejemplo, si se analizan 10.000 filas y se aplican predicados a ellas, los bloqueos se conservan en las 10.000 filas, aunque sólo se califiquen, por ejemplo, 10 de ellas. Otra aplicación no puede insertar ni actualizar una fila que se añadiría a la lista de filas referenciadas por una consulta, si dicha consulta se fuera a ejecutar otra vez. De este modo se evita que se produzcan lecturas fantasma.

Dado que RR puede adquirir un número considerable de bloqueos, este número podría superar los límites especificados por los parámetros de configuración de base de datos **locklist** y **maxlocks**. Para impedir el reajuste de bloqueos, el optimizador podría optar por adquirir un único bloqueo de nivel de tabla para un análisis de índice, si el reajuste de bloqueos parece probable. Si no se quiere que se realice el bloqueo de nivel de tabla, se deberá utilizar el nivel de aislamiento de estabilidad de lectura.

Al evaluar las restricciones de referencia, el servidor DB2 podría actualizar en ocasiones el nivel de aislamiento utilizado para los análisis de la tabla foránea a RR, independientemente del nivel de aislamiento determinado previamente por el usuario. Esto genera que se retengan bloqueos adicionales hasta la hora de la confirmación, lo que aumenta la probabilidad de que se produzca un tiempo muerto o un tiempo de espera de bloqueo excedido. Para evitar estos problemas, cree un índice que contenga únicamente las columnas de clave foránea y que pueda utilizar en su lugar el análisis de integridad referencial.

Estabilidad de lectura (RS)

El nivel de aislamiento de *estabilidad de lectura* bloquea únicamente las filas que recupera una aplicación durante una unidad de trabajo. RS garantiza que otros procesos de aplicación no puedan modificar ninguna fila calificada leída durante una UOW hasta que la UOW finalice, ni que se pueda leer ninguna fila modificada por otro proceso de aplicación hasta que dicho proceso confirme el cambio. En RS, no se pueden producir lecturas de datos no confirmados ni lecturas no repetibles. No obstante, se pueden realizar lecturas fantasmas.

Este nivel de aislamiento garantiza que todos los datos devueltos permanezcan sin cambios hasta el momento en el que la aplicación vea los datos, aunque se utilicen tablas temporales o el bloqueo de filas.

El nivel de aislamiento de RS proporciona un nivel alto de simultaneidad y una vista estable de los datos. Para ello, el optimizador garantiza que no se obtienen bloqueos de nivel de tabla hasta que se produce el reajuste de bloqueos.

El nivel de aislamiento de RS es adecuado para las aplicaciones que:

- Funcionan en entornos simultáneos.
- Requieren filas calificadas para permanecer estables mientras dura una unidad de trabajo.
- No emiten la misma consulta más de una vez durante una unidad de trabajo, y no requieren el mismo conjunto de resultados cuando se emite una consulta más de una vez durante una unidad de trabajo.

Estabilidad del cursor (CS)

El nivel de aislamiento de *estabilidad del cursor* bloquea cualquier fila a la que se esté accediendo durante una transacción mientras el cursor esté posicionado en dicha fila. Este bloqueo continúa vigente hasta que se capta la siguiente fila o finaliza la transacción. Sin embargo, si alguno de los datos de la fila ha cambiado, el bloqueo se retiene hasta que se confirma el cambio.

En este nivel de aislamiento, ninguna otra aplicación puede actualizar o suprimir una fila mientras esté posicionado en ella un cursor actualizable. En CS, no se puede acceder a los datos no confirmados de otras aplicaciones. No obstante, se pueden realizar lecturas no repetibles y lecturas fantasmas.

CS es el nivel de aislamiento por omisión. Es adecuado para obtener la máxima simultaneidad y cuando se necesita ver únicamente los datos confirmados.

Nota: Según la semántica *confirmada actualmente* incorporada en la Versión 9.7, solamente se devuelven datos confirmados, tal y como sucedía anteriormente, con la diferencia de que ahora los lectores no esperan a que los actualizadores liberen los bloqueos de fila. En su lugar, los lectores devuelven datos basados en la versión confirmada actualmente, es decir, los datos anteriores al inicio de la operación de grabación.

Lectura no confirmada (UR)

El nivel de aislamiento de *lectura no confirmada* permite a una aplicación acceder a los cambios no confirmados de otras transacciones. Además, UR no impide que otras aplicaciones accedan a una fila que se está leyendo, salvo que la aplicación esté intentado alterar o descartar la tabla.

En UR, se pueden producir accesos a datos no confirmados, lecturas no repetibles y lecturas fantasmas. Este nivel de aislamiento es adecuado si se ejecutan consultas contra tablas de sólo lectura, o si se emiten sentencias SELECT únicamente, y la visualización de datos que no han confirmado otras aplicaciones no supone ningún problema.

UR funciona de forma distinta en los cursores de sólo lectura y actualizables.

- Los cursores de sólo lectura pueden acceder a la mayoría de los cambios no confirmados de otras transacciones.
- Las tablas, vistas e índices que están creando o descartando otras transacciones no están disponibles mientras la transacción está procesando. Otros cambios realizados por otras transacciones se pueden leer antes de su confirmación o retrotracción. Los cursores actualizables que funcionan en UR se comportan como si el nivel de aislamiento fuera CS.

Si una aplicación de lectura no confirmada utiliza cursores ambiguos, podría utilizar el nivel de aislamiento de CS durante su ejecución. Los cursores ambiguos

se pueden reajustar a CS si el valor de la opción BLOCKING del mandato PREP o BIND es UNAMBIG (valor por omisión). Para evitar que se produzca el reajuste:

- Modifique los cursores en el programa de aplicación para que no sean ambiguos. Cambie las sentencias SELECT para que incluyan la cláusula FOR READ ONLY.
- Permita que los cursores del programa de aplicación continúen siendo ambiguos, pero precompile el programa o vincúlelo con las opciones BLOCKING ALL y STATICREADONLY YES para que se traten como si fueran de sólo lectura cuando se ejecute el programa.

Comparación de niveles de aislamiento

En la Tabla 1 se resumen los niveles de aislamiento soportados.

Tabla 1. Comparación de niveles de aislamiento

	UR	**CS**	**RS**	**RR**
¿Una aplicación puede ver los cambios no confirmados realizados por otros procesos de aplicación?	Sí	No	No	No
¿Una aplicación puede actualizar los cambios no confirmados realizados por otros procesos de aplicación?	No	No	No	No
¿Pueden otros procesos de aplicación afectar a la operación de volver a ejecutar una sentencia? [1]	Sí	Sí	Sí	No [2]
¿Pueden otros procesos de aplicación actualizar filas actualizadas? [3]	No	No	No	No
¿Pueden otros procesos de aplicación que se ejecutan a un nivel de aislamiento que no sea UR leer las filas actualizadas?	No	No	No	No
¿Pueden otros procesos de aplicación que se ejecutan al nivel de aislamiento UR leer las filas actualizadas?	Sí	Sí	Sí	Sí
¿Pueden otros procesos de aplicación actualizar las filas a las que se ha accedido? [4]	Sí	Sí	No	No
¿Pueden otros procesos de aplicación leer las filas a las que se ha accedido?	Sí	Sí	Sí	Sí
¿Pueden otros procesos de aplicación actualizar o suprimir la fila actual? [5]	Sí/No [6]	Sí/No [6]	No	No

Tabla 1. Comparación de niveles de aislamiento (continuación)

	UR	CS	RS	RR

Nota:

1. Un ejemplo del *fenómeno de lectura fantasma* es el siguiente: la unidad de trabajo UW1 lee el conjunto de *n* filas que satisface alguna condición de búsqueda. La unidad de trabajo UW2 inserta una o varias filas que satisfacen la misma condición de búsqueda y luego la confirma. Si UW1 posteriormente repite su lectura con la misma condición de búsqueda, ve un conjunto de resultados diferente: las filas que se habían leído originalmente más las filas que UW2 insertó.
2. Si las credenciales de control de acceso basado en etiquetas (LBAC) cambian entre lecturas, es posible que los resultados para la segunda lectura sean diferentes, ya que se tiene acceso a filas diferentes.
3. El nivel de aislamiento no ofrece protección a la aplicación si ésta lee de una tabla y escribe en ella. Por ejemplo, una aplicación abre un cursor en una tabla y entonces realiza una operación de inserción, actualización o supresión en la misma tabla. Es posible que la aplicación vea datos incoherentes al buscar más filas desde el cursor abierto.
4. Un ejemplo del *fenómeno de lectura no repetible* es el siguiente: la unidad de trabajo UW1 lee una fila. La unidad de trabajo UW2 modifica dicha fila y realiza una confirmación. Si UW1 posteriormente lee esa fila de nuevo, es posible que vea un valor distinto.
5. Un ejemplo del *fenómeno de lectura sucia* es el siguiente: la unidad de trabajo UW1 modifica una fila. La unidad de trabajo UW2 lee dicha fila antes de que UW1 realice una confirmación. Si UW1 posteriormente retrotrae los cambios, UW2 ha leído datos no existentes.
6. Con UR o CS, si el cursor no es actualizable, la fila actual puede actualizarse o suprimirse por otros procesos de aplicación en algunos casos. Por ejemplo, el almacenamiento intermedio puede hacer que la fila actual del cliente sea distinta de la fila actual en el servidor. Además, al utilizar la semántica confirmada actualmente con CS, una fila que se está leyendo puede tener actualizaciones sin confirmar pendientes. En este caso, la versión confirmada actualmente de la fila siempre se devuelve a la aplicación.

Resumen de los niveles de aislamiento

La Tabla 2 lista los problemas de simultaneidad asociados con los diferentes niveles de aislamiento.

Tabla 2. Resumen de los niveles de aislamiento

Nivel de aislamiento	Acceso a los datos no confirmados	Lecturas no repetibles	Lecturas fantasma
Lectura repetible (RR)	No es posible	No es posible	No es posible
Estabilidad de lectura (RS)	No es posible	No es posible	Posible
Estabilidad del cursor (CS)	No es posible	Posible	Posible
Lectura no confirmada (UR)	Posible	Posible	Posible

El nivel de aislamiento influye no solamente en el grado de aislamiento entre aplicaciones sino también en las características de rendimiento de una aplicación individual, ya que los recursos de proceso y memoria necesarios para obtener y liberar bloqueos varían en función del nivel de aislamiento. El potencial de puntos muertos también cambia según el nivel de aislamiento. La Tabla 3 en la página 29

ofrece una sencilla heurística para ayudarle a seleccionar un nivel de aislamiento inicial para su aplicación.

Tabla 3. Directrices para seleccionar un nivel de aislamiento

Tipo de aplicación	Estabilidad de datos alta necesaria	Estabilidad de datos alta no necesaria
Transacciones de lectura-grabación	RS	CS
Transacciones de sólo lectura	RR o RS	UR

Espacios de tabla

Un *espacio de tabla* es una estructura de almacenamiento que contiene tablas, índices, objetos de gran tamaño y datos extensos. Se utilizan para organizar datos de una base de datos en agrupaciones de almacenamiento lógico que se relacionan con la ubicación del sistema en el que están almacenados los datos. Los espacios de tablas se almacenan en grupos de particiones de base de datos.

La utilización de espacios de tablas para organizar el almacenamiento ofrece diversas ventajas:

Recuperabilidad
La colocación en el mismo espacio de tablas de objetos de los que debe realizarse conjuntamente la copia de seguridad o la restauración facilita las operaciones de copia de seguridad y restauración, ya que se puede ejecutar la copia de seguridad o restauración de todos los objetos en espacios de tablas con un único mandato. Si se dispone de índices y tablas particionados distribuidos en diversos espacios de tablas, se puede realizar la copia de seguridad o restauración solamente de las particiones de índice y datos que residen en un espacio de tablas dado.

Más tablas
El número de tablas que se pueden almacenar en un espacio de tablas dado es limitado; si se necesitan más tablas que puedan contenerse en un espacio de tablas, sólo se tendrán que crear espacios de tablas adicionales para ellas.

Flexibilidad del almacenamiento
Con los espacios de tablas DMS y SMS, se pueden especificar los dispositivos de almacenamiento que se utilizarán para almacenar los datos. Se puede optar, por ejemplo, por almacenar los datos operativos actuales en espacios de tablas que residen en dispositivos más rápidos, y los datos históricos en espacios de tablas que residen en dispositivos más lentos (y más baratos).

Capacidad para aislar los datos en agrupaciones de almacenamientos intermedios para mejorar el rendimiento y la utilización de la memoria
Si se dispone de un conjunto de objetos, como tablas o índices, para los que se emiten consultas con frecuencia, se puede asignar una agrupación de almacenamientos intermedios al espacio de tablas en el que residen mediante una única sentencia CREATE o ALTER TABLESPACE. Se puede asignar espacios de tablas temporales a su propia agrupación de almacenamientos intermedios para aumentar el rendimiento de las actividades, como clasificaciones o uniones. En algunos casos, podría tener más sentido definir agrupaciones de almacenamientos intermedios más pequeñas para datos a los que se accede con poca frecuencia, o para aplicaciones que requieren un acceso muy aleatorio a una tabla muy

grande. En estos casos, no es necesario mantener los datos en la agrupación de almacenamientos intermedios durante más tiempo del requerido para una única consulta.

Los espacios de tablas están formados por uno o varios *contenedores*. Un contenedor puede ser un nombre de directorio, un nombre de dispositivo o un nombre de archivo. Un único espacio de tablas puede tener varios contenedores. Es posible crear múltiples contenedores (de uno o varios espacios de tablas) en el mismo dispositivo de almacenamiento físico, aunque el rendimiento será mucho mayor si cada contenedor creado utiliza un dispositivo de almacenamiento diferente. Si se están utilizando espacios de tablas de almacenamiento automático, el administrador de la base de datos se ocupa automáticamente de la creación y gestión de los contenedores. Si no se están utilizando espacios de tablas de almacenamiento automático, el usuario debe definir y gestionar él mismo los contenedores.

La Figura 6 ilustra la relación entre tablas y espacios de tabla dentro de una base de datos, y los contenedores asociados con dicha base de datos.

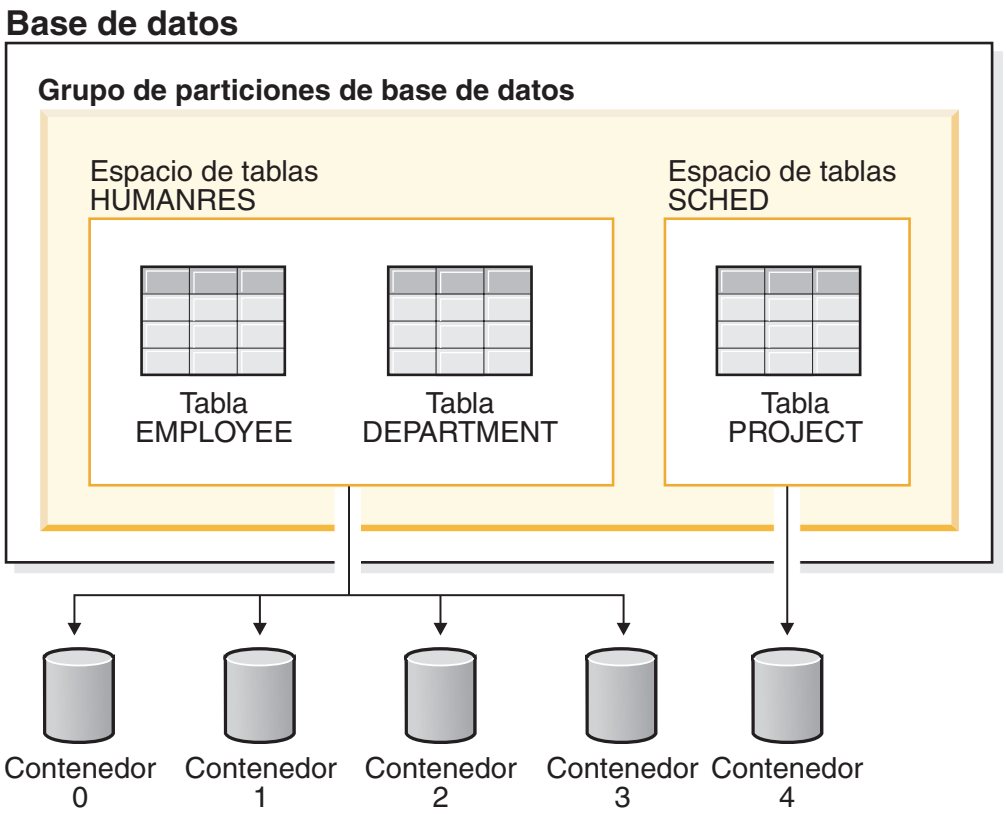


Figura 6. Espacios de tabla y tablas en una base de datos

Las tablas EMPLOYEE y DEPARTMENT están en el espacio de tabla HUMANRES, el cual se distribuye entre los contenedores 0, 1, 2 y 3. La tabla PROJECT está en el espacio de tabla SCHED en el contenedor 4. Este ejemplo muestra cada contenedor que existe en un disco separado.

El gestor de bases de datos intenta equilibrar la carga de datos entre contenedores. Como resultado, se utilizan todos los contenedores para almacenar datos. El número de páginas que el gestor de bases de datos graba en un contenedor antes

de utilizar un contenedor diferente recibe el nombre de *tamaño de extensión*. Un gestor de bases de datos no siempre empieza a almacenar los datos de tabla en el primer contenedor.

La Figura 7 muestra el espacio de tabla HUMANRES con un tamaño de extensión de dos páginas de 4 KB y cuatro contenedores, cada uno de ellos con un número reducido de extensiones asignadas. Las tablas DEPARTMENT y EMPLOYEE tienen siete páginas y se distribuyen entre los cuatro contenedores.

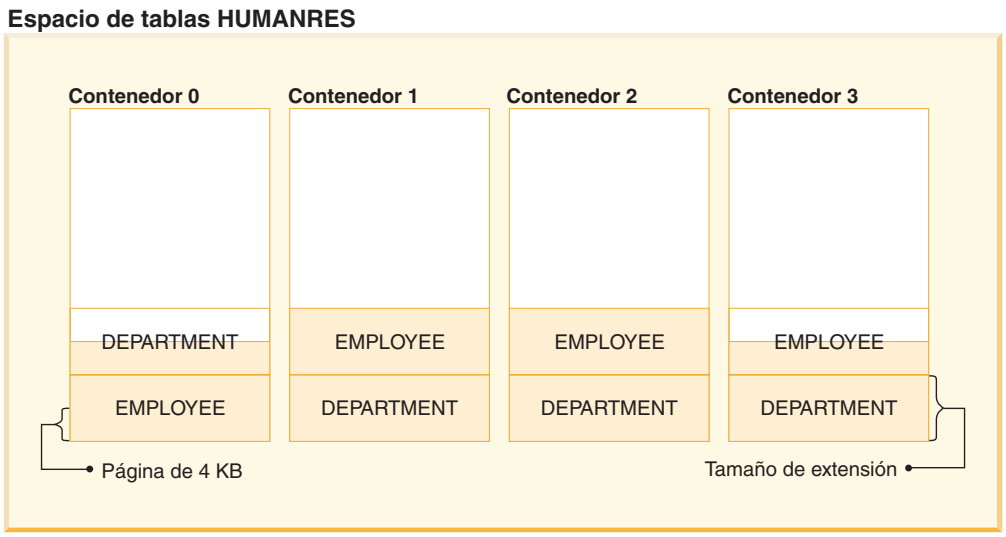


Figura 7. Contenedores y extensiones de un espacio de tabla

Conversión de caracteres

Una *serie* es una secuencia de bytes que puede representar caracteres. Todos los caracteres de una serie tienen una representación de codificación común. En algunos casos, puede ser necesario convertir estos caracteres a una representación de códigos diferente, conocida como *conversión de caracteres*.

Cuando la conversión de caracteres es necesaria, es automática. Las aplicaciones no requieren invocar explícitamente la conversión de caracteres, puesto que el servidor y el cliente de base de datos DB2 llevan a cabo la conversión de caracteres necesaria automáticamente.

La conversión de caracteres puede producirse cuando una sentencia de SQL se ejecuta de manera remota. Tenga en cuenta, por ejemplo, los casos siguientes en los que las representaciones de codificación pueden ser distintas en el sistema de envío y en el de recepción:

- Los valores de las variables del lenguaje principal se envían desde el peticionario de aplicaciones al servidor de aplicaciones.
- Los valores de las columnas del resultado se envían desde el servidor de aplicaciones al peticionario de aplicaciones.

A continuación se muestra una lista de los términos utilizados al tratar la conversión de caracteres:

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

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

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

elemento de código
Patrón de bits que representa de forma exclusiva un carácter.

esquema de codificación
Conjunto de normas utilizadas para representar datos de tipo carácter, por ejemplo:

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

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

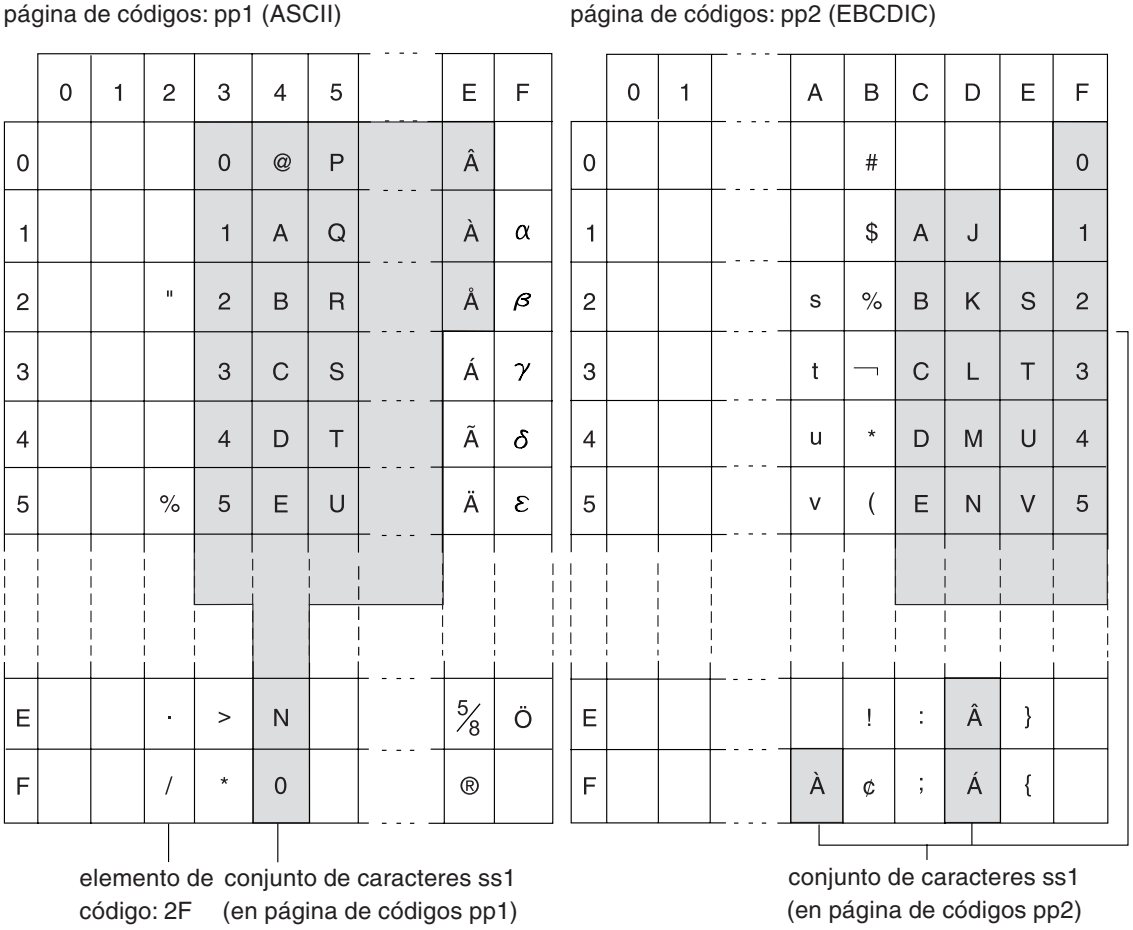


Figura 8. Correlación de un juego de caracteres en diferentes páginas de códigos

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

Página de códigos de base de datos
La página de códigos de base de datos se almacena en el archivo de configuración de la base de datos. El valor se especifica cuando se crea la base de datos y no puede modificarse.

Página de códigos de aplicación
La página de códigos bajo la que se ejecuta la aplicación. No es necesariamente la misma página de códigos bajo la que se ha vinculación la aplicación.

Página de códigos de la sección
La página de códigos bajo la que se ejecuta la sentencia de SQL. Normalmente, la página de códigos de la sección es la página de códigos de la base de datos. Sin embargo, se utiliza la página de códigos Unicode (UTF-8) como página de códigos de la sección si:

- La sentencia hace referencia a una tabla que se ha creado con el esquema de codificación Unicode en una base de datos que no es Unicode
- La sentencia hace referencia a una función de tabla que se ha definido con PARAMETER CCSID UNICODE en una base de datos que no es Unicode.

Página de códigos 0
Representa una serie derivada de una expresión que contiene un valor FOR BIT DATA o un valor BLOB.

Las páginas de códigos de series de caracteres tienen los atributos siguientes:

- Las columnas pueden estar en la página de códigos de la base de datos, en la página de códigos Unicode (UTF-8) o en la página de códigos 0 (si se ha definido como FOR BIT DATA o BLOB).
- Las constantes y los registros especiales (por ejemplo, USER, CURRENT SERVER) están en la página de códigos de selección. Cuando se vincula una sentencia de SQL con la base de datos, las constantes se convierten, en caso necesario, de la página de códigos de la aplicación a la página de códigos de la base de datos y, a continuación, a la página de códigos de la sección.
- Las variables del lenguaje principal de entrada se encuentran en la página de códigos de la aplicación. Para la Versión 8, los datos de serie de variables del lenguaje principal de entrada se convierten, si es necesario, de la página de códigos de la aplicación a la página de códigos de la sección antes de su utilización. La excepción se produce cuando se utiliza una variable del lenguaje principal en un contexto en el que se interpretará como datos de bit; por ejemplo, cuando la variable del lenguaje principal debe asignarse a una columna que está definida como FOR BIT DATA.

Se utiliza un conjunto de normas para determinar los atributos de página de códigos para las operaciones que combinan objetos de serie como, por ejemplo, operaciones escalares, operaciones con conjuntos o concatenaciones. Los atributos de la página de códigos se utilizan para determinar los requisitos para la conversión de páginas de códigos de las series durante la ejecución.

Soporte multicultural y sentencias de SQL

La codificación de las sentencias de SQL no depende del idioma. Las palabras clave de SQL se deben escribir tal como se muestran, aunque se pueden escribir en mayúsculas, minúsculas o mayúsculas y minúsculas combinadas. Los nombres de los objetos de base de datos, de las variables de lenguaje principal y de las etiquetas de programa que aparecen en una sentencia de SQL deben ser caracteres soportados por la página de códigos de aplicación.

El servidor no convierte los nombres de archivo. Para codificar un nombre de archivo, utilice el conjunto ASCII invariable o proporcione la vía de acceso en los valores hexadecimales que se almacenan físicamente en el sistema de archivos.

En un entorno de varios bytes, existen cuatro caracteres que se consideran especiales que no pertenecen al juego de caracteres invariable. Estos caracteres son:

- Los caracteres de porcentaje de doble byte y de subrayado de doble byte utilizados en el proceso LIKE.
- El carácter de espacio de doble byte se utiliza para el relleno de espacios en blanco en series gráficas y otros lugares.
- El carácter de sustitución de doble byte, utilizado como sustitución durante la conversión de página de códigos cuando no existe ninguna correlación entre una página de códigos de fuente y una página de códigos de destino.

Los elementos de código para cada uno de estos caracteres, por página de códigos, son los siguientes:

Tabla 4. Elementos de código para caracteres especiales de doble byte

Página de códigos	Porcentaje de doble byte	Subrayado de doble byte	Espacio de doble byte	Carácter de sustitución de doble byte
932	X'8193'	X'8151'	X'8140'	X'FCFC'
938	X'8193'	X'8151'	X'8140'	X'FCFC'
942	X'8193'	X'8151'	X'8140'	X'FCFC'
943	X'8193'	X'8151'	X'8140'	X'FCFC'
948	X'8193'	X'8151'	X'8140'	X'FCFC'
949	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
950	X'A248'	X'A1C4'	X'A140'	X'C8FE'
954	X'A1F3'	X'A1B2'	X'A1A1'	X'F4FE'
964	X'A2E8'	X'A2A5'	X'A1A1'	X'FDFE'
970	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
1381	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
1383	X'A3A5'	X'A3DF'	X'A1A1'	X'A1A1'
13488	X'FF05'	X'FF3F'	X'3000'	X'FFFD'
1363	X'A3A5'	X'A3DF'	X'A1A1'	X'A1E0'
1386	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
5039	X'8193'	X'8151'	X'8140'	X'FCFC'
1392	X'A3A5'	X'A3DF'	-	-

La página de códigos 5488 es equivalente a 1392 para los programas de utilidad EXPORT, IMPORT y LOAD.

Para bases de datos Unicode, el espacio GRAPHIC es X'0020', que es diferente del espacio GRAPHIC de X'3000' utilizado para las bases de datos eucJP (Extended UNIX Code - Japón) y eucTW (Extended UNIX Code - Taiwán)

X'0020' y X'3000' son caracteres de espacio en el estándar Unicode. Se deberá tener en cuenta la diferencia en los elementos de código de espacio GRAPHIC al comparar datos de estas bases de datos EUC con datos de una base de datos Unicode.

Conexión a bases de datos relacionales distribuidas

Las bases de datos relacionales distribuidas se crean mediante protocolos y funciones de peticionario-servidor formales.

Un *peticionario de aplicaciones* da soporte al extremo correspondiente a la aplicación en una conexión. Transforma una petición de base de datos procedente de la aplicación en protocolos de comunicaciones adecuados para ser utilizados en la red de bases de datos distribuidas. Estas peticiones las recibe y procesa un *servidor de bases de datos* situado en el otro extremo de la conexión. Mediante un trabajo conjunto, el peticionario de aplicaciones y el servidor de bases de datos manejan las consideraciones acerca de las comunicaciones y la ubicación, de forma que la aplicación pueda funcionar como si estuviera accediendo a una base de datos local.

Conexión a bases de datos relacionales distribuidas

Un proceso de aplicación debe conectarse al servidor de aplicaciones de un gestor de bases de datos para que se puedan ejecutar las sentencias de SQL que hacen referencia a tablas o vistas. La sentencia CONNECT establece una conexión entre un proceso de aplicación y su servidor.

Hay dos tipos de sentencias CONNECT:

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

La CLI (call level interface) de DB2 y SQL incorporado dan soporte a una modalidad de conexión llamada *transacciones simultáneas* que permite múltiples conexiones, cada una de las cuales es una transacción independiente. Una aplicación puede tener múltiples conexiones simultáneas con la misma base de datos.

El servidor de aplicaciones puede ser local o remoto con respecto al entorno en el que se inicia el proceso. Existe un servidor de aplicaciones, aunque el entorno no esté utilizando bases de datos relacionales distribuidas. Este entorno incluye un directorio local que describe los servidores de aplicaciones que pueden identificarse en una sentencia CONNECT.

El servidor de aplicaciones ejecuta la forma vinculada de una sentencia de SQL estático que hace referencia a tablas o vistas. La sentencia vinculada se toma de un paquete que el gestor de bases de datos ha creado previamente mediante una operación de vinculación.

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

Supervisores de sucesos que escriben en tablas, archivos y conexiones

Algunos supervisores de sucesos pueden configurarse para grabar información sobre sucesos de base de datos en tablas, conexiones o archivos.

Los supervisores de sucesos se utilizan para reunir información sobre la base de datos y las aplicaciones conectadas cuando se producen los eventos especificados. Los sucesos representan transiciones en la actividad de la base de datos, como por ejemplo, conexiones, puntos muertos, sentencias o transacciones. Es posible definir un supervisor de sucesos por el tipo de suceso o los sucesos que desea supervisar. Por ejemplo, un supervisor de sucesos espera que se produzca un punto muerto. Cuando se produce, reúne la información sobre las aplicaciones implicadas y los bloqueos en contención.

Para crear un supervisor de sucesos, utilice la sentencia de SQL CREATE EVENT MONITOR. Los supervisores de sucesos sólo reúnen datos de sucesos cuando están activos. Para activar o desactivar un supervisor de sucesos, utilice la sentencia de SQL SET EVENT MONITOR STATE. El estado de un supervisor de sucesos (si está activo o inactivo) puede determinarse por la función de SQL EVENT_MON_STATE.

Supervisores de sucesos que escriben en tablas, archivos y conexiones

Cuando se ejecuta la sentencia de SQL CREATE EVENT MONITOR, la definición del supervisor de sucesos que crea se almacena en las tablas de catálogo del sistema de bases de datos siguientes:

- SYSCAT.EVENTMONITORS: los supervisores de sucesos definidos para la base de datos.
- SYSCAT.EVENTS: Los sucesos supervisados para la base de datos.
- SYSCAT.EVENTTABLES: las tablas destino de los supervisores de sucesos de tablas.

Cada supervisor de sucesos tiene su propia vista lógica privada de los datos de la instancia en los elementos del supervisor. Si un supervisor de eventos determinado se desactiva y se vuelve a activar, se restablece su vista de estos contadores. Sólo se ve afectado el supervisor de sucesos que acaba de activarse; todos los otros supervisores de sucesos continuarán utilizando sus vistas de los valores del contador (más cualquier adición nueva).

La salida del supervisor de sucesos puede dirigirse a las tablas de SQL no particionadas, a un archivo o a una área de interconexión con nombre.

Nota: El supervisor de sucesos de punto muerto detallados, DB2DETAILDEADLOCK, que ha quedado en desuso, se crea por omisión para cada base de datos y se inicia cuando se activa la base de datos. Para evitar la actividad general que provoca este monitor de sucesos, descártelo. Ya no se recomienda utilizar el elemento de supervisor DB2DETAILDEADLOCK. Este supervisor de sucesos en desuso podría eliminarse en un futuro 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.

Partición de bases de datos en varias particiones de base de datos

El gestor de bases de datos permite una gran flexibilidad para repartir los datos entre varias particiones (nodos) de una base de datos particionada. Los usuarios pueden elegir, mediante la declaración de claves de distribución, cómo distribuirán los datos, así como determinar sobre qué y cuántas particiones de base de datos se extenderán sus datos de tabla, seleccionando el grupo de particiones de base de datos y el espacio de tablas en el que deberán almacenarse los datos.

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

En una base de datos particionada, la clave de distribución se utiliza para distribuir datos de tabla por un conjunto de particiones de base de datos. Los datos de índice también se particionan con sus tablas correspondientes y se almacenan localmente en cada partición de base de datos.

Para poder utilizar las particiones de base de datos para almacenar datos, es preciso definirlas en el gestor de bases de datos. Las particiones de base de datos están definidas en un archivo llamado `db2nodes.cfg`.

Partición de bases de datos en varias particiones de base de datos

La clave de distribución de una tabla en un espacio de tablas de un grupo de particiones de una base de datos particionada se especifica en la sentencia CREATE TABLE o en la sentencia ALTER TABLE. Si no se especifica, por omisión se creará una clave de distribución para una tabla a partir de la primera columna de la clave primaria. Si no hay ninguna clave primaria definida, la clave de distribución por omisión será la primera columna definida en dicha tabla que tenga un tipo de datos que no sea un tipo de datos LOB o largo. Las tablas de las bases de datos particionadas han de tener, como mínimo, una columna que no sea un tipo de datos LOB ni largo. Una tabla de un espacio de tablas que se encuentre en un grupo de particiones de base de datos de una sola partición tendrá una clave de distribución sólo si se especifica explícitamente.

Las filas se colocan en una partición de base de datos de la forma siguiente:

1. Se aplica un algoritmo de hash (función de partición de bases de datos) a todas las columnas de la clave de distribución, lo que produce la generación de un valor de índice de correlación de distribución.
2. El número de partición de base de datos en dicho valor de índice de la correlación de distribución, identifica la partición de base de datos en la que se almacenará la fila.

El gestor de bases de datos permite la *desagrupación parcial*, lo que significa que se puede distribuir una tabla en un subconjunto de particiones de base de datos del sistema (es decir, un grupo de particiones de base de datos). Las tablas no tienen que distribuirse obligatoriamente por todas las particiones de base de datos del sistema.

El gestor de bases de datos tiene la posibilidad de reconocer si los datos a los que se está accediendo para una unión o una subconsulta se encuentran en la misma partición del mismo grupo de particiones de base de datos. Esto se conoce como *colocación de tablas*. Las filas de las tablas colocadas con los mismos valores de clave de distribución se encuentran en la misma partición de base de datos. El gestor de bases de datos puede elegir realizar el proceso de unión o subconsulta en la partición de base de datos en la que están almacenados los datos. Esto puede tener ventajas significativas para el rendimiento.

Las tablas colocadas deben:

- Estar en el mismo grupo de particiones de base de datos, uno que no se esté redistribuyendo. (Durante la redistribución, es posible que las tablas del grupo de particiones de base de datos utilicen correlaciones de distribución diferentes; no están colocadas).
- Tener claves de distribución con el mismo número de columnas.
- Hacer que las columnas correspondientes de la clave de distribución sean compatibles con la partición.
- Estar en un grupo de particiones de base de datos de una sola partición definido en la misma partición de base de datos.

Comportamiento de objetos grandes en las tablas con particiones

Una tabla con particiones utiliza un esquema de organización de datos donde los datos de las tablas se dividen en varios objetos de almacenamiento, llamados particiones de datos o rangos, de acuerdo con los valores de una o más columnas de clave de particionamiento de la tabla. Los datos de una tabla determinada se particionan en varios objetos de almacenamiento basándose en las especificaciones proporcionadas en la cláusula PARTITION BY de la sentencia CREATE TABLE.

Estos objetos de almacenamiento pueden estar en varios espacios de tablas, en el mismo espacio de tablas o en una combinación de ambos.

Un objeto grande de una tabla particionada queda, por omisión, almacenado en el mismo espacio de tablas que su correspondiente objeto de datos. Esto se aplica a las tablas particionadas que utilizan solamente uno o varios espacios de tablas. Cuando los datos de una tabla particionada se almacenan en varios espacios de tablas, los datos de objetos grandes también se almacenan en varios espacios de tablas.

Utilice la cláusula LONG IN de la sentencia CREATE TABLE para alterar temporalmente este comportamiento por omisión. Puede especificar una lista de espacios de tablas para la tabla donde se almacenarán los datos grandes. Si decide alterar temporalmente el comportamiento por omisión, el espacio de tablas especificado en la cláusula LONG IN debe ser grande. Si especifica que los datos grandes de una o más particiones de datos se deben almacenar en un espacio de tablas diferente, debe hacerlo así para todas las particiones de datos de la tabla. Es decir, no puede tener datos grandes almacenados de manera remota para algunas particiones de datos y almacenados localmente para otras. Si utiliza el comportamiento por omisión o la cláusula LONG IN para alterar temporalmente el comportamiento por omisión, se crea un objeto grande para que se corresponda con cada partición de datos. En los espacios de tablas SMS, los datos grandes deben residir en el mismo espacio de tablas que el objeto de datos al que pertenece. Todos los espacios de tabla utilizados para almacenar objetos de datos largos correspondientes a cada partición de datos deben tener los mismos: tamaño de página, tamaño de extensión, mecanismo de almacenaje (DMS o SMS) y tipo de almacenaje (regular o grande). Los espacios de tabla grandes remotos deben ser de tipo LARGE y no pueden ser SMS.

Por ejemplo, la siguiente sentencia CREATE TABLE crea objetos para los datos CLOB para cada partición de datos del mismo espacio de tablas que los datos:

```
CREATE TABLE documento(id INT, contenido CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

Puede utilizar LONG IN para situar los datos CLOB en uno o más espacios de tablas grandes, diferentes de aquellos donde se encuentran los datos.

```
CREATE TABLE documento(id INT, contenido CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

Nota: Sólo se permite una única cláusula LONG IN en el nivel de tabla y para cada partición de datos.

Sistemas federados de DB2

Sistemas federados

Un *sistema federado* es un tipo especial de sistema de gestión de bases de datos distribuidas (DBMS). Un sistema federado consta de una instancia DB2 que funciona como un servidor federado, una base de datos que actúa como la base de

datos federada, una o más fuentes de datos y clientes (usuarios y aplicaciones) que acceden a la base de datos y a fuentes de datos.

Con un sistema federado, puede enviar peticiones distribuidas a varias fuentes de datos utilizando de una sola sentencia de SQL. Por ejemplo, puede unir datos que estén ubicados en una tabla DB2, una tabla Oracle y un archivo con identificador XML en una única sentencia de SQL. La figura siguiente muestra los componentes de un sistema federado y un ejemplo de las fuentes de datos a las que el usuario puede acceder.

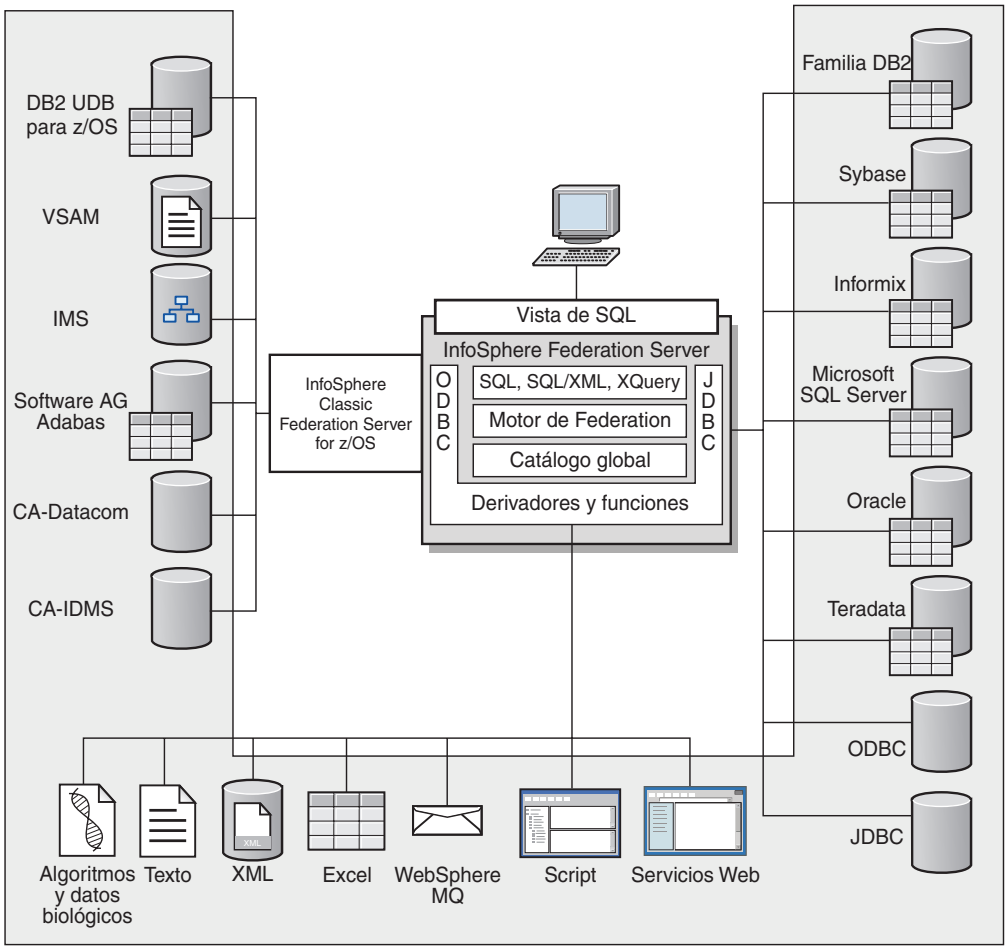


Figura 9. Componentes de un sistema federado

La potencia de un sistema federado se encuentra en su capacidad de:

- Correlacionar datos desde tablas locales y fuentes de datos remotas, como si todos los datos estuviesen almacenados localmente en la base de datos federada
- Actualizar datos en las fuentes de datos relacionales, como si los datos estuvieran almacenados en la base de datos federada
- Mover datos a fuentes de datos relacionales y desde las mismas
- Sacar partido de las fuerzas de proceso de fuentes de datos, mediante el envío de consultas a las fuentes de datos para procesarlas
- Compensar las limitaciones de SQL existentes en la fuente de datos, mediante el proceso de partes de una petición distribuida en el servidor federado

¿Qué es una fuente de datos?

En un sistema federado, una *fuente de datos* puede ser una base de datos relacional (como Oracle o Sybase) o una fuente de datos no relacional (como un archivo con identificador XML).

Por medio de algunas fuentes de datos, puede acceder a otras fuentes de datos. Por ejemplo, con el derivador ODBC se puede acceder a fuentes de datos de IBM InfoSphere Classic Federation Server for z/OS como DB2 para z/OS, IMS, CA-IDMS, CA-Datacom, Software AG Adabas y VSAM.

El método, o protocolo, que se utiliza para acceder a una fuente de datos depende del tipo de fuente de datos. Por ejemplo, DRDA se utiliza para acceder a fuentes de datos DB2 para z/OS.

Las fuentes de datos son autónomas. Por ejemplo, el servidor federado puede enviar consultas a fuentes de datos Oracle al mismo tiempo que aplicaciones Oracle acceden a estas fuentes de datos. Un sistema federado no monopoliza o restringe el acceso al resto de fuentes de datos, más allá de restricciones de integridad y bloqueo.

La base de datos federada

Para usuarios finales y aplicaciones cliente, las fuentes de datos aparecen como una única base de datos colectiva en el sistema de base de datos DB2. Los usuarios y las aplicaciones interactúan con la *base de datos federada* gestionada por el servidor federado.

La base de datos federada contiene un catálogo del sistema que almacena información sobre los datos. El catálogo del sistema de la base de datos federada contiene entradas que identifican las fuentes de datos y sus características. El servidor federado consulta la información que está almacenada en el catálogo del sistema de la base de datos federada y el derivador de la fuente de datos para determinar cuál es el mejor plan para procesar las sentencias de SQL.

El sistema federado procesa sentencias de SQL como si los datos de las fuentes de datos fuesen tablas relacionales ordinarias o vistas dentro de la base de datos federada. Como resultado de ello:

- El sistema federado puede correlacionar datos relacionales con datos en formatos no relacionales. Esto también se aplica cuando las fuentes de datos utilizan distintos dialectos de SQL o cuando no dan soporte a SQL.
- Las características de la base de datos federada tienen prioridad cuando existen diferencias entre las características de la base de datos federada y las características de las fuentes de datos. Los resultados de la consulta se ajustan a la semántica de DB2, incluso si se utilizan datos de otras fuentes de datos no DB2 para calcular el resultado de la consulta.

  Ejemplos:

  – La página de códigos que el servidor federado utiliza es diferente a la página de códigos utilizada por la fuente de datos. En este caso, los datos de carácter de la fuente de datos se convierten en base a la página de códigos utilizada por la base de datos federada, cuando los datos se devuelven a un usuario federado.

  – La secuencia de clasificación que el servidor federado utiliza es diferente a la secuencia de clasificación utilizada por la fuente de datos. En este caso, cualquier operación de clasificación en datos de carácter se realiza en el servidor federado en lugar de en la fuente de datos.

Compilador de SQL

El compilador de SQL de DB2 recopila información para ayudarle a procesar consultas.

Para obtener datos desde fuentes de datos, los usuarios y las aplicaciones envían consultas en SQL a la base de datos federada. Cuando se envía una consulta, el compilador de SQL de DB2 consulta información del catálogo global y del derivador de fuentes de datos para ayudarle a procesar la consulta. Esto incluye información sobre la conexión a la fuente de datos, información de servidor, correlaciones, información de índice y estadísticas de proceso.

Derivadores y módulos de derivador

Los *derivadores* son mecanismos mediante los cuales la base de datos federada interactúa con fuentes de datos. La base de datos federada utiliza rutinas almacenadas en una biblioteca llamada un *módulo de derivador* para implementar un derivador.

Estas rutinas permiten a la base de datos federada realizar operaciones tales como conectar con una fuente de datos y recuperar datos de la misma interactivamente. Normalmente, el propietario de la instancia federada utiliza la sentencia CREATE WRAPPER para registrar un derivador en la base de datos federada. Puede registrar un derivador como protegido o fiable utilizando la opción de derivador DB2_FENCED.

Debe crear un derivador para cada tipo de fuente de datos al que desee acceder. Por ejemplo, desea acceder a tres tablas de base de datos DB2 para z/OS, una tabla de DB2 para System i, dos tablas de Informix y una vista de Informix. En este caso, necesita crear un derivador para los objetos de fuente de datos DB2 y un derivador para los objetos de fuente de datos Informix. Después de registrar estos derivadores en la base de datos federada, puede utilizar estos derivadores para acceder a otros objetos desde esas fuentes de datos. Por ejemplo, puede utilizar el derivador DRDA con todos los objetos de fuente de datos de la familia DB2-DB2 Database para Linux, UNIX y Windows, DB2 para z/OS, DB2 para System i y DB2 Server para VM y VSE.

Para identificar los datos específicos (nombre, ubicación, etc.) de cada objeto de fuente de datos, debe utilizar las definiciones y apodos del servidor.

Un derivador realiza muchas tareas. A continuación se indican algunas de ellas:

- Se conecta con la fuente de datos. El derivador utiliza la API de conexión estándar de la fuente de datos.
- Envía consultas a la fuente de datos.
  - Para las fuentes de datos que dan soporte a SQL, la consulta se envía en SQL.
  - Para las fuentes de datos que no dan soporte a SQL, la consulta se traduce al lenguaje de consulta nativo de la fuente de datos o se convierte en una serie de llamadas de API de la fuente de datos.
- Recibe los conjuntos de resultados de la fuente de datos. El derivador utiliza las API estándar de la fuente de datos para recibir los conjuntos de resultados.
- Responde a consultas de base de datos federada sobre las correlaciones de tipo de datos por omisión para una fuente de datos. El derivador contiene las correlaciones de tipos por omisión que se utilizan cuando se crean apodos para un objeto de fuente de datos. Para los derivadores relacionales, las correlaciones de tipos de datos que el usuario crea alteran temporalmente las correlaciones de

tipos de datos por omisión. Las correlaciones de tipos de datos definidos por el usuario se almacenan en el catálogo global.

- Responde a consultas de base de datos federada sobre las correlaciones de funciones por omisión para una fuente de datos. La base de datos federada necesita información de correlación de tipo de datos con la finalidad de planificar las consultas. El derivador contiene información que la base de datos federada necesita para determinar si las funciones de DB2 se correlacionan con funciones de la fuente de datos y cómo se correlacionan las funciones. El Compilador de SQL utiliza esta información para determinar si la fuente de datos es capaz de realizar las operaciones de consulta. Para los derivadores relacionales, las correlaciones de funciones que crea el usuario alteran temporalmente las correlaciones de tipos de funciones por omisión. Las correlaciones de funciones definidas por el usuario se almacenan en el catálogo global.

Las *opciones de derivador* se utilizan para configurar el derivador o para definir el modo en que IBM InfoSphere Federation Server utiliza el derivador.

Definiciones de servidor y opciones de servidor

Después de haber creado los derivadores para las fuentes de datos, el propietario de la instancia federada define las fuentes de datos para la base de datos federada.

El propietario de la instancia proporciona un nombre para identificar la fuente de datos y otra información relacionada con la fuente de datos. Esta información incluye:

- El tipo y la versión de la fuente de datos
- El nombre de base de datos para la fuente de datos (solo para RDBMS)
- Metadatos que son específicos de la fuente de datos

Por ejemplo, una fuente de datos de la familia DB2 puede tener varias bases de datos. La definición debe especificar la base de datos a la que puede conectarse el servidor federado. En cambio, una fuente de datos Oracle tiene una sola base de datos, y el servidor federado puede conectarse a la base de datos sin conocer su nombre. El nombre de la base de datos no está incluido en la definición de servidor federado de una fuente de datos Oracle.

El nombre y la otra información que el propietario de la instancia proporciona al servidor federado se denominan, colectivamente, *definición de servidor*. Las fuentes de datos responden a las peticiones de datos y son servidores por derecho propio.

Las sentencias CREATE SERVER y ALTER SERVER se utilizan para crear y modificar una definición de servidor.

Parte de la información de una definición de servidor se almacena en forma de *opciones de servidor*. Cuando cree definiciones de servidor, es importante que entienda las opciones que puede especificar acerca del servidor.

Las opciones de servidor se pueden definir de forma que se conserven de una conexión a otra con la base de datos o bien se pueden definir para que sean vigentes durante una sola conexión.

Correlaciones de usuarios

Una *correlación de usuario* es una asociación entre un ID de autorización en el servidor federado y la información necesaria para conectar con la fuente de datos remota.

Para crear una correlación de usuario, debe utilizar la sentencia CREATE USER MAPPING. En la sentencia, se especifica el ID de autorización local, el nombre local del servidor de fuente de datos remota que se especifica en la definición de servidor y el ID y contraseña remotos.

Por ejemplo, supongamos que ha creado una definición de servidor para un servidor remoto y ha especificado 'argon' como nombre local para el servidor remoto. Para hacer que Mary pueda acceder al servidor remoto, cree esta correlación de usuario:

```
CREATE USER MAPPING FOR Mary
SERVER argon
OPTIONS (REMOTE_AUTHID 'ID_remoto', REMOTE_PASSWORD 'contraseña_remota')
```

Cuando Mary emite una sentencia de SQL para conectarse con el servidor remoto, el servidor federado realiza estos pasos:

1. Recupera la correlación de usuario de Mary
2. Descifra la contraseña remota 'contraseña_remota' asociada con el servidor remoto
3. Llama al derivador para conectar con el servidor remoto
4. Pasa al derivador el ID remoto 'ID_remoto' y la contraseña remota descifrada
5. Crea una conexión con el servidor remoto para Mary

Por omisión, el servidor federado almacena la correlación de usuario en la vista SYSCAT.USEROPTIONS del catálogo global y cifra las contraseñas remotas. Como alternativa, puede utilizar un depósito externo, por ejemplo, un archivo o un servidor LDAP, para almacenar correlaciones de usuarios. Para proporcionar la interfaz entre el servidor federado y el depósito externo, debe crear un conector de correlación de usuario.

No importa cómo almacene las correlaciones de usuarios, limite con cuidado el acceso a ellas. Si las correlaciones de usuarios están comprometidas, los datos de las bases de datos remotas pueden ser vulnerables a actividades no autorizadas.

Apodos y objetos de fuente de datos

Un *apodo* es un identificador que se utiliza para identificar el objeto de fuente de datos al que desea acceder. Se hace referencia al objeto que identifica un apodo como *objeto de fuente de datos*.

Un apodo no es un nombre alternativo para un objeto de fuente de datos de la misma forma que un alias es un nombre alternativo. Un apodo es el puntero mediante el que el servidor federado hace referencia al objeto. Normalmente los apodos se definen mediante la sentencia CREATE NICKNAME junto con determinadas opciones de columna de apodo y opciones de apodo.

Cuando una aplicación cliente o un usuario envía una petición distribuida al servidor federado, la petición no necesita especificar las fuentes de datos. En lugar de ello, la petición especifica los objetos de fuente de datos utilizando sus apodos. Los apodos están correlacionados con objetos específicos contenidos en la fuente de datos. Estas correlaciones eliminan la necesidad de calificar los apodos con los

nombres de las fuentes de datos. La ubicación de los objetos de fuente de datos es transparente a la aplicación cliente o al usuario.

Suponga que define el apodo *DEPT* para representar una tabla de bases de datos Informix llamada *NFX1.PERSON*. El servidor federado admite la sentencia SELECT * FROM *DEPT*. Sin embargo, la sentencia SELECT * FROM *NFX1.PERSON* no está permitida desde el servidor federado (excepto en una sesión de paso a través) a menos que haya una tabla local en el servidor federado llamada *NFX1.PERSON*.

Cuando crea un apodo para un objeto de fuente de datos, se añaden metadatos acerca del objeto al catálogo global. El optimizador de consultas utiliza estos metadatos, así como la información del derivador, para facilitar el acceso al objeto de fuente de datos. Por ejemplo, si un apodo es para una tabla que tiene un índice, el catálogo global contiene información acerca del índice y el derivador contiene las correlaciones entre los tipos de datos de DB2 y los tipos de datos de fuente de datos.

Los apodos para objetos que utilizan control de acceso basado en etiquetas (LBAC) no se colocan en la antememoria. Por lo tanto, los datos en el objeto permanecen seguros. Por ejemplo, si utiliza el derivador Oracle (Net8) para crear un apodo en una tabla que utiliza Oracle Label Security (seguridad de etiqueta de Oracle), la tabla se identifica automáticamente como segura. Los datos de apodo resultantes no se pueden poner en la antememoria. Como consecuencia, las tablas de consultas materializadas no se pueden crear en los mismos. La utilización de LBAC garantiza que sólo vean la información los usuarios con los privilegios de seguridad adecuados. Para apodos creados antes de que LBAC estuviera soportado, debe utilizar la sentencia ALTER NICKNAME para inhabilitar la colocación en la antememoria. LBAC está soportado tanto por derivador de DRDA (para fuentes de datos que utilizan DB2 para Linux, UNIX, and Windows versión 9.1 y posteriores) como por el derivador de Net8.

Opciones de columna de apodo

Puede proporcionar al catálogo global información de metadatos adicional acerca del objeto con apodo. Estos metadatos describen valores en determinadas columnas del objeto de fuente de datos. El usuario asigna estos metadatos a parámetros llamados *opciones de columna de apodo*.

Las opciones de columna de apodo indican al derivador que debe manejar los datos de una columna de forma distinta a como normalmente los manejaría. El compilador de SQL y el optimizador de consultas utilizan los metadatos para desarrollar mejores planes para acceder a los datos.

Las opciones de columna de apodo también se utilizan para proporcionar otros elementos de información al derivador. Por ejemplo, para las fuentes de datos XML, se utiliza una opción de columna de apodo para indicar al derivador la expresión XPath que debe utilizar cuando el derivador analice la columna fuera del documento XML.

Con la federación, el servidor DB2 trata el objeto de fuente de datos al que hace referencia un apodo como si fuese una tabla de DB2 local. Como resultado, el usuario puede definir opciones de columna de apodo para cualquier objeto de fuente de datos para el que se cree un apodo. Algunas opciones de columna de apodo están pensadas para tipos determinados de fuentes de datos y sólo pueden aplicarse a esas fuentes de datos.

Suponga que una fuente de datos tiene una secuencia de clasificación que difiere de la secuencia de clasificación de la base de datos federada. Normalmente, el servidor federado no clasificaría ninguna columna que contuviese datos de carácter en la fuente de datos. Devolvería los datos a la base de datos federada y realizaría la clasificación localmente. Sin embargo, suponga que los datos de la columna son de tipo carácter (CHAR o VARCHAR) y que sólo contiene caracteres numéricos ('0','1',...,'9'). Puede indicar este hecho asignando el valor 'Y' a la opción de columna de apodo NUMERIC_STRING. Esto proporciona al optimizador de consultas de DB2 la opción de realizar la clasificación en la fuente de datos. Si la clasificación se realiza de forma remota, puede evitar la actividad que supone transferir los datos al servidor federado y realizar la clasificación localmente.

Puede definir opciones de columna de apodo para apodos relacionales utilizando las sentencias ALTER NICKNAME. Puede definir opciones de columna de apodo para apodos no relacionales utilizando las sentencias CREATE NICKNAME y ALTER NICKNAME.

Correlaciones de tipos de datos

Los tipos de datos en la fuente de datos deben correlacionarse con los tipos de datos DB2 correspondientes de manera que el servidor federado pueda recuperar datos desde las fuentes de datos.

A continuación se muestran algunos ejemplos de correlaciones de tipos de datos por omisión:

- El tipo FLOAT de Oracle se correlaciona con el tipo DOUBLE de DB2
- El tipo DATE de Oracle se correlaciona con el tipo TIMESTAMP de DB2
- El tipo DATE de DB2 para z/OS™ se correlaciona con el tipo DATE de DB2

Para la mayor parte de fuentes de datos, las correlaciones de tipos por omisión se encuentran en los derivadores. Las correlaciones de tipos por omisión para fuentes de datos DB2 están en el derivador DRDA. Las correlaciones de tipos por omisión para Informix están en el derivador INFORMIX, etc.

Para algunas fuentes de datos no relacionales, hay que especificar la información de tipos de datos en la sentencia CREATE NICKNAME. Los tipos de datos DB2 correspondientes deben especificarse para cada columna del objeto de fuente de datos cuando se crea el apodo. Cada columna debe correlacionarse con un campo o columna determinados del objeto de la fuente de datos.

Para las fuentes de datos relacionales, puede alterar temporalmente las correlaciones de tipos de datos definidas por omisión. Por ejemplo, el tipo de datos INTEGER de Informix por omisión está correlacionado con el tipo de datos INTEGER de DB2. Puede alterar temporalmente las correlaciones por omisión y correlacionar el tipo de datos INTEGER de Informix con el tipo de datos DECIMAL(10,0) de DB2.

El servidor federado

En un sistema federado se hace referencia al servidor DB2 como el servidor federado. Puede configurarse el número de instancias de DB2 que se desee para que funcionen como servidores federados. Puede utilizar instancias de DB2 existentes como servidores federados o puede crear nuevas instancias específicamente para el sistema federado.

La instancia de DB2 que gestiona el sistema federado se denomina servidor dado que responde a las peticiones de los usuarios finales y de las aplicaciones cliente. Con frecuencia, el servidor federado envía partes de las peticiones que recibe a las fuentes de datos para su proceso. Una operación de desplazamiento es una operación que se procesa de forma remota. La instancia de DB2 que gestiona el sistema federado se denomina servidor federado, aunque actúe como cliente cuando envía las peticiones a las fuentes de datos.

Como cualquier otro servidor de aplicaciones, el servidor federado es una instancia del gestor de bases de datos. Los procesos de aplicación se conectan y envían peticiones a la base de datos que está dentro del servidor federado. Sin embargo, existen dos características principales que lo diferencian de otros servidores de aplicaciones:

- Un servidor federado está configurado para recibir peticiones que podrían destinarse total o parcialmente a las fuentes de datos. El servidor federado distribuye esas peticiones a las fuentes de datos.
- Como otros servidores de aplicaciones, un servidor federado utiliza protocolos de comunicación DRDA (mediante TCP/IP) para comunicarse con instancias de la familia DB2. Sin embargo, a diferencia de otros servidores de aplicaciones, un servidor federado utiliza el cliente nativo de la fuente de datos para acceder a la fuente de datos. Por ejemplo, un servidor federado utiliza Sybase Open Client para acceder a fuentes de datos Sybase y un controlador ODBC de Microsoft SQL Server para acceder a fuentes de datos Microsoft SQL Server.

Fuentes de datos soportadas

Son muchas las fuentes de datos a las que puede acceder utilizando un sistema federado.

IBM InfoSphere Federation Server soporta las fuentes de datos que se muestran en las tablas siguientes. La primera tabla enumera los requisitos para el software de cliente de datos. El software de cliente debe adquirirse por separado salvo que se indique lo contrario.

Debe instalar el software de cliente de las fuentes de datos a las que desea acceder. El software de cliente debe estar instalado en el mismo sistema que IBM InfoSphere Federation Server. También necesita el SDK de Java adecuado para utilizar algunas herramientas, como el Centro de control de DB2, y para crear y ejecutar aplicaciones Java, incluidos procedimientos almacenados y funciones definidas por el usuario.

Para obtener la información más actualizada, consulte la página de requisitos de fuente de datos en Internet.

Tabla 5. Fuentes de datos soportadas, requisitos del software de cliente y soporte de sistemas operativos desde 32 bits.

			Sistema operativo y arquitectura de hardware de 32 bits	
			X86-32	**X86-32**
Fuente de datos	**Versiones que reciben soporte**	**Software de cliente**	**Linux, RedHat Enterprise Linux (RHEL), SUSE**	**Windows**
BioRS	5.2, 5.3	Ninguno	S	S
DB2 para Linux, UNIX y Windows	8.1.x, 8.2.x, 9.1, 9.5, 9.7	Ninguna	S	S
DB2 para z/OS	7.x, 8.x, 9.x	DB2 Connect V9.7	S	S
DB2 para System i	5.2, 5.3, 5.4, 6.1	DB2 Connect V9.7	S	S
DB2 Server for VSE and VM	7.2 , 7.4	DB2 Connect V9.7	S	S
Archivos planos		Ninguna	S	S
Informix	Informix XPS 8.50, 8.51 e Informix IDS IDS 7.31, IDS 9.40, IDS 10.0, 11.5, 11.10	Informix Client SDK versión 2.81.TC2 o posteriores; 3.0 requerida para SLES 10 en Power	S	S
JDBC	3.0 o posteriores	Controladores JDBC conformes con JDBC 3.0 o posteriores	S	S
Microsoft Excel	2000, 2002, 2003, 2007	Ninguna		S

Tabla 5. Fuentes de datos soportadas, requisitos del software de cliente y soporte de sistemas operativos desde 32 bits. (continuación)

			Sistema operativo y arquitectura de hardware de 32 bits	
			X86-32	X86-32
Fuente de datos	**Versiones que reciben soporte**	**Software de cliente**	**Linux, RedHat Enterprise Linux (RHEL), SUSE**	**Windows**
Microsoft SQL Server	Microsoft SQL Server 2000/SP4, 2005, 2008	Para Windows, controlador de Microsoft SQL Server Client ODBC 3.0 (o posteriores). Para Unix, DataDirect ODBC 5.3	S	S
MQ	MQ7	MQ7	S	S
ODBC	3.0	Controladores ODBC conformes con ODBC 3.0, o posteriores**	S	S
OLE DB	2.7, 2.8	OLE DB 2.0 o posteriores	S	S
Oracle	10g, 10gR2, 11g, 11gR1	Cliente de Oracle NET 10.0 - 10.1, 10.2.0.1 con parche 3807408, 10.1.0.3 con parche 3807408, 11.1.0.6.0	S	S
Sybase	ASE 12.5, 15.0 de Sybase	Open Client 12.5 - 15.0 de Sybase	S	S
Teradata	2.5, 2.6, 12	Shared Common Components for Internationalization for Teradata (tdicu) versión 1.01 o posteriores, Teradata Generic Security Services (TeraGSS) versión 6.01 o posteriores y el software en los sistemas operativos siguientes: Para Windows el cliente Teradata TTU 8.0 o posterior y la biblioteca API de Teradata CLIv2 4.8.0 o posterior Para UNIX y Linux Teradata Call-Level Interface Versión 2 CLIv2 Release 4.8.0 o posterior	S	S

Tabla 5. Fuentes de datos soportadas, requisitos del software de cliente y soporte de sistemas operativos desde 32 bits. (continuación)

			Sistema operativo y arquitectura de hardware de 32 bits	
			X86-32	X86-32
Fuente de datos	**Versiones que reciben soporte**	**Software de cliente**	**Linux, RedHat Enterprise Linux (RHEL), SUSE**	**Windows**
Servicios Web	WSDL 1.0, 1.1 SOAP 1.0, 1.1	Ninguna	S	S
XML	XML1.0, XML1.1	Ninguna	S	S

** ODBC se puede utilizar para acceder a RedBrick 6.20cu5 e InfoSphere Classic Federation V8.2 y superiores, entre otras fuentes de datos.

Tabla 6. Soporte desde sistemas operativos de 64 bits.

Arquitectura de hardware de 64 bits	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries**
Sistema operativo	**Linux RHEL SUSE**	**Windows**	**AIX**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
Fuente de datos							
BioRS	S	S	S	S	S	S	S
DB2 para Linux, UNIX y Windows	S	S	S	S	S	S	S
DB2 para z/OS	S	S	S	S	S	S	S
DB2 para System i	S	S	S	S	S	S	S
DB2 Server for VSE and VM	S	S	S	S	S	S	S
Informix	S		S	S	S	S	S
JDBC	S	S	S	S	S	S	S
Microsoft Excel							
Microsoft SQL Server	S	S	S	S		S	
MQ	S	N	S	S	S	S	S
ODBC	S	S	S***	S		S***	S
OLE DB		S		S			
Oracle	S	S	S	S	S	S	S
Script	S	S	S	S	S	S	S
Sybase	S		S	S	S	S	
Teradata	S		S	S		S	
Servicios Web	S	S	S	S	S	S	S

Tabla 6. Soporte desde sistemas operativos de 64 bits. (continuación)

Arquitectura de hardware de 64 bits	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries**
Sistema operativo	**Linux RHEL SUSE**	**Windows**	**AIX**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
Fuente de datos							
XML	S	S	S	S	S	S	S

*** ODBC se puede utilizar para acceder a RedBrick 6.20cu5 y IBM InfoSphere Classic Federation Server for z/OS utilizando clientes de 32 y 64 bits.

Catálogo del sistema de bases de datos federadas

El catálogo del sistema de bases de datos federadas contiene información acerca de los objetos de la base de datos federada e información acerca de los objetos de las fuentes de datos.

En una base de datos federada, el catálogo se denomina catálogo global, pues contiene información acerca de todo el sistema federado. El optimizador de consultas de DB2 utiliza la información del catálogo global y del derivador de fuente de datos para planificar la mejor manera de procesar sentencias de SQL. La información almacenada en el catálogo global incluye información remota y local, como por ejemplo nombres de columna, tipos de datos de columna, valores por omisión de columna, información de índice e información de estadísticas.

La información de catálogo remota es la información o el nombre que utiliza la fuente de datos. La información de catálogo local es la información o el nombre que utiliza la base de datos federada. Por ejemplo, imaginemos que una tabla remota incluye una columna cuyo nombre es *EMPNO*. El catálogo global almacenaría el nombre de la columna remota como *EMPNO*. A menos que designe un nombre distinto, el nombre de la columna local se almacenará como *EMPNO*. Puede cambiar el nombre de la columna local por *Empleado_número*. Los usuarios que envíen consultas que incluyan esta columna utilizarán *Empleado_número* en sus consultas en lugar de *EMPNO*. Utilice la sentencia ALTER NICKNAME para cambiar el nombre local de las columnas de la fuente de datos.

Para fuentes de datos relacionales y no relacionales, la información almacenada en el catálogo global incluye tanto información remota como local.

Para ver la información de tabla de fuente de datos que está almacenada en el catálogo global, consulte las vistas de catálogo SYSCAT.TABLES, SYSCAT.NICKNAMES, SYSCAT.TABOPTIONS, SYSCAT.INDEXES, SYSCAT.INDEXOPTIONS, SYSCAT.COLUMNS y SYSCAT.COLOPTIONS en la base de datos federada.

El catálogo global también incluye información acerca de las fuentes de datos. Por ejemplo, el catálogo global incluye información que el servidor federado utiliza para conectarse con la fuente de datos y para correlacionar las autorizaciones de los usuarios federados con las autorizaciones de los usuarios de la fuente de datos. El catálogo global contiene atributos acerca de la fuente de datos que el usuario establece explícitamente, como por ejemplo, las opciones de servidor.

Optimizador de consultas

Como parte del proceso del compilado de SQL, el optimizador de consultas analiza una consulta. El compilador desarrolla estrategias alternativas, llamadas planes de acceso, para procesar la consulta.

Los planes de acceso pueden llamar a la consulta para que ésta:

- La procesen las fuentes de datos.
- La procese el servidor federado.
- La procesen en parte las fuentes de datos y en parte el servidor federado.

El optimizador de consultas evalúa los planes de acceso principalmente en base a la información sobre las capacidades y los datos de la base de datos. El derivador y el catálogo global contienen esta información. El optimizador de consultas descompone la consulta en segmentos que se llaman fragmentos de consulta. Por lo general, es más eficaz enviar un fragmento de consulta a una fuente de datos, si ésta puede procesar el fragmento. Sin embargo, el optimizador de consultas tiene en cuenta otros factores, tales como:

- El volumen de datos que se deben procesar.
- La velocidad de proceso de la fuente de datos.
- El volumen de datos que devolverá el fragmento.
- El ancho de banda de las comunicaciones.
- El hecho de si en el servidor federado existe o no una tabla de consulta materializada utilizable que represente el mismo resultado de consulta.

El optimizador de consultas genera alternativas de plan de acceso para procesar un fragmento de consulta. Las alternativas de plan realizan cantidades variables de trabajo localmente en el servidor federado y en las fuentes de datos remotas. Puesto que el optimizador de consultas está basado en los costes, asigna costes de consumo de recursos a las alternativas de plan de acceso. A continuación, el optimizador de consultas elige el plan que mejor procesará la consulta con el menor consumo de recursos.

Si cualquiera de los fragmentos se van a procesar mediante fuentes de datos, la base de datos federada envía estos fragmentos a las fuentes de datos. Una vez las fuentes de datos procesan los fragmentos, los resultados se recuperan y se devuelven a la base de datos federada. Si la base de datos federada ha realizado cualquier parte del proceso, éste combina sus resultados con los resultados recuperados desde la fuente de datos. La base de datos federada, a continuación, devuelve todos los resultados al cliente.

Secuencias de clasificación

El orden en el que los datos de carácter se almacenan en una base de datos depende de la estructura de los datos y de la secuencia de clasificación definida para la base de datos.

Suponga que los datos de una base de datos están todos en letras mayúsculas y no contienen caracteres numéricos o especiales. Una clasificación de los datos debería dar como resultado la misma salida, independientemente de si los datos están almacenados en la fuente de datos o en la base de datos federada. La secuencia de clasificación utilizada por cada base de datos no debería impactar los resultados de clasificación. De la misma manera, si los datos de la base de datos están todos en letras minúsculas o son todos caracteres numéricos, una clasificación de los datos debería generar los mismos resultados independientemente de si la clasificación se efectúa realmente.

Si los datos consisten en cualquiera de las siguientes estructuras:

- Una combinación de letras y caracteres numéricos
- Letras tanto mayúsculas como minúsculas
- Caracteres especiales tales como @, #, €

La clasificación de estos datos puede dar como resultado diferentes salidas, si la base de datos federada y la fuente de datos utilizan diferentes secuencias de clasificación.

En términos generales, una secuencia de clasificación es un orden definido para datos de carácter que determina si un carácter en particular se clasifica por encima, por debajo o al mismo nivel que otro carácter.

Cómo las secuencias de clasificación determinan los órdenes de clasificación

Una secuencia de clasificación determina el orden de clasificación de los caracteres en un conjunto de caracteres codificado.

Un conjunto de caracteres es el agregado de caracteres que se utilizan en un sistema informático o lenguaje de programación. En un conjunto de caracteres codificado, cada carácter se asigna a un número diferente dentro del rango de 0 a 255 (o el equivalente hexadecimal del mismo). Los números reciben el nombre de elemento de código; las asignaciones de números a caracteres en un conjunto se denominan colectivamente una página de códigos.

Además de asignarse a un carácter, un elemento de código se puede correlacionar con la posición del carácter en un orden de clasificación. En términos técnicos, por tanto, una secuencia de clasificación en la correlación colectiva de los elementos de código de un conjunto de caracteres a las posiciones de orden de clasificación de los caracteres del conjunto. La posición de un carácter se representa mediante un número; este número se denomina el peso del carácter. En la secuencia de clasificación más simple, llamada una secuencia de identidad, los pesos son idénticos a los elementos de código.

Ejemplo: La base de datos ALPHA utiliza la secuencia de clasificación por omisión de la página de códigos EBCDIC. La base de datos BETA utiliza la secuencia de clasificación por omisión de la página de códigos ASCII. Los órdenes de clasificación para series de caracteres en estas dos bases de datos diferirían:

```
SELECT.....
  ORDER BY COL2

Clasif. basada       Clasif. basada
en EBCDIC            en ASCII

COL2                     COL2
----                        ----
V1G                         7AB
Y2W                         V1G
7AB                         Y2W
```

Ejemplo: De manera similar, las comparaciones de caracteres en una base de datos dependen de la secuencia de clasificación definida para esa base de datos. La base de datos ALPHA utiliza la secuencia de clasificación por omisión de la página de códigos EBCDIC. La base de datos BETA utiliza la secuencia de clasificación por omisión de la página de códigos ASCII. Las comparaciones de caracteres en estas dos bases de datos generarían resultados distintos:

```
SELECT.....
  WHERE COL2 > 'TT3'

Result. basados      Result. basados
en EBCDIC            en ASCII

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

Establecimiento de la secuencia de clasificación para optimizar consultas

Los administradores pueden crear bases de datos federadas con una secuencia de clasificación determinada que coincida con la secuencia de clasificación de una fuente de datos.

Para cada definición de servidor de fuente de datos, la opción de servidor COLLATING_SEQUENCE se establece en 'Y'. Este valor le indica a la base de datos federada que las secuencias de clasificación de la base de datos federada y de la fuente de datos coinciden.

La secuencia de clasificación de la base de datos federada se establece como parte del mandato CREATE DATABASE. Con este mandato, puede especificar una de las siguientes secuencias:

- Una secuencia de identidad
- Una secuencia de sistema (la secuencia utilizada por el sistema operativo que da soporte a la base de datos)
- Una secuencia personalizada (una secuencia predefinida que el sistema de bases de datos DB2 suministra o que el usuario define)

Supongamos que la fuente de datos es DB2 para z/OS. Las clasificaciones que se definen en una cláusula ORDER BY las implementa una secuencia de clasificación basada en una página de códigos EBCDIC. Para recuperar datos de DB2 para z/OS clasificados de acuerdo con cláusulas ORDER BY, configure la base de datos federada de manera que utilice la secuencia de clasificación predefinida basada en la página de códigos EBCDIC adecuada.

Capítulo 2. Elementos de idioma

Caracteres

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

Una *letra* es cualquiera de las 26 letras mayúsculas (A - Z) o 26 letras minúsculas (a - z). Las letras también incluyen tres elementos de código reservados como expansores alfabéticos para idiomas nacionales (#, @ y $ en Estados Unidos). Sin embargo, estos tres elementos de código deben evitarse, especialmente para aplicaciones portátiles, porque representan caracteres distintos en función del CCSID. Las letras también incluyen los caracteres alfabéticos de los juegos de caracteres ampliados. Los juegos de caracteres ampliados contienen caracteres alfabéticos adicionales, tales como caracteres con signos diacríticos (´ es un ejemplo de signo diacrítico). Los caracteres disponibles dependen de la página de códigos que se utiliza.

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

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

Carácter	Descripción	Carácter	Descripción
	espacio o blanco	-	signo menos
″	apóstrofo, comillas simples o comillas dobles	.	punto
%	porcentaje	/	barra inclinada
&	signo &	:	dos puntos
’	apóstrofo o comillas simples	;	punto y coma
(	abrir paréntesis	<	menor que
)	cerrar paréntesis	=	igual
*	asterisco	>	mayor que
+	signo más	?	signo de interrogación
,	coma	_	subrayado
|	barra vertical[1]	^	signo de intercalación
!	signo de admiración	[	abrir corchete
{	abrir llave	]	cerrar corchete
}	cerrar llave	\	barra inclinada invertida[2]

[1] El uso del carácter de barra vertical (|) puede inhibir la portabilidad de código entre los productos relacionales de IBM. Debe utilizarse el operador CONCAT en lugar del operador ||.

[2] Algunas páginas de códigos no tienen ningún elemento de código para el carácter de barra inclinada invertida (\). Al entrar constantes de tipo serie de Unicode, se puede utilizar la cláusula UESCAPE para especificar un carácter de escape Unicode distinto de la barra inclinada invertida.

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

Símbolos

Los símbolos son las unidades sintácticas básicas de SQL. Un *símbolo* es una secuencia de uno o varios caracteres. Un símbolo no puede contener caracteres en blanco, a menos que sea una constante de tipo serie o un identificador delimitado, que pueden contener blancos.

Los símbolos se clasifican en ordinarios y delimitadores:

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

  Ejemplos

  ```
  1       .1       +2       SELECT       E       3
  ```

- Un *símbolo delimitador* es una constante de tipo serie, un identificador delimitado, un símbolo de operador o cualquier carácter especial mostrado en los diagramas de sintaxis. Un signo de interrogación también es un símbolo delimitador cuando actúa como marcador de parámetro.

  Ejemplos

  ```
  ,       'serie'       "fld1"       =       .
  ```

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

Comentarios: Los comentarios de SQL son compuestos (empiezan por /* y finalizan por */) o simples (empiezan por dos guiones consecutivos y finalizan al final de la línea). Las sentencias de SQL estático pueden incluir comentarios SQL o del lenguaje principal. Se pueden especificar comentarios dondequiera que se pueda especificar un espacio, excepto dentro de un símbolo delimitador o entre las palabras clave EXEC y SQL.

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

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

después de la conversión, es equivalente a:

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

Las letras alfabéticas de múltiples bytes no se convierten a mayúsculas. Los caracteres de un solo byte (de la ″a″ a la ″z″) *sí* se convierten a mayúsculas.

Para caracteres en Unicode:

- Un carácter se convierte a mayúsculas, si procede, si el carácter en mayúsculas en UTF-8 tiene la misma longitud que el carácter en minúsculas en UTF-8. Por ejemplo, el carácter 'i' sin punto y en minúscula del turco no se convierte porque en UTF-8 dicho carácter tiene l valor X'C4B1', mientras que el carácter 'I' sin punto en mayúscula tiene el valor X'49'.

- La conversión se realiza sin tener en cuenta el entorno local. Por ejemplo, el carácter 'i' sin punto en minúscula se convierte en el carácter 'I? sin punto en mayúscula del inglés.
- Amas letras alfabéticas de anchura media y anchura completa se convierten en minúscula. Por ejemplo, la 'a' minúscula de anchura completa (U+FF41) se convierte en la 'A' mayúscula de anchura completa (U+FF21).

Identificadores

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

- identificadores de SQL

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

  – Un *identificador ordinario* es una letra mayúscula seguida por cero o más caracteres, cada uno de los cuales es una letra en mayúscula, un dígito o el carácter de subrayado. Tenga en cuenta que los identificadores ordinarios no se convierten a mayúsculas. Un identificador ordinario no debe ser una palabra reservada.

    Ejemplos

    ```
    WKLYSAL     WKLY_SAL
    ```

  – Un *identificador delimitado* es una secuencia de uno o varios caracteres entre comillas dobles. Dos comillas consecutivas se utilizan para representar unas comillas dentro del identificador delimitado. De esta manera un identificador puede incluir letras en minúsculas.

    Ejemplos

    ```
    "WKLY_SAL"     "WKLY SAL"     "UNION"     "wkly_sal"
    ```

  La conversión de caracteres de los identificadores creados en una página de códigos de doble byte pero utilizados por una aplicación o una base de datos con página de códigos de múltiples bytes puede necesitar una consideración especial: tras la conversión, dichos identificadores pueden superar el límite de longitud de un identificador.

- Identificadores del lenguaje principal

  Un *identificador del lenguaje principal* es un nombre declarado en el programa de lenguaje principal. Las normas para formar un identificador de lenguaje principal son las normas del lenguaje principal. Un identificador de sistema principal no debería tener más de 255 bytes de longitud y no debería empezar por SQL o DB2 (en mayúscula o en minúscula).

Convenios de denominación y calificaciones de nombre de objeto implícitas

Las reglas para formar un nombre de objeto de base de datos dependen del tipo del objeto designado por el nombre. Los nombres pueden estar formados por un identificador de SQL único o pueden calificarse con uno o varios identificadores que identifican con más detalle al objeto. Los identificadores deben estar separados por un punto.

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

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

Los diagramas de sintaxis utilizan distintos términos para tipos diferentes de nombres. La lista siguiente define dichos términos.

nombre-alias
Nombre calificado mediante esquema que designa un alias.

nombre-atributo
Identificador que designa un atributo de un tipo de datos estructurados.

nombre-autorización
Identificador que designa un usuario, grupo o rol. Para un usuario o un grupo:

- Los caracteres válidos son: 'A'-'Z'; 'a'-'z'; '0'-'9'; '#'; '@'; '$'; '_'; '!'; ' '('; ')'; '{'; '}'; '-'; '.'y '^'.
- Los caracteres siguientes deben delimitarse entre comillas si se especifican en el procesador de línea de mandatos: '!'; ' '('; ')'; '{'; '}'; '-'; '.'y '^'.
- El nombre no puede empezar por los caracteres 'SYS', 'IBM' ni 'SQL'.
- El nombre no puede ser: 'ADMINS', 'GUESTS', 'LOCAL', 'PUBLIC' ni 'USERS'.
- Un ID de autorización delimitado no debe contener letras en minúsculas.

nombre-agrup-almac-interm
Identificador que designa una agrupación de almacenamientos intermedios.

nombre-columna
Nombre calificado o no calificado que designa una columna de una tabla o de una vista. El calificador es un nombre de tabla, un nombre de vista, un apodo o un nombre de correlación.

nombre-componente
Identificador que designa un componente de etiqueta de seguridad.

nombre-condición
Nombre calificado o no calificado que designa una condición. Un nombre de condición no calificado en una sentencia de SQL se califica implícitamente, según el contexto. Si la condición está definida en un módulo y se utiliza fuera de éste, debe estar calificada por el nombre-módulo.

nombre-restricción
Identificador que designa una restricción de referencia, una restricción de clave primaria, una restricción de unicidad o una restricción de comprobación de tabla.

nombre-correlación
Identificador que designa una tabla resultante.

nombre-cursor
Identificador que designa un cursor de SQL. Para compatibilidad entre sistemas principales, se puede utilizar un carácter de guión en el nombre.

nombre-tipo-cursor
Nombre calificado o no calificado que designa un tipo de cursor definido por el usuario. Un nombre-tipo-cursor no calificado de una sentencia de SQL se califica implícitamente, según el contexto.

nombre-variable-cursor
Nombre, calificado o no calificado, que designa una variable global, una

variable local o un parámetro de SQL de tipo cursor. Un nombre de variable de cursor no calificado en una sentencia de SQL se califica implícitamente, según el contexto.

nombre-fuente-datos
Identificador que designa una fuente de datos. Este identificador es la primera de las tres partes de un nombre de objeto remoto.

nombre-grupo-particiones-bd
Identificador que designa un grupo de particiones de base de datos.

nombre-descriptor
Dos puntos seguidos de un identificador del lenguaje principal que designa un área de descriptores de SQL (SQLDA). Para ver la descripción de un identificador de lenguaje principal, consulte el apartado "Referencias a variables del lenguaje principal" en la página 77. Observe que un nombre de descriptor nunca incluye una variable indicadora.

nombre-tipo-diferenciado
Nombre calificado o no calificado que designa un tipo diferenciado. El formato no calificado de un nombre-tipo-diferenciado es un identificador SQL. Un nombre de tipo diferenciado no calificado de una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema o un nombre de módulo, que se determina mediante el contexto en el que aparece el nombre-tipo-diferenciado. El formato calificado es un nombre de esquema seguido de un punto y de un identificador de SQL o un nombre de módulo (que también puede calificarse mediante un nombre-esquema) seguido de un punto y de un identificador de SQL. Si el tipo diferenciado está definido en un módulo y se utiliza fuera del mismo módulo, debe estar calificado por el nombre-módulo.

nombre-supervisor-sucesos
Identificador que designa un supervisor de sucesos.

nombre-correlación-funciones
Identificador que designa una correlación de funciones.

nombre-función
Nombre calificado o no calificado que designa una función. El formato no calificado de un nombre-función es un identificador de SQL. Un nombre de función no calificado de una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema, que se determina mediante el contexto en el que aparece la función. El formato calificado puede ser un nombre de esquema seguido de un punto y de un identificador SQL o un nombre de módulo seguido de un punto y de un identificador SQL. Si la función se publica en un módulo y se utiliza fuera del mismo módulo, debe estar calificada por el nombre-módulo.

nombre-variable-global
Nombre, calificado o no calificado, que designa una variable global. Un nombre de variable global no calificado en una sentencia de SQL se califica implícitamente, según el contexto. Si la variable global está definida en un módulo y se utiliza fuera del mismo módulo, debe calificarse mediante el nombre-módulo.

nombre-grupo
Identificador no calificado que designa un grupo de transformación definido para un tipo estructurado.

variable-lenguaje-principal
Secuencia de símbolos que designa una variable del lenguaje principal.

Una variable del lenguaje principal incluye, como mínimo, un identificador de lenguaje principal, como se explica en el apartado “Referencias a variables del lenguaje principal” en la página 77.

nombre-índice
Nombre calificado mediante esquema que designa un índice o una especificación de índice.

etiqueta
Identificador que designa una etiqueta en un procedimiento de SQL.

nombre-método
Identificador que designa un método. El contexto de esquema de un método está determinado por el esquema del tipo indicado (o de un supertipo del tipo indicado) del método.

nombre-módulo
Nombre calificado o no calificado que designa un módulo. Un nombre-módulo no calificado de una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema, que se determina mediante el contexto en el que aparece el nombre-módulo. La forma calificada es un nombre de esquema seguido de un punto y un identificador de SQL.

apodo Nombre calificado mediante esquema que designa una referencia de servidor federado a una tabla o vista.

nombre-paquete
Nombre calificado mediante esquema que designa un paquete. Si un paquete tiene un ID de versión que no es la serie vacía, el nombre del paquete también incluye el ID de versión al final del nombre, en el formato siguiente: `id-esquema.id-paquete.id-versión`.

nombre-parámetro
Identificador que designa un parámetro al que se puede hacer referencia en un procedimiento, una función definida por el usuario, un método o una extensión de índice.

nombre-partición
Identificador que designa una partición de datos en una tabla particionada.

nombre-procedimiento
Nombre calificado o no calificado que designa un procedimiento. El formato no calificado de un nombre-procedimiento es un identificador de SQL. Un nombre de procedimiento no calificado de una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema, que se determina mediante el contexto en el que aparece el procedimiento. El formato calificado es un nombre de esquema seguido de un punto y de un identificador SQL o un nombre de módulo seguido de un punto y de un identificador SQL. Si el procedimiento está definido en un módulo y se utiliza fuera de éste, debe estar calificado por el nombre-módulo.

nombre-autorización-remota
Identificador que designa un usuario de fuente de datos. Las normas para los nombres de autorización varían de fuente de datos a fuente de datos.

nombre-función-remota
Nombre que designa una función registrada en una base de datos de fuente de datos.

nombre-objeto-remoto
Nombre de tres partes que designa una tabla de fuente de datos o vista y que identifica la fuente de datos en la que reside la tabla o la vista. Las partes de este nombre son nombre-fuente-datos, nombre-esquema-remoto y nombre-tabla-remota.

nombre-esquema-remoto
Nombre que designa el esquema al que pertenece una tabla de fuente de datos o vista. Este nombre es la segunda de las tres partes de un nombre de objeto remoto.

nombre-tabla-remota
Nombre que designa una tabla o una vista en una fuente de datos. Este nombre es la tercera de las tres partes de un nombre de objeto remoto.

nombre-tipo-remoto
Tipo de datos soportado por una base de datos de fuente de datos. No utilice el formato largo para los tipos internos (utilice CHAR en vez de CHARACTER, por ejemplo).

nombre-rol
Identificador que designa un rol.

nombre-tipo-fila
Nombre calificado o no calificado que designa un tipo de fila. El formato no calificado de un nombre-tipo-fila es un identificador SQL. Un nombre-tipo-fila no calificado en una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema, que determina el contexto en el que aparece el nombre de tipo de fila, tal como describen las reglas de "Nombres no calificados de función, procedimiento, específico, variable global, módulo y tipo definido por el usuario" en la página 85. La forma calificada es un nombre de esquema seguido de un punto y un identificador de SQL.

nombre-puntosalvaguarda
Identificador que designa un punto de salvar.

nombre-esquema
Identificador que proporciona una agrupación lógica de objetos de SQL. Un nombre de esquema que se utiliza como calificador del nombre de un objeto puede determinarse implícitamente:

- a partir del valor del registro especial CURRENT SCHEMA
- a partir del valor de la opción de precompilación/vinculación QUALIFIER
- sobre la base de un algoritmo de resolución que utilice el registro especial CURRENT PATH
- sobre la base del nombre de esquema de otro objeto en la misma sentencia de SQL.

Para evitar complicaciones, es recomendable no utilizar el nombre SESSION como esquema, excepto para el esquema de tablas temporales globales declaradas (las cuales *deben* utilizar el nombre de esquema SESSION).

nombre-etiqueta-seguridad
Nombre calificado o no calificado que designa una etiqueta de seguridad. El nombre-política-seguridad aplicable califica implícitamente un nombre de etiqueta de seguridad no calificado en una sentencia de SQL, en caso de

aplicarse. Si no hay ningún nombre-política-seguridad aplicable implícitamente, el nombre debe ser calificado.

nombre-política-seguridad
Identificador que designa una política de seguridad.

nombre-secuencia
Identificador que designa una secuencia.

nombre-servidor
Identificador que designa un servidor de aplicaciones. En un sistema federado, el nombre de servidor también designa el nombre local de una fuente de datos.

nombre-específico
Nombre, calificado o no calificado, que designa un nombre específico. Un nombre específico no calificado en una sentencia de SQL se califica implícitamente, según el contexto.

nombre-variable-SQL
Nombre de una variable local en una sentencia de procedimiento de SQL. Los nombres de variables SQL se pueden utilizar en otras sentencias de SQL donde esté permitido un nombre de variable del lenguaje principal. El nombre puede estar calificado por la etiqueta de la sentencia compuesta donde se declaró la variable de SQL.

nombre-sentencia
Identificador que designa una sentencia de SQL preparada.

nombre-supertipo
Nombre calificado o no calificado que designa el supertipo de un tipo. Un nombre de supertipo no calificado en una sentencia de SQL se califica implícitamente, según el contexto.

nombre-tabla
Nombre calificado mediante esquema que designa una tabla.

referencia-tabla
Nombre calificado o no calificado que designa una tabla. El esquema de tabla califica implícitamente una referencia de tabla no calificada en una expresión.

nombre-espacio-tablas
Identificador que designa un espacio de tablas.

nombre-activador
Nombre calificado mediante esquema que designa un activador.

nombre-correlación-tipos
Identificador que designa una correlación de tipos de datos.

nombre-tipo
Nombre calificado o no calificado que designa un tipo. Un nombre de tipo no calificado en una sentencia de SQL se califica implícitamente, según el contexto.

nombre-tabla-tipo
Nombre calificado mediante esquema que designa una tabla con tipo.

nombre-vista-tipo
Nombre calificado mediante esquema que designa una vista con tipo.

nombre-tipo-definido-por-usuario
Nombre calificado o no calificado que designa un tipo de datos definido

por el usuario. El formato no calificado de un nombre-tipo-definido-por-usuario es un identificador SQL. Un nombre-tipo-definido-por-usuario no calificado en una sentencia de SQL se califica implícitamente. El calificador implícito es un nombre de esquema o un nombre de módulo, que se determina mediante el contexto en el que aparece el nombre-tipo-definido-por-usuario. El formato calificado es un nombre de esquema seguido de un punto y de un identificador de SQL o un nombre de módulo (que también puede calificarse mediante un nombre-esquema) seguido de un punto y de un identificador de SQL. Si el tipo de datos definido por el usuario está definido en un módulo y se utiliza fuera de éste, debe estar calificado por el nombre-módulo.

nombre-vista
Nombre calificado mediante esquema que designa una vista.

nombre-derivador
Identificador que designa un derivador.

esquema-XML-nombre
Nombre calificado o no calificado que designa un esquema XML.

nombre-objetoxsr
Nombre calificado o no calificado que designa un objeto en el repositorio de esquemas XML.

Alias para los objetos de la base de datos

Se puede considerar que un alias es un nombre alternativo para un objeto SQL. Por consiguiente, puede hacerse referencia a un objeto SQL en una sentencia de SQL utilizando su nombre o su alias.

Un alias público es un nombre al que se puede hacer referencia siempre sin calificar su nombre con un nombre de esquema. El calificador implícito de un alias público es SYSPUBLIC, que también se puede especificar explícitamente.

Los alias también se conocen con el nombre de sinónimos.

Un alias se puede utilizar donde esté basado el objeto. Se puede crear un alias aunque no exista el objeto (aunque debe existir en el momento de compilar una sentencia que hace referencia al mismo). Puede hacer referencia a otro alias si no se realizan referencias circulares ni repetitivas a lo largo de la cadena de alias. Un alias sólo puede hacer referencia a un módulo, apodo, secuencia, tabla, vista u otro alias dentro de la misma base de datos. No se puede utilizar un nombre de alias donde se espere un nombre de objeto nuevo como ocurre en las sentencias CREATE TABLE o CREATE VIEW; por ejemplo, si se ha creado el nombre de alias de tabla PERSONNEL, las sentencias posteriores como CREATE TABLE PERSONNEL... devolverán un error.

La opción de hacer referencia a un objeto mediante un alias no se muestra explícitamente en los diagramas de sintaxis, ni se menciona en las descripciones de sentencias de SQL.

Un alias no calificado nuevo de un tipo de objeto determinado, por ejemplo, para una secuencia, no puede tener el mismo nombre completamente calificado que un objeto ya existente de dicho tipo de objeto. Por ejemplo, un alias de secuencia llamado ORDERID no se puede definir en el esquema KANDIL para el nombre de secuencia KANDIL ORDERID.

El efecto de utilizar un alias en una sentencia de SQL es similar al de la sustitución de texto. El alias, que debe estar definido cuando se compila la sentencia de SQL, se sustituye en el momento de compilación de la sentencia por el nombre del objeto calificado. Por ejemplo, si PBIRD.SALES es un alias de DSPN014.DIST4_SALES_148, entonces en el momento de la compilación:

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

se convierte en realidad en

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

ID de autorización y nombres de autorización

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

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

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

Se aplica un ID de autorización a cada sentencia de SQL. El ID de autorización que se aplica a una sentencia de SQL estático es el ID de autorización que se utiliza durante la vinculación de programas. El ID de autorización correspondiente a una sentencia de SQL dinámico se basa en la opción DYNAMICRULES proporcionada durante el momento de la vinculación y en el entorno actual de ejecución del paquete que emite la sentencia de SQL dinámico:

- En un paquete que tenga un comportamiento de vinculación, el ID de autorización utilizado es el ID de autorización del propietario del paquete.
- En un paquete que tenga un comportamiento de definición, el ID de autorización utilizado es el ID de autorización correspondiente a la persona que define la rutina.
- En un paquete que tenga un comportamiento de ejecución, el ID de autorización utilizado es el ID de autorización actual del usuario que ejecute el paquete.
- En un paquete que tenga un comportamiento de invocación, el ID de autorización utilizado es el ID de autorización actualmente en vigor al invocar la rutina. Este ID se denomina ID de autorización de ejecución.

Para obtener más información, consulte el apartado “Características de SQL dinámico durante la ejecución” en la página 67.

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

Ejemplos:

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

  ```
  GRANT SELECT ON TDEPT TO KEENE
  ```

  SMITH es el ID de autorización de la sentencia. Por lo tanto, en una sentencia de SQL dinámico, el valor por omisión del registro especial CURRENT SCHEMA será SMITH y, en SQL estático, el valor por omisión de la opción de precompilación/vinculación QUALIFIER será SMITH. La autorización para ejecutar la sentencia se compara con SMITH y SMITH es el calificador implícito de *nombre-tabla* de acuerdo con las normas de calificación descritas en el apartado "Convenios de denominación y calificaciones de nombre de objeto implícitas" en la página 59.

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

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

  ```
  DROP TABLE TDEPT
  ```

  Elimina la tabla SMITH.TDEPT.

  ```
  DROP TABLE SMITH.TDEPT
  ```

  Elimina la tabla SMITH.TDEPT.

  ```
  DROP TABLE KEENE.TDEPT
  ```

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

  ```
  CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
  ```

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

Características de SQL dinámico durante la ejecución

La opción DYNAMICRULES BIND determina el ID de autorización que se utiliza para comprobar la autorización cuando se procesan sentencias de SQL dinámico. Además, la opción también controla otros atributos de SQL dinámico como, por ejemplo, el calificador implícito que se utiliza para las referencias a objetos no calificadas y si es posible invocar dinámicamente ciertas sentencias de SQL.

El conjunto de valores para el ID de autorización y otros atributos de SQL dinámico se denomina el comportamiento de las sentencias de SQL dinámico. Los cuatro comportamientos posibles son ejecución, vinculación, definición e invocación. Tal como se muestra en la tabla siguiente, la combinación del valor de la opción DYNAMICRULES BIND y el entorno de ejecución determina el comportamiento que se utiliza. Es valor por omisión es DYNAMICRULES RUN, que implica un comportamiento de ejecución.

Tabla 7. Forma en que DYNAMICRULES y el entorno de ejecución determinan el comportamiento de las sentencias de SQL dinámico

	Comportamiento de las sentencias de SQL dinámico	
Valor de DYNAMICRULES	**Entorno de programa autónomo**	**Entorno de rutina**
BIND	Comportamiento de vinculación	Comportamiento de vinculación
RUN	Comportamiento de ejecución	Comportamiento de ejecución
DEFINEBIND	Comportamiento de vinculación	Comportamiento de definición
DEFINERUN	Comportamiento de ejecución	Comportamiento de definición
INVOKEBIND	Comportamiento de vinculación	Comportamiento de invocación
INVOKERUN	Comportamiento de ejecución	Comportamiento de invocación

Comportamiento de ejecución
DB2 utiliza el ID de autorización del usuario (el ID que inicialmente se ha conectado a DB2) que ejecuta el paquete como el valor que debe utilizarse para la comprobación de autorización de las sentencias de SQL dinámico y para el valor inicial utilizado para la calificación implícita de referencias de objetos no calificados dentro de sentencias de SQL dinámico.

Comportamiento de vinculación
Durante la ejecución, DB2 utiliza todas las normas que se aplican a SQL estático para la autorización y la calificación. Utiliza el ID de autorización del propietario del paquete como el valor que se utilizará para la comprobación de autorización de las sentencias de SQL dinámico y el calificador por omisión del paquete para la calificación implícita de las referencias a objetos no calificadas de las sentencias de SQL dinámico.

Comportamiento de definición
El comportamiento de definición sólo se aplica si la sentencia de SQL dinámico está en un paquete que se ejecuta en un contexto de rutina y el paquete se ha vinculado con DYNAMICRULES DEFINEBIND o DYNAMICRULES DEFINERUN. DB2 utiliza el ID d autorización del definidor de rutina (no el vinculador de paquetes de la rutina) como valor que debe utilizarse para la comprobación de autorización de las sentencias de SQL dinámico, y para la cualificación implícita de referencias objetos sin cualificar dentro de sentencias de SQL dentro de dicha rutina.

Comportamiento de invocación
El comportamiento de invocación sólo se aplica si la sentencia de SQL dinámico está en un paquete que se ejecuta en un contexto de rutina y el paquete se ha vinculado con DYNAMICRULES INVOKEBIND o DYNAMICRULES INVOKERUN. DB2 utiliza el ID de autorización de sentencias vigente cuando la rutina se invoca como el valor que debe utilizarse para comprobar la autorización del SQL dinámico y para la cualificación implícita de referencias de objetos no calificados dentro de las sentencias de SQL dinámico dentro de dicha rutina. La tabla siguiente muestra un resumen.

Entorno que se invoca	ID utilizado
Cualquier SQL estático	Valor implícito o explícito del propietario (OWNER) del paquete del que procedía el SQL que invocaba la rutina
Utilizado en definiciones de vistas o activadores	Persona que define la vista o el activador
SQL dinámico de un paquete con comportamiento de vinculación	Valor implícito o explícito del propietario (OWNER) del paquete del que procedía el SQL que invocaba la rutina
SQL dinámico de un paquete con comportamiento de ejecución	ID que se utiliza para establecer la conexión inicial con DB2
SQL dinámico de un paquete con comportamiento de definición	Persona que define la rutina que utiliza el paquete del que procedía el SQL que invocaba la rutina
SQL dinámico de un paquete con comportamiento de invocación	El ID autorización actual que invoca la rutina

Sentencias restringidas cuando no se aplica el comportamiento de ejecución

Cuando está en vigor un comportamiento de vinculación, definición o invocación, no es posible utilizar las siguientes sentencias de SQL dinámico: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT, RENAME, SET INTEGRITY, SET EVENT MONITOR STATE; o consultas que hacen referencia a un apodo.

Consideraciones respecto a la opción DYNAMICRULES

No se puede utilizar el registro especial CURRENT SCHEMA para calificar las referencias a objetos no calificadas en las sentencias de SQL dinámico ejecutadas desde un paquete con comportamiento de vinculación, definición o invocación. Esto es así incluso después de emitir la sentencia SET CURRENT SCHEMA para cambiar el registro especial CURRENT SCHEMA; el valor del registro se cambia pero no se utiliza.

En caso de que se haga referencia a varios paquetes durante una sola conexión, todas las sentencias de SQL dinámico que estos paquetes hayan preparado mostrarán el comportamiento especificado por la opción DYNAMICRULES para dicho paquete en concreto y el entorno en el que se utilicen.

Es importante tener presente que, cuando un paquete muestra un comportamiento de vinculación, no debe otorgarse a la persona que vincula el paquete ninguna autorización que no se desee que tenga el usuario del paquete ya que una sentencia dinámica utilizará el ID de autorización del propietario del paquete. De forma similar, cuando un paquete muestra un comportamiento de definición, no debe otorgarse a la persona que define la rutina ninguna autorización que no se desee que tenga el usuario del paquete.

ID de autorización y preparación de sentencias

Si se especifica la opción VALIDATE BIND durante la ejecución, los privilegios necesarios para manejar tablas y vistas también deben existir durante la vinculación. Si estos privilegios o los objetos referenciados no existen y está en vigor la opción SQLERROR NOPACKAGE, la operación de vinculación no será satisfactoria. Si se especifica la opción SQLERROR CONTINUE, la operación de

vinculación será satisfactoria y se marcarán las sentencias erróneas. Si se intenta ejecutar una de estas sentencias, se producirá un error.

Si un paquete se vincula con la opción VALIDATE RUN, se realiza todo el proceso normal de vinculación, pero no es necesario que existan todavía los privilegios necesarios para utilizar las tablas y vistas referenciadas en la aplicación. Si durante la vinculación no existe un privilegio necesario, se realiza una operación de vinculación incremental cada vez que se ejecuta por primera vez la sentencia en una aplicación y deben existir todos los privilegios que la sentencia necesita. Si no existe un privilegio necesario, la ejecución de la sentencia no es satisfactoria.

La comprobación de autorización durante la ejecución se realiza utilizando el ID de autorización del propietario del paquete.

Nombres de columna

El significado de un *nombre de columna* depende de su contexto. Un nombre de columna sirve para:

- Declarar el nombre de una columna como, por ejemplo, en una sentencia CREATE TABLE.
- Identificar una columna como, por ejemplo, en una sentencia CREATE INDEX.
- Especificar los valores de la columna como, por ejemplo, en los contextos siguientes:
  - En una función agregada, un nombre de columna especifica todos los valores de la columna en la tabla de resultados intermedia o de grupo a la que se aplica la función. Por ejemplo, MAX(SALARY) aplica la función MAX a todos los valores de la columna SALARY de un grupo.
  - En una cláusula GROUP BY o ORDER BY, un nombre de columna especifica todos los valores de la tabla de resultado intermedia a los que se aplica la cláusula. Por ejemplo, ORDER BY DEPT ordena una tabla de resultado intermedia según los valores de la columna DEPT.
  - En una expresión, una condición de búsqueda o una función escalar, un nombre de columna especifica un valor para cada fila o grupo al que se aplica la construcción. Por ejemplo, cuando la condición de búsqueda CODE = 20 se aplica a alguna fila, el valor especificado por el nombre de columna CODE es el valor de la columna CODE en esa fila.
- Redenominar temporalmente una columna, como en la *cláusula-correlación* de una *referencia-tabla* en una cláusula FROM.

Nombres de columna calificados

Un calificador para un nombre de columna puede ser un nombre de tabla, vista, apodo o alias o un nombre de correlación.

El hecho de que un nombre de columna pueda calificarse depende del contexto:

- Según la forma de la sentencia COMMENT ON, puede que se deba calificar un nombre de una sola columna. No se deben calificar nombres de varias columnas.
- Donde el nombre de columna especifique valores de la columna, puede calificarse como opción del usuario.
- En la cláusula de asignación de una sentencia UPDATE, puede calificarse en la opción del usuario.
- En todos los demás contextos, un nombre de columna no debe calificarse.

Cuando un calificador es opcional, puede cumplir dos finalidades. Estos casos se describen en el apartado “Calificadores de nombres de columna para evitar ambigüedades” en la página 73 y “Calificadores de nombres de columna en referencias correlacionadas” en la página 75.

Nombres de correlación

Un *nombre de correlación* puede definirse en la cláusula FROM de una consulta y en la primera cláusula de una sentencia UPDATE o DELETE. Por ejemplo, la cláusula FROM X.MYTABLE Z establece Z como nombre de correlación para X.MYTABLE.

```
FROM X.MYTABLE Z
```

Con Z definida como nombre de correlación para X.MYTABLE, sólo puede utilizarse Z para calificar una referencia a una columna de esa instancia de X.MYTABLE en esa sentencia SELECT.

Un nombre de correlación se asocia con una tabla, una vista, un apodo, un alias, una expresión de tabla anidada, una función de tabla o una referencia de tabla de cambio de datos sólo dentro del contexto en el que se ha definido. Por lo tanto, puede definirse el mismo nombre de correlación con distintos propósitos en diferentes sentencias o bien en distintas cláusulas de la misma sentencia.

Como calificador, un nombre de correlación puede utilizarse para evitar ambigüedades o para establecer una referencia correlacionada. También se puede utilizar simplemente como un nombre abreviado de una referencia de tabla. En el ejemplo, Z podría haberse utilizado simplemente para evitar tener que entrar X.MYTABLE más de una vez.

Si se especifica un nombre de correlación para una tabla, vista, apodo o alias, cualquier referencia a una columna de esa instancia de la tabla, vista, apodo o alias debe utilizar el nombre de correlación en lugar del nombre de tabla, vista, apodo o alias. Por ejemplo, la referencia a EMPLOYEE.PROJECT del ejemplo siguiente no es correcto, porque se ha especificado un nombre de correlación para EMPLOYEE:

Ejemplo

```
FROM EMPLOYEE E
  WHERE EMPLOYEE.PROJECT='ABC'      * incorrect*
```

La referencia calificada para PROJECT debe utilizar, en su lugar, el nombre de correlación ″E″, tal como se muestra abajo:

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

Los nombres especificados en una cláusula FROM pueden estar *expuestos* o *no expuestos*. Se dice que un nombre de tabla, vista, apodo o alias está expuesto en la cláusula FROM si no se especifica un nombre de correlación. El nombre de la correlación es siempre un nombre expuesto. Por ejemplo, en la siguiente cláusula FROM, se especifica un nombre de correlación para EMPLOYEE pero no para DEPARTMENT, de modo que DEPARTMENT es un nombre expuesto y EMPLOYEE no lo es:

```
FROM EMPLOYEE E, DEPARTMENT
```

Un nombre de tabla, vista, apodo o alias que está expuesto en una cláusula FROM puede ser igual a otro nombre de tabla, vista o apodo expuesto en esa cláusula FROM o cualquier nombre de correlación de la cláusula FROM. Esta situación

puede dar como resultado una serie de referencias ambiguas de nombres de columna que acaban devolviendo un código de error (SQLSTATE 42702).

Las dos primeras cláusulas FROM mostradas más abajo son correctas, porque cada una no contiene más de una referencia a EMPLOYEE que esté expuesta:

1. Dada una cláusula FROM:

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   una referencia calificada como, por ejemplo, EMPLOYEE.PROJECT indica una columna de la segunda instancia de EMPLOYEE en la cláusula FROM. La referencia calificada a la primera instancia de EMPLOYEE debe utilizar el nombre de correlación "E1" (E1.PROJECT).
2. Dada una cláusula FROM:

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   una referencia calificada como, por ejemplo, EMPLOYEE.PROJECT indica una columna de la primera instancia de EMPLOYEE en la cláusula FROM. Una referencia calificada a la segunda instancia de EMPLOYEE debe utilizar el nombre de correlación "E2" (E2.PROJECT).
3. Dada una cláusula FROM:

   ```
   FROM EMPLOYEE, EMPLOYEE
   ```

   los dos nombres de tabla expuestos que se incluyen en esta cláusula (EMPLOYEE y EMPLOYEE) son los mismos. Esto está permitido, pero las referencias a nombres de columnas específicos resultarían ambiguas (SQLSTATE 42702).
4. Dada la sentencia siguiente:

   ```
   SELECT *
     FROM EMPLOYEE E1, EMPLOYEE E2            * incorrect *
     WHERE EMPLOYEE.PROJECT = 'ABC'
   ```

   la referencia calificada EMPLOYEE.PROJECT es incorrecta, porque las dos instancias de EMPLOYEE en la cláusula FROM tienen nombres de correlación. En cambio, las referencias a PROJECT deben estar calificadas con algún nombre de correlación (E1.PROJECT o E2.PROJECT).
5. Dada una cláusula FROM:

   ```
   FROM EMPLOYEE, X.EMPLOYEE
   ```

   una referencia a una columna en la segunda instancia de EMPLOYEE debe utilizar X.EMPLOYEE (X.EMPLOYEE.PROJECT). Si X es el valor del registro especial CURRENT SCHEMA en SQL dinámico o la opción de precompilación/vinculación QUALIFIER de SQL estático, no se puede hacer ninguna referencia a las columnas porque resultaría ambigua.

La utilización del nombre de correlación en la cláusula FROM permite, también, la opción de especificar una lista de nombres de columna que se han de asociar con las columnas de la tabla resultante. Igual que los nombres de correlación, estos nombres de columna listados se convierten en los nombres *expuestos* de las columnas que deben utilizarse en las referencias a las columnas en toda la consulta. Si se especifica una lista de nombres de columna, los nombres de columna de la tabla principal se convierten en *no expuestos*.

Dada una cláusula FROM:

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

una referencia calificada como, por ejemplo, D.NUM indica la primera columna de la tabla DEPARTMENT que se ha definido en la tabla como DEPTNO. Una referencia a D.DEPTNO utilizando esta cláusula FROM es incorrecta ya que el

nombre de columna DEPTNO es un nombre de columna no expuesto.

Calificadores de nombres de columna para evitar ambigüedades

En el contexto de una función, de una cláusula GROUP BY, de una cláusula ORDER BY, de una expresión o de una condición de búsqueda, un nombre de columna hace referencia a los valores de una columna en alguna tabla, vista, apodo, expresión de tabla anidada o función de tabla. Las tablas, vistas, apodos, expresiones de tablas anidadas y funciones de tabla donde puede residir la columna se denominan *tablas de objetos* del contexto. Dos o más tablas de objetos pueden contener columnas con el mismo nombre; un nombre de columna se puede calificar para indicar la tabla de la cual procede la columna. Los calificadores de nombres de columna también son útiles en los procedimientos de SQL para diferenciar los nombres de columna de los nombres de variables de SQL utilizados en sentencias de SQL.

Una expresión de tabla anidada o una función de tabla trata las *referencias-tabla* que la preceden en la cláusula FROM como tablas de objetos. Las *referencias-tabla* que siguen no se tratan como tablas de objetos.

Designadores de tabla

Un calificador que designa una tabla de objeto específica se conoce como *designador de tabla*. La cláusula que identifica las tablas de objetos también establece los designadores de tabla para ellas. Por ejemplo, las tablas de objetos de una expresión en una cláusula SELECT se nombran en la cláusula FROM que la sigue:

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

Los designadores en la cláusula FROM se establecen como sigue:

- Un nombre que sigue a una tabla, vista, apodo, alias, expresión de tabla anidada o función de tabla es a la vez un nombre de correlación y un designador de tabla. Así pues, CORZ es un designador de tabla. CORZ sirve para calificar el primer nombre de columna de la lista de selección.
- Una tabla expuesta, un nombre de vista, un apodo o alias es un designador de tabla. Así pues, OWNY.MYTABLE es un designador de tabla. OWNY.MYTABLE sirve para calificar el nombre de la segunda columna de la lista de selección.

Al calificar una columna con el nombre de tabla expuesto de un designador de tabla, se puede utilizar la forma calificada o no calificada del nombre expuesto de tabla. Si se utiliza la forma calificada, el calificador debe ser el mismo que el calificador por omisión del nombre expuesto de tabla.

Pongamos, por ejemplo, que el esquema actual es CORPDATA.

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

es válido porque la tabla EMPLOYEE a la que se hace referencia en la cláusula FROM califica completamente CORPDATA.EMPLOYEE, que coincide con el calificador de la columna WORKDEPT.

```
SELECT EMPLOYEE.WORKDEPT, REGEMP.WORKDEPT
  FROM CORPDATA.EMPLOYEE, REGION.EMPLOYEE REGEMP
```

también es válido, porque la primera columna de lista de selección hace referencia al designador de tabla expuesto no calificado CORPDATA.EMPLOYEE, que se encuentra en la cláusula FROM y la segunda columna de lista de selección hace

referencia al nombre de correlación REGEMP del objeto de tabla REGION.EMPLOYEE, que también se encuentra en la cláusula FROM.

Pongamos ahora que el esquema actual es REGION.

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

no es válido porque la tabla EMPLOYEE a la que se hace referencia en la cláusula FROM califica completamente REGION.EMPLOYEE y el calificador de la columna WORKDEPT representa la tabla CORPDATA.EMPLOYEE.

Cada designador de tabla debe ser exclusivo en una cláusula FROM determinada para evitar la aparición de referencias ambiguas a columnas.

Evitar referencias no definidas o ambiguas

Cuando un nombre de columna hace referencia a valores de una columna, debe existir una sola tabla de objetos que incluya una columna con ese nombre. Las situaciones siguientes se consideran errores:

- Ninguna tabla de objetos contiene una columna con el nombre especificado. La referencia no está definida.
- El nombre de columna está calificado mediante un designador de tabla, pero la tabla designada no incluye una columna con el nombre especificado. De nuevo, la referencia no está definida.
- El nombre no está calificado, y hay más de una tabla de objetos que incluye una columna con ese nombre. La referencia es ambigua.
- El designador de tabla califica al nombre de columna, pero la tabla designada no es única en la cláusula FROM y ambas apariciones de la tabla designada incluyen la columna. La referencia es ambigua.
- El nombre de columna de una expresión de tabla anidada que no va precedida por la palabra clave TABLE o en una función de tabla o expresión de tabla anidada que es el operando derecho de una unión externa derecha o una unión externa completa y el nombre de columna no hace referencia a una columna de una *referencia-tabla* de la selección completa de la expresión de tabla anidada. La referencia no está definida.

Evite las referencias ambiguas calificando un nombre de columna con un designador de tabla definido exclusivamente. Si la columna está en varias tablas de objetos con nombres distintos, los nombres de tabla pueden utilizarse como designadores. Las referencias ambiguas también se pueden evitar sin la utilización del designador de tabla dando nombres exclusivos a las columnas de una de las tablas de objetos utilizando la lista de nombres de columna que siguen al nombre de correlación.

Al calificar una columna con la forma de nombre expuesto de tabla de un designador de tabla, se puede utilizar la forma calificada o no calificada del nombre de tabla expuesto. Sin embargo, el calificador y la tabla utilizados deben ser iguales después de calificar completamente el nombre de tabla, vista o apodo y el designador de tabla.

1. Si el ID de autorización de la sentencia es CORPDATA:

   ```
   SELECT CORPDATA.EMPLOYEE.WORKDEPT
     FROM EMPLOYEE
   ```

   es una sentencia válida.
2. Si el ID de autorización de la sentencia es REGION:

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE                          * incorrect *
```

no es válido, porque EMPLOYEE representa la tabla REGION.EMPLOYEE, pero el calificador para WORKDEPT representa una tabla distinta, CORPDATA.EMPLOYEE.

Calificadores de nombres de columna en referencias correlacionadas

Una *selección completa* es una forma de consulta que puede utilizarse como componente de varias sentencias de SQL. Una selección completa utilizada en una condición de búsqueda de cualquier sentencia se denomina *subconsulta*. Una selección completa utilizada para recuperar un único valor como, por ejemplo, una expresión en una sentencia se denomina una *selección completa escalar* o *subconsulta escalar*. Una selección completa utilizada en la cláusula FROM de una consulta se denomina *expresión de tabla anidada*. Se hace referencia a las subconsultas de las condiciones de búsqueda, subconsultas escalares y expresiones de tabla anidadas como subconsultas en el resto de este tema.

Una subconsulta puede contener subconsultas propias y éstas, a su vez, pueden contener subconsultas. De este modo, una sentencia de SQL puede contener una jerarquía de subconsultas. Los elementos de la jerarquía que contienen subconsultas están en un nivel superior que las subconsultas que contienen.

Cada elemento de la jerarquía contiene uno o más designadores de tabla. Una consulta puede hacer referencia no solamente a las columnas de las tablas identificadas en su mismo nivel dentro de la jerarquía, sino también a las columnas de las tablas anteriormente identificadas en la jerarquía, hasta alcanzar el estrato más elevado. Una referencia a una columna de una tabla identificada en un nivel superior se llama *referencia correlacionada*.

Para la compatibilidad con los estándares existentes de SQL, se permiten nombres de columna calificados y no calificados como referencias correlacionadas. Sin embargo, es aconsejable calificar todas las referencias de columnas utilizadas en subconsultas; de lo contrario, los nombres de columna idénticos pueden conducir a resultados no deseados. Por ejemplo, si se modifica una tabla de una jerarquía de modo que contenga el mismo nombre de columna que la referencia correlacionada y la sentencia se vuelve a preparar, la referencia se aplicará en la tabla modificada.

Cuando se califica un nombre de columna en una subconsulta, se busca en cada nivel de jerarquía, comenzando en la misma subconsulta en la que aparece el nombre de columna calificado y continuando hacia niveles superiores de la jerarquía, hasta que se encuentre un designador de tabla que coincida con el calificador. Una vez encontrado, se verifica que la tabla contenga la columna en cuestión. Si se encuentra la tabla en un nivel superior que el nivel que contiene el nombre de columna, es que éste es una referencia correlacionada para el nivel donde se encontró el designador de tabla. Una expresión de tabla anidada debe ir precedida por la palabra clave TABLE opcional para buscar en la jerarquía superior la selección completa de la expresión de tabla anidada.

Cuando el nombre de columna de una subconsulta no se califica, se busca en las tablas a las que se hace referencia en cada nivel de la jerarquía, empezando en la misma subconsulta en la que aparece el nombre de columna y siguiendo hacia niveles superiores de la jerarquía hasta que se encuentre un nombre de columna que coincida. Si la columna se encuentra en una tabla en un nivel superior al nivel que contiene el nombre de columna, es que éste es una referencia correlacionada

para el nivel donde se ha encontrado la tabla que contiene la columna. Si se encuentra el nombre de columna en más de una tabla en un nivel en concreto, la referencia es ambigua y se considera un error.

En cualquier caso, en el siguiente ejemplo T hace referencia al designador de tabla que contiene la columna C. Un nombre de columna, T.C (donde T representa un calificador implícito o explícito), es una referencia correlacionada solamente si se dan estas condiciones:

- T.C se utiliza en una expresión de una subconsulta.
- T no designa una tabla utilizada en la cláusula de la subconsulta.
- T designa una tabla utilizada en un nivel superior de la jerarquía que contiene la subconsulta.

Debido a que una misma tabla, vista o apodo pueden estar identificados en muchos niveles, se recomienda utilizar nombres de correlación exclusivos como designadores de tabla. Si se utiliza T para designar una tabla en más de un nivel (T es el propio nombre de tabla o es un nombre de correlación duplicado), T.C hace referencia al nivel donde se utiliza T que contiene de forma más directa la subconsulta que incluye T.C. Si es necesario un nivel de correlación superior, debe utilizarse un nombre de correlación exclusivo.

La referencia correlacionada T.C identifica un valor de C en una fila o grupo de T a la que se aplican dos condiciones de búsqueda: la condición 1 en la subconsulta, y la condición 2 en algún nivel superior. Si se utiliza la condición 2 en una cláusula WHERE, se evalúa la subconsulta para cada fila a la que se aplica la condición 2. Si se utiliza la condición 2 en una cláusula HAVING, se evalúa la subconsulta para cada grupo al que se aplica la condición 2.

Por ejemplo, en la sentencia siguiente, la referencia correlacionada X.WORKDEPT (en la última línea) hace referencia al valor de WORKDEPT en la tabla EMPLOYEE en el nivel de la primera cláusula FROM. (Dicha cláusula establece X como nombre de correlación para EMPLOYEE.) La sentencia lista los empleados que tienen un salario inferior al promedio de su departamento.

```
SELECT EMPNO, LASTNAME, WORKDEPT
  FROM EMPLOYEE X
  WHERE SALARY < (SELECT AVG(SALARY)
                  FROM EMPLOYEE
                  WHERE WORKDEPT = X.WORKDEPT)
```

El ejemplo siguiente utiliza ESTE como nombre de correlación. La sentencia elimina las filas de los departamentos que no tienen empleados.

```
DELETE FROM DEPARTMENT THIS
   WHERE NOT EXISTS(SELECT *
                    FROM EMPLOYEE
                    WHERE WORKDEPT = THIS.DEPTNO)
```

Referencias a variables

Una *variable* de una sentencia de SQL especifica un valor que puede cambiarse cuando se ejecuta la sentencia de SQL. Existen diferentes tipos de variables utilizadas en sentencias de SQL:

variable del lenguaje principal
Las sentencias de un lenguaje principal definen las variables del lenguaje principal. Para obtener más información sobre cómo hacer referencia a variables del lenguaje principal, consulte el apartado “Referencias a variables del lenguaje principal” en la página 77.

variable de transición
Las variables de transición se definen en un activador y hacen referencia a los valores nuevos o anteriores de las columnas. Para obtener más información sobre cómo hacer referencia a variables de transición, consulte el apartado "Sentencia CREATE TRIGGER" del manual *Consulta de SQL, Volumen 2* ..

variable de SQL
Las variables de SQL se definen mediante una sentencia compuesta de SQL en una función de SQL, método de SQL, procedimiento de SQL, activador o sentencia de SQL dinámico. Para obtener más información sobre las variables de SQL, consulte el apartado "Referencias a parámetros SQL, variables SQL y variables globales" en el manual *Consulta de SQL, Volumen 2* ..

variable global
La sentencia CREATE VARIABLE define las variables globales. Para obtener más información sobre variables globales, consulte los apartados "CREATE VARIABLE" y "Referencias a parámetros de SQL, variables de SQL y variables globales" en el manual *Consulta de SQL, Volumen 2* .

variable de módulo
Las variables de módulo se definen mediante la sentencia ALTER MODULE utilizando la operación ADD VARIABLE o PUBLISH VARIABLE. Para obtener más información acerca de las variables de módulo, consulte "ALTER MODULE" en la publicación *Consulta de SQL, Volumen 2* .

parámetro de SQL
Los parámetros de SQL se definen en una sentencia CREATE FUNCTION, CREATE METHOD o CREATE PROCEDURE. Para obtener más información sobre los parámetros de SQL, consulte el apartado "Referencias a parámetros SQL, variables SQL y variables globales" en el manual *Consulta de SQL, Volumen 2* ..

marcador de parámetro
Los marcadores de parámetro se especifican en una sentencia de SQL dinámico donde se especificarían las variables del lenguaje principal si la sentencia fuera una sentencia de SQL estático. Se utiliza un descriptor de SQL o una vinculación de parámetros para asociar un valor con un marcador de parámetro durante el proceso de sentencia de SQL dinámico. Para obtener más información sobre marcadores de parámetro, consulte el apartado "Marcadores de parámetro" en *Consulta de SQL, Volumen 2* .

Referencias a variables del lenguaje principal

Una *variable del lenguaje principal* es:

- Variable de un lenguaje de sistema principal como una variable C, una variable C++, un elemento de datos COBOL, una variable FORTRAN, o una variable Java.

o:

- Una construcción del lenguaje principal generada por un precompilador de SQL a partir de una variable declarada mediante extensiones de SQL

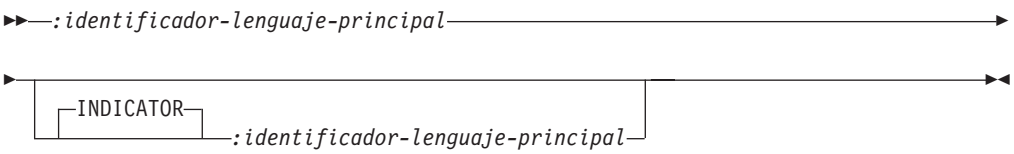
a la que se hace referencia en una sentencia de SQL. Las variables del lenguaje principal se definen directamente mediante las sentencias del lenguaje principal o indirectamente mediante extensiones de SQL.

Una variable del lenguaje principal en una sentencia de SQL debe identificar una variable del lenguaje principal descrita en el programa según las normas para la declaración de variables del lenguaje principal.

Todas las variables del lenguaje principal utilizadas en una sentencia de SQL deben estar declaradas en una sección DECLARE de SQL en todos los lenguajes principales excepto en REXX. No se debe declarar ninguna variable fuera de una sección DECLARE de SQL con nombres que sean idénticos a variables declaradas en una sección DECLARE de SQL. Una sección DECLARE de SQL empieza por BEGIN DECLARE SECTION y termina por END DECLARE SECTION.

La metavariable *variable-lenguaje-principal*, tal como se utiliza en los diagramas de sintaxis, muestra una referencia a una variable del lenguaje principal. Una variable del lenguaje principal utilizada como variable de destino en una sentencia de variable SET o en la cláusula INTO de una sentencia FETCH, SELECT INTO o VALUES INTO identifica una variable del lenguaje principal a la que se asigna un valor procedente de una columna de una fila o una expresión. En todos los demás contextos, una variable-lenguaje-principal especifica un valor que ha de pasarse al gestor de bases de datos desde el programa de aplicación.

Generalmente, la metavariable *variable-lenguaje-principal* se puede expandir en los diagramas de sintaxis a:



Cada *identificador-lenguaje-principal* debe declararse en el programa fuente. La variable designada por el segundo identificador-lenguaje-principal debe tener un tipo de datos de entero pequeño.

El primer identificador-lenguaje-principal designa la *variable principal*. Según la operación, proporciona un valor al gestor de bases de datos o bien el gestor de bases de datos le proporciona un valor. Una variable del lenguaje principal de entrada proporciona un valor en la página de códigos de la aplicación en tiempo de ejecución. A la variable del lenguaje principal de salida se le proporciona un valor que, si es necesario, se convierte a la página de códigos de la aplicación en tiempo de ejecución cuando los datos se copian en la variable de la aplicación de salida. Una variable del lenguaje principal determinada puede servir tanto de variable de entrada como de salida en el mismo programa.

El segundo identificador-lenguaje-principal designa su *variable indicadora*. La finalidad de la variable de indicador es:

- Especificar el valor nulo. Un valor negativo de la variable indicadora especifica el valor nulo. Un valor de -2 indica una conversión numérica o un error de expresión aritmética ocurrido al obtener el resultado
- Registra la longitud original de una serie truncada (si la fuente del valor no es un tipo de objeto grande)
- Registra la parte correspondiente a los segundos de una hora si la hora se trunca al asignarse a una variable del lenguaje principal.

Por ejemplo, si se utiliza :HV1:HV2 para especificar un valor de inserción o de actualización y si HV2 es negativo, el valor especificado es el valor nulo. Si HV2 no es negativo, el valor especificado es el valor de HV1.

De igual modo, si se especifica :HV1:HV2 en una cláusula INTO de una sentencia FETCH, SELECT INTO o VALUES INTO, , y se devuelve el valor nulo, HV1 no se cambia y HV2 se establece en un valor negativo Si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en yes (o lo estaba durante la vinculación de una sentencia de SQL estático), HV2 podría ser -2. Si HV2 es -2, no podría devolverse un valor para HV1 debido a un error en la conversión al tipo numérico de HV1 o a un error al evaluar una expresión aritmética utilizada para determinar el valor de HV1. Cuando se accede a una base de datos con una versión de cliente anterior a DB2 Universal Database, versión 5, HV2 será -1 para excepciones aritméticas. Si el valor devuelto no es nulo, se asigna dicho valor a HV1 y HV2 se establece en cero (a no ser que la asignación a HV1 necesite el truncamiento de una serie que sea no LOB, en cuyo caso HV2 se establece en la longitud original de la serie). Si una asignación necesita el truncamiento de la parte correspondiente a los segundos de una hora, HV2 se establece en el número de segundos.

Si se omite el segundo identificador del lenguaje principal, la variable del lenguaje principal carece de variable indicadora. El valor especificado por la referencia a la variable del lenguaje principal :HV1 siempre es el valor de HV1 y los valores nulos no se pueden asignar a la variable. Por este motivo, esta forma no debe utilizarse en una cláusula INTO a no ser que la columna correspondiente no pueda incluir valores nulos. Si se utiliza esta forma y la columna contiene valores nulos, el gestor de bases de datos generará un error en tiempo de ejecución.

Una sentencia de SQL que haga referencia a variables del lenguaje principal debe pertenecer al ámbito de la declaración de esas variables del lenguaje principal. En cuanto a las variables a las que la sentencia SELECT del cursor hace referencia, esa norma se aplica más a la sentencia OPEN que a la sentencia DECLARE CURSOR.

Ejemplo

Mediante la utilización de la tabla PROJECT, establezca la variable del lenguaje principal PNAME (VARCHAR(26)) en el nombre de proyecto (PROJNAME), la variable del lenguaje principal STAFF (DECIMAL(5,2)) en la dotación media de personal (PRSTAFF) y la variable del lenguaje principal MAJPROJ (CHAR(6)) en el proyecto principal (MAJPROJ) para el proyecto (PROJNO) 'IF1000'. Las columnas PRSTAFF y MAJPROJ podrían contener valores nulos; por lo tanto, proporcione las variables de indicador STAFF_IND (SMALLINT) y MAJPROJ_IND (SMALLINT).

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

Consideraciones acerca de MBCS: Si es o no es posible utilizar los caracteres de múltiples bytes en un nombre de variable del lenguaje principal depende del lenguaje principal.

Variables en SQL dinámico

En las sentencias de SQL dinámico, se utilizan marcadores de parámetro en vez de variables del lenguaje principal. Un marcador de parámetro representa una posición en una sentencia de SQL dinámico en la que la aplicación proporcionará

un valor; es decir, en la que se encontraría una variable del lenguaje principal si la serie de la sentencia fuera una sentencia de SQL estático. El siguiente ejemplo muestra una sentencia de SQL estático que emplea variables del lenguaje principal:

```
INSERT INTO DEPARTMENT
  VALUES (:HV_DEPTNO, :HV_DEPTNAME, :HV_MGRNO, :HV_ADMRDEPT)
```

Este ejemplo muestra una sentencia de SQL dinámico que utiliza marcadores de parámetro sin nombre:

```
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?)
```

Este ejemplo muestra una sentencia de SQL dinámico que utiliza los marcadores de parámetro con nombre:

```
INSERT INTO DEPARTMENT
  VALUES (:DEPTNO, :DEPTNAME, :MGRNO, :ADMRDEPT)
```

Los marcadores de parámetro con nombre se pueden utilizar para mejorar la legibilidad de la sentencia dinámica. Aunque los marcadores de parámetro con nombre se parecen a las variables del lenguaje principal, no tienen valores asociados, por lo que debe proporcionarse un valor para el marcador de parámetro al ejecutar la variable. Si se ha preparado la sentencia INSERT utilizando marcadores de parámetro con nombre y se ha denominado la sentencia preparada como DYNSTMT, se pueden proporcionar valores para los marcadores de parámetro mediante la sentencia siguiente:

```
EXECUTE DYNSTMT
  USING :HV_DEPTNO, :HV_DEPTNAME :HV_MGRNO, :HV_ADMRDEPT
```

Se podría utilizar esta misma sentencia EXECUTE si se hubiera preparado la sentencia INSERT utilizando marcadores de parámetro sin nombre y se hubiera denominado la sentencia preparada como DYNSTMT.

Referencias a variables LOB

Las variables regulares BLOB, CLOB y DBCLOB, las variables localizadoras LOB (consulte “Referencias a variables de localizador de LOB” en la página 81), y las variables de referencia a archivos LOB (consulte “Referencias a variables de referencia de archivo LOB” en la página 81) se pueden definir en todos los lenguajes principales. Donde se pueden utilizar valores LOB, el término *variable-lenguaje-principal* en un diagrama de sintaxis puede hacer referencia a una variable del lenguaje principal normal, a una variable localizadora o a una variable de referencia a archivos. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable. En cuanto a REXX, las variables LOB se correlacionan con series.

A veces es posible definir una variable lo suficientemente grande como para contener todo un valor de objeto grande. Si es así y no hay ninguna ventaja de rendimiento si se utiliza la transferencia diferida de datos desde el servidor, no es necesario un localizador. No obstante, puesto que el lenguaje principal o las restricciones de espacio se oponen al almacenamiento de un objeto grande entero en el almacenamiento temporal de una vez o por motivos de rendimiento, se puede hacer referencia a un objeto grande por medio de un localizador y las partes de dicho objeto se pueden seleccionar o actualizar en las variables del lenguaje principal que contengan sólo una parte del objeto grande.

Referencias a variables de localizador de LOB

Una *variable localizadora* es una variable del lenguaje principal que contiene el localizador que representa un valor de LOB en el servidor de aplicaciones.

Una variable localizadora de una sentencia de SQL debe identificar una variable localizadora descrita en el programa de acuerdo a las normas de declaración de variables localizadoras. Siempre se produce indirectamente a través de una sentencia de SQL.

El término variable localizadora, tal como se utiliza en los diagramas de sintaxis, muestra una referencia a una variable localizadora. La metavariable *variable-localizadora* puede expandirse para que incluya un *identificador-lenguaje-principal* igual que para la *variable-lenguaje-principal*.

Como sucede con el resto de variables del lenguaje principal, una variable localizadora de LOB puede tener asociada una variable indicadora. Las variables de indicador para las variables de localizador del lenguaje principal de objeto grande funcionan de la misma manera que las variables de indicador de otros tipos de datos. Cuando una base de datos devuelve un valor nulo, se define la variable indicadora y la variable localizadora del lenguaje principal no se cambia. Esto significa que un localizador jamás puede apuntar a un valor nulo.

Si se hace referencia a una variable localizadora que en ese momento no represente ningún valor, se producirá un error (SQLSTATE 0F001).

Durante la confirmación de la transacción, o en cualquier finalización de transacción, se liberan todos los localizadores que la transacción había adquirido.

Referencias a variables de referencia de archivo LOB

Las variables de referencia a archivos BLOB, CLOB y DBCLOB sirven para la entrada y salida directa de archivo para los LOB y pueden definirse en todos los lenguajes principales. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable. En cuanto a REXX, las variables LOB se correlacionan con series.

Una variable de referencia a archivos representa (más que contiene) al archivo, de igual manera que un localizador de LOB representa, más que contiene, a los bytes LOB. Las consultas, actualizaciones e inserciones pueden utilizar variables de referencia a archivos para almacenar o recuperar valores de una sola columna.

Una variable de referencia a archivos tiene las siguientes propiedades:

Tipo de datos
BLOB, CLOB o DBCLOB. Esta propiedad se especifica al declarar la variable.

Dirección
La dirección debe ser especificada por el programa de aplicación durante la ejecución (como parte del valor de Opciones de archivo). La dirección puede ser:

- De entrada (se utiliza como fuente de datos en las sentencias EXECUTE, OPEN, UPDATE, INSERT o DELETE).

- De salida (se utiliza como datos de destino en sentencias las FETCH o SELECT INTO).

Nombre del archivo
Debe especificarlo el programa de aplicación en tiempo de ejecución. Es uno de los siguientes:

- El nombre completo de la vía de acceso de un archivo (opción que se recomienda).
- Un nombre de archivo relativo. Si se proporciona un nombre de archivo relativo, se añade a la vía de acceso actual del proceso cliente.

En una aplicación, sólo debe hacerse referencia a un archivo en una variable de referencia a archivos.

Longitud del nombre de archivo
Debe especificarlo el programa de aplicación en tiempo de ejecución. Es la longitud del nombre de archivo (en bytes).

Opciones de archivo
Una aplicación debe asignar una las opciones a una variable de referencia a archivos antes de utilizar dicha variable. Las opciones se establecen mediante un valor INTEGER en un campo de la estructura de la variable de referencia a archivos. Se debe especificar alguna de estas opciones para cada variable de referencia a archivos:

- Entrada (de cliente a servidor)

  SQL_FILE_READ
  Archivo regular que se puede abrir, leer y cerrar. (La opción es SQL-FILE-READ en COBOL, sql_file_read en FORTRAN y READ en REXX.)

- Salida (de servidor a cliente)

  SQL_FILE_CREATE
  Crear un nuevo archivo. Si el archivo ya existe, se devuelve un error. (La opción es SQL-FILE-CREATE in COBOL, sql_file_create en FORTRAN y CREATE en REXX.)

  SQL_FILE_OVERWRITE (sobreescribir)
  Si ya existe un archivo con el nombre especificado, se sobreescribe el contenido del archivo; de lo contrario, se crea un nuevo archivo. (La opción es SQL-FILE-OVERWRITE en COBOL, sql_file_overwrite en FORTRAN y OVERWRITE en REXX.)

  SQL_FILE_APPEND
  Si ya existe un archivo con el nombre especificado, la salida se añade a éste; de lo contrario, se crea un nuevo archivo. (La opción es SQL-FILE-APPEND en COBOL, sql_file_append en FORTRAN y APPEND en REXX.)

Longitud de datos
No se utiliza en la entrada. En la salida, la implantación establece la longitud de datos en la longitud de los nuevos datos grabados en el archivo. La longitud se mide en bytes.

Como sucede con el resto de variables del lenguaje principal, una variable de referencia a archivo puede tener asociada una variable de indicador.

Ejemplo de una variable de referencia a archivos de salida (en C)

Supongamos una sección de declaración codificada como:

```
EXEC SQL BEGIN DECLARE SECTION
   SQL TYPE IS CLOB_FILE  hv_text_file;
   char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

Una vez procesada:

```
EXEC SQL BEGIN DECLARE SECTION
   /* SQL TYPE IS CLOB_FILE  hv_text_file; */
   struct {
       unsigned long  name_length; //  Longitud del nombre del archivo
       unsigned long  data_length; //  Longitud de datos
       unsigned long  file_options; // Opciones de archivo
       char           name[255];   // Nombre del archivo
   } hv_text_file;
   char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

El código siguiente puede utilizarse para seleccionar en una columna CLOB de la base de datos para un nuevo archivo al que :hv_text_file hace referencia.

```
strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_CREATE;

EXEC SQL SELECT content INTO :hv_text_file from papers
     WHERE TITLE = 'The Relational Theory behind Juggling';
```

Ejemplo de una variable de referencia a archivos de entrada (en C)

Tomando la misma sección de declaración que antes, se puede utilizar el siguiente código para insertar datos de un archivo normal al que :hv_text_file hace referencia en una columna CLOB.

```
strcpy(hv_text_file.name, "/u/gainer/patents/chips.13");
hv_text_file.name_length = strlen("/u/gainer/patents/chips.13");
hv_text_file.file_options = SQL_FILE_READ:
strcpy(:hv_patent_title, "A Method for Pipelining Chip Consumption");

EXEC SQL INSERT INTO patents( title, text )
         VALUES(:hv_patent_title, :hv_text_file);
```

Referencias a variables del lenguaje principal de tipo estructurado

Las variables de tipo estructurado pueden definirse en todos los lenguajes de sistema principal, excepto FORTRAN, REXX y Java. Puesto que no son tipos de datos nativos, se utilizan las extensiones SQL y los precompiladores generan las construcciones de lenguaje principal necesarias para poder representar a cada variable.

Al igual que en todas las demás variables del lenguaje principal, una variable de tipo estructurado puede tener una variable indicadora asociada. Las variables de indicador correspondientes a las variables del lenguaje principal de tipo estructurado actúan de la misma manera que las variables de indicador de otros tipos de datos. Cuando una base de datos devuelve un valor nulo, se define la variable indicadora y la variable del lenguaje principal de tipo estructurado no cambia.

La variable del lenguaje principal propiamente dicha correspondiente a un tipo estructurado está definida como tipo de datos interno. El tipo de datos interno asociado al tipo estructurado debe ser asignable:

- desde el resultado de la función de transformación FROM SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación; y
- al parámetro de la función de transformación TO SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación.

Si se utiliza un marcador de parámetros en lugar de una variable del lenguaje principal, se deben especificar las características apropiadas del tipo de parámetro en la SQLDA. Esto requiere un conjunto "duplicado" de estructuras SQLVAR en la SQLDA, y el campo SQLDATATYPE_NAME de la SQLVAR debe contener el nombre de esquema y nombre de tipo del tipo estructurado. Si se omite el esquema en la estructura SQLDA, se produce un error (SQLSTATE 07002).

Ejemplo

Defina las variables del lenguaje principal *hv_poly* y *hv_point* (de tipo POLYGON, utilizando el tipo interno BLOB(1048576)) en un programa C.

```
EXEC SQL BEGIN DECLARE SECTION;
      SQL estático
         TYPE IS POLYGON AS BLOB(1M)
         hv_poly, hv_point;
EXEC SQL END DECLARE SECTION;
```

vía de acceso de SQL

La vía de acceso de SQL es una lista ordenada de nombres de esquema. El gestor de bases de datos utiliza la vía de acceso de SQL para resolver el nombre de esquema para nombres de tipos de datos no calificados (tanto tipo incorporado como tipo diferenciado), de variable global, de módulo, de función y de procedimiento que aparecen en cualquier contexto que no sea el del objeto principal de una sentencia CREATE, DROP, COMMENT, GRANT o REVOKE. Para obtener información detallada, consulte “Calificación de los nombres de objeto no calificados”.

Por ejemplo, si la vía de acceso de SQL es SYSIBM. SYSFUN, SYSPROC, SYSIBMADM, SMITH, XGRAPHICS2 y se ha especificado un nombre de tipo diferenciado no calificado MYTYPE, el gestor de bases de datos busca MYTYPE en el esquema SYSIBM, después en SYSFUN, a continuación en SYSPROC, luego en SYSIBMADM, después en SMITH y por último en XGRAPHICS2.

La vía de acceso de SQL utilizada depende de la sentencia de SQL:

- Para las sentencias de SQL estático (salvo para la sentencia de variable CALL), la vía de acceso de SQL utilizada es la vía de acceso de SQL especificada al crear el paquete contenedor, el procedimiento, la función, el activador o la vista.
- Para las sentencias de SQL dinámico (y para la sentencia de variable CALL), la vía de acceso de SQL es el valor del registro especial CURRENT PATH. CURRENT PATH puede definirse mediante la sentencia SET PATH.

Si no se ha especificado explícitamente la vía de acceso de SQL, la vía de acceso de SQL será la vía de acceso del sistema seguida por el ID de autorización de la sentencia. .

Calificación de los nombres de objeto no calificados

Los nombres de objeto no calificados se califican de forma implícita. Las normas para la calificación de un nombre varían en función del tipo de objeto al que identifica el nombre.

Nombres de alias, índice, paquete, secuencia, tabla, activador y vista no calificados

El esquema por omisión califica de forma implícita los nombres de alias, índice, paquete, secuencia, tabla, activador y vista no calificados.

Para las sentencias de SQL estático, el esquema por omisión es el esquema por omisión especificado al crear la función, paquete, procedimiento o activador contenedor.

Para las sentencias de SQL dinámico, el esquema por omisión es el esquema especificado para el proceso de aplicación. El proceso de aplicación puede especificar el esquema por omisión mediante la sentencia SET SCHEMA. Si no se ha especificado explícitamente el esquema por omisión, el esquema por omisión será el ID de autorización de la sentencia.

Nombres no calificados de función, procedimiento, específico, variable global, módulo y tipo definido por el usuario

La calificación de los nombres de tipo de datos (tanto incorporado como diferenciado), variable global, módulo, función, procedimiento y específicos depende de la sentencia de SQL en la que aparezca el nombre no calificado:

- Si un nombre no calificado es el objeto principal de una sentencia CREATE, ALTER, COMMENT, DROP, GRANT o REVOKE, el nombre se califica implícitamente aplicando las mismas normas que se usan para calificar los nombres de tabla no calificados (consulte el apartado “Nombres de alias, índice, paquete, secuencia, tabla, activador y vista no calificados ”). El objeto principal de una operación ADD, COMMENT, DROP o PUBLISH de la sentencia ALTER MODULE debe especificarse sin ningún calificador.
- Si el contexto de la referencia está ubicado en un módulo, el gestor de bases de datos busca el objeto en el módulo, aplicando la resolución adecuada al tipo de objeto para encontrar una coincidencia. Si no encuentra coincidencias, la búsqueda continúa tal y como se especifica en el punto siguiente.
- En caso contrario, el nombre de esquema implícito se determina de la manera siguiente:
  - Para los nombres de tipo diferenciado, el gestor de bases de datos busca en la vía de acceso de SQL y selecciona el primer esquema en dicha vía en el que exista el tipo de datos.
  - Para las variables globales, el gestor de bases de datos busca en la vía de acceso de SQL y selecciona el primer esquema en dicha vía en el que exista la variable global.
  - Para los nombres de procedimiento, el gestor de bases de datos utiliza la vía de acceso de SQL en combinación con la resolución de procedimiento.
  - Para los nombres de función, el gestor de bases de datos utiliza la vía de acceso de SQL en combinación con la resolución de función.
  - Para los nombres específicos determinados para las funciones con fuente, consulte la sección “CREATE FUNCTION (con fuente)”.

Resolución de los nombres de objeto calificados

Los objetos definidos en un módulo que están disponibles para su uso fuera del mismo deben estar calificados por el nombre de módulo. Dado que un módulo es un objeto de esquema que también se puede calificar implícitamente, los objetos de módulo publicados pueden calificarse utilizando un nombre de módulo no calificado o un nombre de módulo calificado mediante esquema. Cuando se utiliza un nombre de módulo no calificado, la referencia al objeto de módulo tiene el mismo aspecto que un objeto calificado mediante esquema que no forma parte del módulo. Dentro de un ámbito específico, por ejemplo una sentencia de SQL compuesto, un identificador de dos partes podría ser también:

- un nombre de columna calificado por un nombre de tabla
- un nombre de campo de fila calificado por un nombre de variable
- un nombre de variable calificado por una etiqueta
- un nombre de parámetro de rutina calificado por un nombre de rutina

Estos objetos se resuelven dentro de su ámbito, antes de tener en cuenta los objetos de esquema o los objetos de módulo. El proceso siguiente se utiliza para resolver objetos con identificadores de dos partes que podrían ser un objeto de esquema o un objeto de módulo.

- Si el contexto de la referencia está ubicado en un módulo y el calificador coincide con el nombre del módulo, el gestor de bases de datos busca el objeto en el módulo, aplicando la resolución adecuada al tipo de objeto para encontrar una coincidencia entre los objetos de módulo publicados y no publicados. Si no encuentra coincidencias, la búsqueda continúa tal y como se especifica en los puntos siguientes.
- Se presupone que el calificador es un nombre de esquema y, si el esquema existe, se resuelve el objeto en el esquema.
- Si el calificador no es un esquema existente o el objeto no se encuentra en el esquema que coincide con el calificador, y éste no coincidió con el nombre de módulo de contexto, se busca el primer módulo que coincida con el calificador en los esquemas de la vía de acceso de SQL. Si se dispone de autorización para el módulo que coincide, se resuelve al objeto de dicho módulo, teniendo en cuenta únicamente los objetos de módulo publicados.
- Si el módulo no se encuentra como módulo en la vía de acceso de SQL y el calificador no coincidió con el nombre de módulo de contexto, se busca un sinónimo público de módulo que coincida con el calificador. Si se encuentra, se resuelve el objeto en el módulo identificado por el sinónimo público de módulo, teniendo en cuenta únicamente los objetos de módulo publicados.

Tipos de datos

La unidad más pequeña de datos que se puede manipular en SQL se denomina un *valor.* Los valores se interpretan según el tipo de datos de su fuente. Entre los fuentes se incluyen:

- Constantes
- Columnas
- Funciones
- Expresiones
- Registros especiales.
- Variables (como variables de sistema principal, variables de SQL, variables globales, marcadores de parámetro, variables de módulo y parámetros de rutinas)
- Valores booleanos

DB2 da soporte a una serie de tipos de datos incorporados. También proporciona soporte para los tipos de datos definidos por el usuario. La Figura 10 en la página 88 ilustra los tipos de datos internos a los que se da soporte.

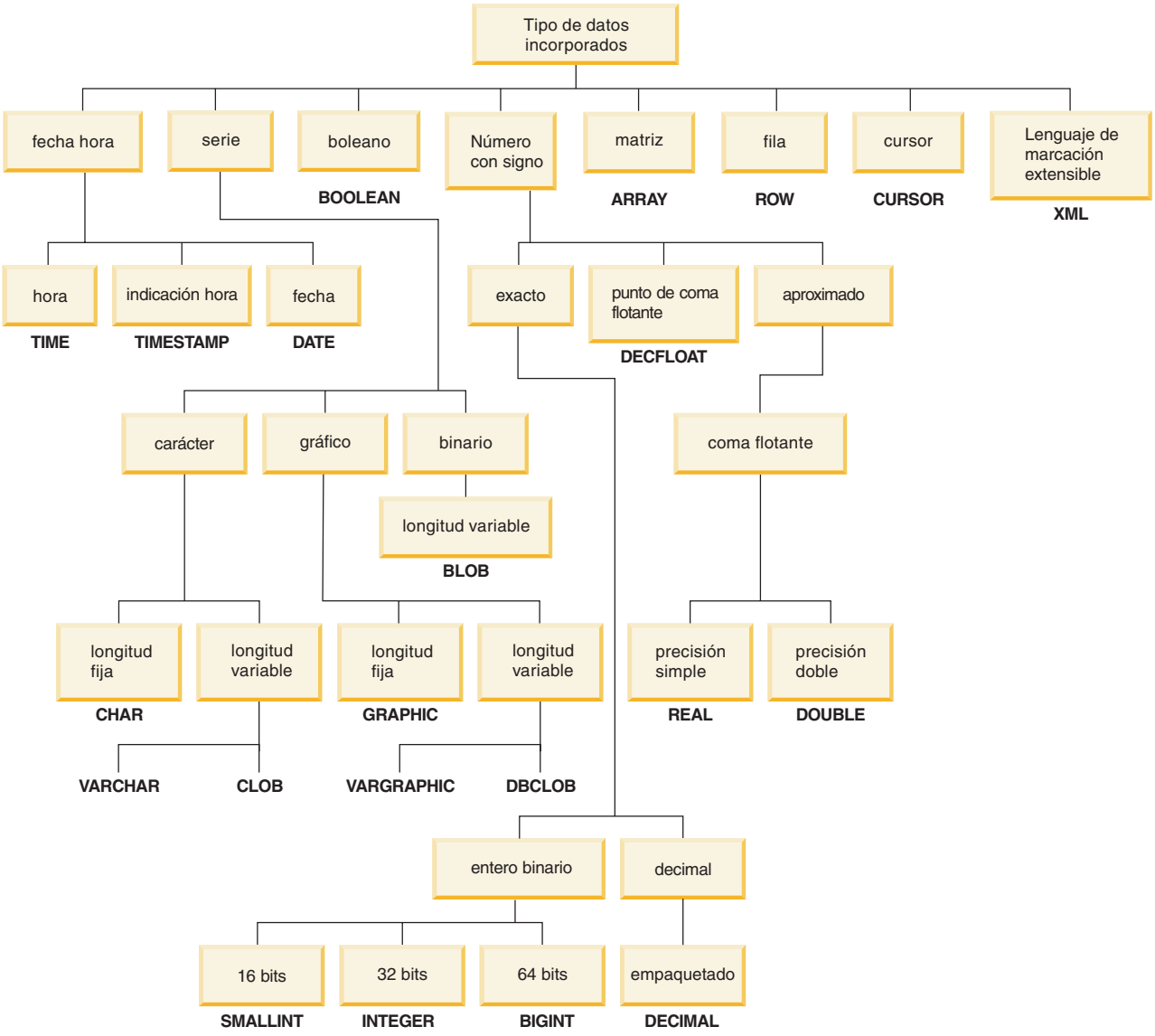


Figura 10. Tipos de datos internos de DB2 soportados

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

Lista de tipos de datos

Números

Los tipos de datos numéricos son entero, decimal, coma flotante y coma flotante decimal.

Los tipos de datos numéricos se clasifican de la siguiente manera:

- Numéricos exactos: enteros y decimales
- Coma flotante decimal
- Numéricos aproximados: coma flotante

Entre los enteros se incluyen los enteros pequeños, los grandes enteros y entero superior. Los números enteros son representaciones exactas de los enteros. Los números decimales son representaciones exactas de números con una precisión y una escala fijas. Los números decimales y enteros se consideran tipos numéricos exactos.

Los números de coma flotante pueden tener una precisión de 16 ó 34. La coma flotante decimal soporta las representaciones exactas de números reales y la aproximación de los números reales y, por lo tanto, no se considera un tipo numérico exacto ni un tipo numérico aproximado.

La coma flotante incluye la precisión simple y la precisión doble. Los números de coma flotante son aproximaciones de números reales y se consideran tipos numéricos aproximados.

Todos los números tienen un *signo*, una *precisión* y una *escala*. Para todos los números excepto la coma flotante decimal, si el valor de columna es cero, el signo será positivo. Los números de coma flotante decimal incluyen ceros positivos y negativos. La coma flotante decimal tiene valores diferenciados para un número y el mismo número con varios exponentes (por ejemplo: 0,0, 0,00, 0,0E5, 1,0, 1,00, 1,0000). La precisión es el número total de dígitos decimales, excluyendo el signo. La escala es el número total de dígitos decimales a la derecha de la coma decimal. Si no hay una coma decimal, la escala es cero.

Consulte asimismo el apartado sobre tipo de datos en la descripción de la sentencia CREATE TABLE.

Entero pequeño (SMALLINT)

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

Entero grande (INTEGER)

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

Entero superior (BIGINT)

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

Decimal (DECIMAL o NUMERIC)

Un valor *decimal* es un número decimal empaquetado con una coma decimal implícita. La posición de la coma decimal la determinan la precisión y la escala del número. La escala, que es el número de dígitos en la parte de la fracción del número, no puede ser negativa ni mayor que la precisión. La precisión máxima es de 31 dígitos.

Todos los valores de una columna decimal tienen la misma precisión y escala. El rango de una variable decimal o de los números de una columna decimal es de *-n* a *+n*, donde el valor absoluto de *n* es el número mayor que puede representarse con la precisión y escalas aplicables. El rango máximo va de $-10^{31}+1$ a $10^{31}-1$.

Coma flotante de precisión simple (REAL)

Un número de *coma flotante de precisión simple* es una aproximación de 32 bits de un número real. El número puede ser cero o puede estar en el rango de -3,4028234663852886e+38 a -1.1754943508222875e-38, o de 1,1754943508222875e-38 a 3,4028234663852886e+38.

Coma flotante de doble precisión (DOUBLE o FLOAT)

Un número de *coma flotante de doble precisión* es una aproximación de 64 bits de un número real. El número puede ser cero o puede estar en el rango de -1,7976931348623158e+308 a -2,2250738585072014e-308, o de 2,2250738585072014e-308 a 1,7976931348623158e+308.

Coma flotante decimal (DECFLOAT)

Un valor de *coma flotante decimal* es un número IEEE 754r con una coma decimal. La posición de la coma decimal se almacena en cada uno de los valores de coma flotante decimal. La precisión máxima es de 34 dígitos. El rango de un número de coma flotante decimal es de 16 ó 34 dígitos de precisión y un rango de exponentes de 10^{-383} a 10^{+384} ó de 10^{-6143} a 10^{+6144}, respectivamente. El exponente mínimo, E_{min}, para valores de DECFLOAT es -383 para DECFLOAT(16) y -6143 para DECFLOAT(34). El exponente máximo, E_{max}, para valores DECFLOAT es 384 para DECFLOAT(16) y 6144 para DECFLOAT(34).

Además de los números finitos, los números de coma flotante decimal pueden representar uno de los siguientes valores especiales de coma flotante decimal con nombre:

- Infinity - Un valor que representa un número cuya magnitud es infinitamente grande
- Quiet NaN - Es un valor que representa resultados indefinidos y que no causa un aviso de número no válido
- Signalling NaN - Es un valor que representa resultados indefinidos y que causa un aviso de número no válido si se utiliza en cualquier operación numérica

Cuando un número tiene uno de estos valores especiales, su coeficiente y exponente no están definidos. El signo de un valor de infinito es significativo, ya que es posible tener un infinito positivo o negativo. El signo de un valor NaN no tiene significado en las operaciones aritméticas.

Números anormales y subdesbordamiento

Los números distintos de cero cuyos exponentes ajustados son menores que E_{min} se denominan números anormales. Estos números anormales se aceptan como operandos para todas las operaciones y pueden ser el resultado de cualquier operación.

Para un resultado anormal, los valores mínimos del exponente se convierten en $_{min}$ - (precisión-1), denominado E_{tiny}, donde la precisión es la precisión de trabajo. Si es necesario, el resultado se redondea para asegurarse de que el exponente no sea inferior a E_{tiny}. Si el resultado queda inexacto durante el redondeo, se devuelve un aviso de subdesbordamiento. Un resultado anormal no siempre devolverá el aviso de subdesbordamiento.

Cuando un número se subdesborda a cero durante un cálculo, su exponente será E_{tiny}. El valor máximo del exponente no resulta afectado.

El valor máximo del exponente para números anormales es el mismo que el valor mínimo del exponente que puede surgir durante las operaciones que no den como resultado números anormales. Esto se produce cuando la longitud del coeficiente en dígitos decimales es igual a la precisión.

Series de caracteres

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

Serie de caracteres de longitud fija (CHAR)

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

Series de caracteres de longitud variable

Existen dos tipos de series de caracteres de longitud variable:

- Un valor VARCHAR puede tener una longitud máxima de 32.672 bytes.
- Un valor CLOB (objeto grande de caracteres) puede tener una longitud máxima de 2 gigabytes menos 1 byte (2.147.483.647 bytes). Un CLOB se utiliza para almacenar datos basados en caracteres SBCS o mixtos (SBCS y MBCS) (como, por ejemplo, documentos escritos con un solo juego de caracteres) y, por lo tanto, tiene una página de códigos SBCS o mixta asociada).

Se aplican restricciones especiales a las expresiones que dan como resultado un valor de tipo de datos CLOB y a columnas de tipo estructurado; estas expresiones y columnas no se permiten en:

- Una lista SELECT precedida por la cláusula DISTINCT
- Una cláusula GROUP BY
- Una cláusula ORDER BY
- Una subselección de un operador de conjunto que no sea UNION ALL
- Un predicado BETWEEN o IN básico y cuantificado
- Una función agregada
- Las funciones escalares VARGRAPHIC, TRANSLATE y de fecha y hora
- El operando patrón de un predicado LIKE o el operando de serie de búsqueda de una función POSSTR
- La representación en una serie de un valor de fecha y hora.

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

Las series de caracteres terminadas en nulo que se encuentran en C se manejan de manera diferente, dependiendo del nivel de estándares de la opción de precompilación.

Cada serie de caracteres se define con más detalle como:

Datos de bit
Datos que no están asociado con una página de códigos.

Datos de juego de caracteres de un solo byte (SBCS)
Datos en los que cada carácter está representado por un solo byte.

Datos mixtos
Datos que pueden contener una mezcla de caracteres de un juego de caracteres de un solo byte y de un juego de caracteres de múltiples bytes (MBCS).

Nota: El tipo de datos LONG VARCHAR sigue estando soportado pero ha quedado obsoleto, no es recomendable y puede eliminarse en un release futuro.

Unidades de serie en las funciones incorporadas

La capacidad para especificar unidades de serie para determinadas funciones incorporadas permite procesar los datos de las series basándose más en los caracteres que en los bytes. La *unidad de serie* determina la longitud sobre la que se realizará una operación. Puede especificar CODEUNITS16, CODEUNITS32 u OCTETS como unidad de serie para una operación.

CODEUNITS16
Especifica que la unidad de la operación será Unicode UTF-16. CODEUNITS16 es útil en el caso de aplicaciones que procesan datos en unidades de código que tienen dos bytes de ancho. Tenga en cuenta que algunos caracteres, conocidos como *caracteres suplementarios*, necesitan dos unidades de código UTF-16 para codificarse. Por ejemplo, la clave G de símbolo musical necesita dos unidades de código UTF-16 (X'D834' y X'DD1E' en UTF-16BE).

CODEUNITS32
Especifica que la unidad de operación será Unicode UTF-32. CODEUNITS32 es útil en el caso de aplicaciones que procesan datos en un formato sencillo, de longitud fija, que debe devolver la misma respuesta, con independencia del formato de almacenamiento de los datos (ASCII, UTF-8 o UTF-16).

OCTETS
Especifica que las unidades de operación son los bytes. OCTETS se utiliza a menudo cuando una aplicación tiene interés en asignar espacio de almacenamiento intermedio o cuando las operaciones necesitan utilizar un proceso de bytes sencillo.

La longitud calculada de una serie obtenida mediante OCTETS (bytes) puede ser diferente de la que se calcula mediante CODEUNITS16 o CODEUNITS32. Cuando se utiliza OCTETS, la longitud de la serie se determina simplemente contando el número de bytes de la serie, mientras que cuando se utiliza CODEUNITS16 o CODEUNITS32, la longitud de la serie se determina contando el número de unidades de código de 16 bits o de 32 bits necesarias para representar la serie en UTF-16 o UTF-32, respectivamente. La longitud que se determine mediante CODEUNITS16 o CODEUNITS32 será idéntica, a menos que los datos contengan caracteres suplementarios (vea "Diferencia entre CODEUNITS16 y CODEUNITS32" en la página 94).

Por ejemplo, supongamos que NAME, una columna VARCHAR(128) codificada en Unicode UTF-8, contiene el valor 'Jürgen'. Las dos consultas siguientes, que cuentan la longitud de la serie CODEUNITS16 y CODEUNITS32, respectivamente, devuelven el mismo valor (6).

```
SELECT CHARACTER_LENGTH(NAME,CODEUNITS16) FROM T1
  WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME,CODEUNITS32) FROM T1
  WHERE NAME = 'Jürgen'
```

La consulta siguiente, que cuenta la longitud de la serie en OCTETS, devuelve el valor 7.

```
SELECT CHARACTER_LENGTH(NAME,OCTETS) FROM T1
  WHERE NAME = 'Jürgen'
```

Estos valores representan la longitud de la serie expresada en la unidad de serie especificada.

La tabla siguiente muestra las representaciones UTF-8, UTF-16BE (big endian) y UTF-32BE (big endian) del nombre 'Jürgen':

```
Formato   Representación del nombre 'Jürgen'
--------  ------------------------------------
UTF-8   X'4AC3BC7267656E'
UTF-16BE  X'004A00FC007200670065006E'
UTF-32BE  X'0000004A000000FC00000072000000670000006500000006E'
```

El carácter 'ü' se representa de forma diferente en las tres unidades de serie:

- La representación UTF-8 del carácter 'ü' es X'C3BC'.
- La representación UTF-16BE del carácter 'ü' es X'00FC'.
- La representación UTF-32BE del carácter 'ü' es X'000000FC'.

La especificación de unidades de serie para una función incorporada no repercute sobre los tipos de datos o la página de códigos del resultado de la función. Si es preciso, DB2 convierte los datos a Unicode para una evaluación, cuando se especifica CODEUNITS16 o CODEUNITS32.

Cuando se especifica OCTETS para la función LOCATE o POSITION y las páginas de código de los argumentos de la serie varían, DB2 convierte los datos en la página de códigos del argumento *serie-fuente*. En dicho caso, el resultado de la función se encuentra en la página de códigos del argumento *serie-fuente*. Cuando se especifica OCTETS para funciones que toman un único argumento de serie, los datos se evalúan en la página de códigos del argumento de la serie y el resultado de la función se encontrará en la página de códigos del argumento de la serie.

Diferencia entre CODEUNITS16 y CODEUNITS32

Cuando se especifica CODEUNITS16 o CODEUNITS32, el resultado será el mismo, salvo en el caso de los datos que contengan caracteres Unicode suplementarios. Esto se debe a que los caracteres Unicode suplementarios se representan mediante dos unidades de código UTF-16 o una unidad de código UTF-32. En UTF-8, los caracteres que no sean suplementarios se representan mediante 1 a 3 bytes, y un carácter suplementario se representa mediante 4 bytes. En UTF-16, los caracteres que no sean suplementarios están representados por una unidad de código CODEUNITS16 o 2 bytes y un carácter suplementario está representado mediante dos unidades de código CODEUNITS16 o 4 bytes. En UTF-32, un carácter se representa mediante una unidad de código CODEUNITS32 o 4 bytes.

Por ejemplo, la tabla siguiente muestra los valores hexadecimales de la A mayúscula matemática en negrita y la A mayúscula latina. La A mayúscula matemática en negrita es un carácter suplementario que se representa mediante 4

bytes en UTF-8, UTF-16 y UTF-32.

Carácter	Representación UTF-8	Representación UTF-16BE	Representación UTF-32BE
Valor Unicode X'1D400' - 'A'; A mayúscula matemática en negrita	X'F09D9080'	X'D835DC00'	X'0001D400'
Valor Unicode X'0041' - 'A'; A mayúscula latina	X'41'	X'0041'	X'00000041'

Supongamos que C1 es una columna VARCHAR(128), codificada en Unicode UTF-8, y que la tabla T1 contiene una fila con el valor de la A mayúscula matemática en negrita (X'F09D9080'). Las consultas siguientes devuelven resultados diferentes:

```
Consulta                                          Devuelve
-----                                             -------
SELECT CHARACTER_LENGTH(C1,CODEUNITS16) FROM T1   2

SELECT CHARACTER_LENGTH(C1,CODEUNITS32) FROM T1   1

SELECT CHARACTER_LENGTH(C1,OCTETS) FROM T1        4
```

Series gráficas

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

Las series gráficas no se comprueban para asegurarse de que sus valores sólo contienen elementos de código de caracteres de doble byte. (La excepción a esta norma es una aplicación precompilada con la opción WCHARTYPE CONVERT. En este caso, sí que se efectúa la validación.) En lugar de esto, el gestor de bases de datos supone que los datos de caracteres de doble byte están contenidos en campos de datos gráficos. El gestor de bases de datos *sí* que comprueba que un valor de la longitud de una serie gráfica sea número par de bytes.

Las series gráficas terminadas en nulo que se encuentran en C se manejan de manera diferente, dependiendo del nivel de estándares de la opción de precompilación. Este tipo de datos no puede crearse en una tabla. Sólo se puede utilizar para insertar datos en la base de datos y recuperarlos de la misma.

Series gráficas de longitud fija (GRAPHIC)

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

Series gráficas de longitud variable

Existen dos tipos de serie gráfica de longitud variable:

- Un valor VARGRAPHIC puede tener una longitud máxima de 16.336 caracteres de doble byte.
- Un valor DBCLOB (objeto grande de caracteres de doble byte) puede tener una longitud máxima de 1.073.741.823 caracteres de doble byte. Un DBCLOB se utiliza para almacenar datos DBCS grandes basados en caracteres (por ejemplo, documentos escritos con un solo juego de caracteres) y, por lo tanto, tiene asociada una página de códigos DBCS).

Se aplican restricciones especiales a una expresión que dé como resultado una serie gráfica de longitud variable cuya longitud máxima sea mayor que 127 bytes. Estas restricciones son las mismas que las especificadas en el apartado “Series de caracteres de longitud variable” en la página 92.

Nota: El tipo de datos LONG VARGRAPHIC sigue estando soportado pero ha quedado obsoleto, no es recomendable y puede eliminarse en un release futuro.

Series binarias

Una *serie binaria* es una secuencia de bytes. A diferencia de las series de caracteres, que suelen contener datos de texto, las series binarios se utilizan para contener datos no tradicionales como, por ejemplo, imágenes, voz o soportes mixtos. Se pueden utilizar series de caracteres del subtipo FOR BIT DATA para objetivos similares. Las series binarias no están asociadas a ninguna página de códigos. Las series binarias tienen las mismas restricciones que las series de caracteres (consulte los detalles en el apartado "Series de caracteres de longitud variable" en la página 92). Sólo son compatible con series binarias las series de caracteres del subtipo FOR BIT DATA.

Objeto grande binario (BLOB)

Un *objeto grande binario* es una serie binaria de longitud variable que puede tener una longitud máxima de 2 gigabytes menos 1 byte (2.147.483.647 bytes). Los valores BLOB pueden contener datos estructurados para que los utilicen las funciones definidas por el usuario y los tipos definidos por el usuario. Igual que las series de caracteres FOR BIT DATA, las series BLOB no están asociadas a ninguna página de códigos.

Objeto grande (LOB)

El término *objeto grande* y el acrónimo genérico LOB hace referencia al tipo de datos BLOB, CLOB o DBCLOB. Los valores LOB están sujetos a restricciones, como se describe en “Series de caracteres de longitud variable” en la página 92. Estas restricciones se aplican incluso si el atributo de longitud de la serie LOB es de 254 bytes o menor.

Los valores LOB pueden ser muy grandes y la transferencia de dichos valores desde servidor de bases de datos a las variables del lenguaje principal del programa de aplicación cliente puede tardar mucho tiempo. Como normalmente los programas de aplicación procesan los valores LOB de fragmento en fragmento en lugar de como un todo, las aplicaciones pueden hacer referencia a un valor LOB utilizando un localizador de objeto grande.

Un *localizador de objeto grande* o localizador de LOB es una variable del lenguaje principal cuyo valor representa un solo valor LOB del servidor de bases de datos.

Un programa de aplicación puede seleccionar un valor LOB en un localizador de LOB. Entonces, utilizando el localizador de LOB, el programa de aplicación puede solicitar operaciones de base de datos basadas en el valor LOB (por ejemplo, aplicar las funciones escalares SUBSTR, CONCAT, VALUE o LENGTH, realizar una asignación, efectuar búsquedas en el LOB con LIKE o POSSTR o aplicar funciones definidas por el usuario sobre el LOB) proporcionando el valor del localizador como entrada. La salida resultante (los datos asignados a una variable del lenguaje principal cliente), sería normalmente un subconjunto pequeño del valor LOB de entrada.

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

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

Cuando se selecciona un valor nulo en una variable del lenguaje principal normal, la variable de indicador se establece en -1, lo que significa que el valor es nulo. Sin embargo, en el caso de los localizadores de LOB, el significado de las variables de indicador es ligeramente distinto. Como una variable del lenguaje principal del localizador en sí nunca puede ser nula, un valor negativo de variable de indicador significa que el valor LOB representado por el localizador de LOB es nulo. La información de nulo se mantiene local para el cliente en virtud del valor de la variable de indicador — el servidor no hace ningún seguimiento de los valores nulos con localizadores válidos.

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

Un localizador de LOB es sólo un mecanismo utilizado para hacer referencia a un valor LOB durante una transacción; no persiste más allá de la transacción en la que se ha creado. No es un tipo de base de datos; nunca se almacena en la base de datos y, como resultado, no puede participar en vistas ni en restricciones de comprobación. Sin embargo, como un localizador de LOB es una representación cliente de un tipo LOB, hay SQLTYPE para localizadores de LOB para que puedan describirse dentro de una estructura SQLDA que se utiliza por sentencias FETCH, OPEN y EXECUTE.

Valores de fecha y hora

Entre los tipos de datos de fecha y hora se incluyen DATE, TIME y TIMESTAMP. Aunque los valores de fecha y hora se pueden utilizar en algunas operaciones aritméticas y de series y son compatibles con algunas series, no son ni series ni números.

Fecha

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

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

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

Hora

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

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

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

Indicación de fecha y hora

Una *indicación de fecha y hora* es un valor de seis o siete partes (año, mes, día, hora, minuto y segundos fraccionales opcionales) que designa una fecha y hora tal como se ha descrito anteriormente, salvo que la hora también puede incluir una parte adicional que designa una fracción de un segundo. El número de dígitos de los segundos fraccionales se especifica mediante un atributo en el rango comprendido entre 0 y 12, con un valor por omisión de 6.

La representación interna de una indicación de fecha y hora es una serie que tieneentre 7 y 13 bytes. Cada byte consta de 2 dígitos decimales empaquetados. Los primeros 4 bytes representan la fecha, los 3 siguientes la hora y los últimos 0 a 6 bytes los segundos fraccionales.

La longitud de una columna TIMESTAMP, tal como se describe en la SQLDA, está comprendida entre 19 y 32 bytes, que es la longitud adecuada para la representación de serie de caracteres del valor.

Representación mediante series de los valores de fecha y hora

Los valores cuyos tipos de datos son DATE, TIME o TIMESTAMP se representan en un formato interno que es transparente para el usuario. Sin embargo, los valores de fecha, hora e indicación de fecha y hora también pueden representarse mediante series. Esto resulta útil porque no existen constantes ni variables cuyo tipo de datos sean DATE, TIME o TIMESTAMP. Antes de poder recuperar un valor de fecha y hora, éste debe asignarse a una variable de serie. La función GRAPHIC (sólo para bases de datos Unicode) puede utilizarse para cambiar el valor de fecha y hora a una representación de serie. Normalmente, la representación de serie es el formato por omisión de los valores de fecha y hora asociados con el código territorial de la aplicación, a menos que se alteren temporalmente por la especificación de la opción DATETIME al precompilar el programa o vincularlo con la base de datos.

Independientemente de la longitud, no se puede utilizar una serie de gran objeto como representación en forma de serie de un valor de fecha y hora (SQLSTATE 42884).

Cuando se utiliza una representación de serie válida de un valor de fecha y hora en una operación con un valor de fecha y hora interno, la representación de serie se convierte al formato interno del valor de fecha, hora o indicación de fecha y hora antes de realizar la operación.

Las series de fecha, hora e indicación de fecha y hora sólo deben contener caracteres y dígitos.

Series de fecha

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

Los formatos válidos para las series se indican en la tabla siguiente. Cada formato se identifica mediante el nombre y la abreviatura asociada.

Tabla 8. Formatos para las representaciones de serie de fechas

Nombre del formato	Abreviatura	Formato de fecha	Ejemplo
International Standards Organization	ISO	aaaa-mm-dd	1991-10-27
Estándar IBM USA	USA	mm/dd/aaaa	10/27/1991
Estándar IBM European	EUR	dd.mm.aaaa	27.10.1991
Era Japanese Industrial Standard Christian	JIS	aaaa-mm-dd	1991-10-27
Definido-sitio	LOC	Depende del código territorial de la aplicación	—

Series de hora

Una representación de serie de una hora es una serie que empieza por un dígito y que tiene una longitud de 4 caracteres como mínimo. Pueden incluirse blancos de

cola; puede omitirse un cero inicial de la parte correspondiente a la hora y pueden omitirse por completo los segundos. Si se omiten los segundos, se supone una especificación implícita de 0 segundos. De este modo, 13:30 es equivalente a 13:30:00.

Los formatos válidos para las series de horas se indican en la tabla siguiente. Cada formato se identifica mediante el nombre y la abreviatura asociada.

Tabla 9. Formatos para representaciones de serie de horas

Nombre del formato	Abreviatura	Formato de la hora	Ejemplo
International Standards Organization	ISO	hh.mm.ss	13.30.05
Estándar IBM USA	USA	hh:mm AM o PM	1:30 PM
Estándar IBM European	EUR	hh.mm.ss	13.30.05
Era Japanese Industrial Standard Christian	JIS	hh:mm:ss	13:30:05
Definido-sitio	LOC	Depende del código territorial de la aplicación	—

Nota:

1. En el formato ISO, EUR o JIS, .ss (o :ss) es opcional.
2. La organización International Standards Organization ha cambiado el formato de la hora, de modo que ahora es idéntico al de Japanese Industrial Standard Christian Era. Por lo tanto, utilice el formato JIS si una aplicación necesita el formato actual de International Standards Organization.
3. En el formato de serie de hora USA, puede omitirse la especificación de los minutos, con lo que se indica una especificación implícita de 00 minutos. Por lo tanto, 1 PM equivale a 1:00 PM.
4. En el formato de hora USA, la hora no debe ser mayor que 12 y no puede ser 0, excepto en el caso especial de 00:00 AM. Hay un solo espacio antes de 'AM' o 'PM'. 'AM' y 'PM' se pueden representar en minúsculas o en mayúsculas.

   Si se utiliza el formato JIS del reloj de 24 horas, la correspondencia entre el formato USA y el reloj de 24 horas es la siguiente:

   - 12:01 AM a 12:59 AM corresponde a 00:01:00 a 00:59:00.
   - 01:00 AM a 11:59 AM corresponde a 01:00:00 a 11:59:00.
   - 12:00 PM (mediodía) a 11:59 PM corresponde a 12:00:00 a 23:59:00.
   - 12:00 AM (medianoche) corresponde a 24:00:00 y 00:00 AM (medianoche) corresponde a 00:00:00.

Series de indicación de fecha y hora

Una representación de serie de una indicación de fecha y hora es una serie que empieza por un dígito y que tiene una longitud de 16 caracteres como mínimo. La representación de la serie completa de una indicación de fecha y hora tiene el formato *aaaa-mm-dd-hh.mm.ss* o *aaaa-mm-dd-hh.mm.ss.nnnnnnnnnnnn*, donde el número de dígitos de los segundos fraccionales puede estar comprendido entre 0 y 12. Se pueden incluir los blancos de cola. Se pueden omitir los ceros del principio de la parte del mes, del día y de la hora de la indicación de fecha y hora. Los ceros de la cola se pueden truncar u omitir totalmente de los segundos fraccionales. Si

una representación de serie de una fecha y hora se convierte implícitamente en un valor con un tipo de datos TIMESTAMP, la precisión de la indicación de fecha y hora del resultado de la conversión se determinará mediante la precisión del operando TIMESTAMP en una expresión o la precisión del destino TIMESTAMP de una asignación. Los dígitos de la serie que superen la precisión de indicación de fecha y hora de la conversión se truncarán, o se presupondrá que los dígitos que faltan que alcancen la precisión de indicación de fecha y hora de la conversión serán ceros. Por ejemplo, 1991-3-2-8.30.00 equivale a 1991-03-02-08.30.00.000000000000.

Se puede dar una precisión de indicación de fecha y hora diferente a una representación de serie de una indicación de fecha y hora convirtiendo explícitamente el valor a una indicación de fecha y hora con una precisión especificada. Si la serie es una constante, una alternativa consiste en poner delante de la constante de serie la palabra clave TIMESTAMP. Por ejemplo, TIMESTAMP '2007-03-28 14:50:35.123' tiene el tipo de datos TIMESTAMP(3).

Las sentencias de SQL también dan soporte a la representación de serie ODBC de una indicación de fecha y hora, pero sólo como un valor de entrada. La representación de serie ODBC de una indicación de fecha y hora tiene el formato *aaaa-mm-dd hh:mm:ss.nnnnnnnnnnnn*, donde el número de dígitos de los segundos fraccionales puede estar comprendido entre 0 y 12..

Valores booleanos

Un valor booleano representa un valor que puede ser verdadero (TRUE) o falso (FALSE). Una expresión o predicado booleano puede producir como resultado un valor desconocido, que se representa como el valor nulo.

El tipo BOOLEAN es un tipo de datos incorporado que solo se puede utilizar 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 el tipo BOOLEAN solo se puede utilizar en sentencias de SQL compuesto (compilado).

Valores de cursor

Un valor de cursor se utiliza para representar una referencia a un cursor subyacente.

El tipo CURSOR es un tipo de datos incorporado que solo se puede utilizar 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 el tipo CURSOR sólo se puede utilizar en sentencias de SQL compuesto (compilado).

Una variable de cursor es una variable de SQL, un parámetro de SQL o una variable global de tipo cursor. Una variable de cursor tiene un cursor subyacente que corresponde al cursor creado para una sentencia SELECT y asignado a esa variable. Varias variables de cursor pueden compartir el mismo cursor subyacente.

Las variables de cursor se pueden utilizar de la misma forma que los cursores de SQL convencionales para realizar iteraciones en un conjunto de resultados de una sentencia SELECT con sentencias OPEN, FETCH y CLOSE.

Valores XML

Un valor XML representa el XML con formato correcto en forma de documento XML, contenido XML o secuencia de nodos XML. Un valor XML que está almacenado en una tabla como valor de una columna definida con el tipo de datos XML debe ser un documento XML con formato correcto. Los valores XML se procesan en una representación interna que no se puede comparar con ningún valor de serie. Un valor XML puede transformarse en un valor de serie serializado que representa el documento XML mediante la función XMLSERIALIZE. Igualmente, un valor de serie que representa un documento XML puede transformarse en un valor XML utilizando la función XMLPARSE. Un valor XML puede analizarse o serializarse implícitamente cuando se intercambia con tipos de datos binarios y de serie de aplicación.

Se aplican restricciones especiales a las expresiones que dan como resultado un valor de tipo de datos XML; dichas expresiones y columnas no están permitidas en (SQLSTATE 42818):

- Una lista SELECT precedida por la cláusula DISTINCT
- Una cláusula GROUP BY
- Una cláusula ORDER BY
- Una subselección de un operador de conjunto que no sea UNION ALL
- Un predicado BETWEEN, IN o LIKE básico y cuantificado
- Una función agregada con DISTINCT

Valores de matriz

Una *matriz* es una estructura que contiene una colección ordenada de elementos de datos en la que se puede hacer referencia a cada elemento por su valor de índice dentro de la colección.La *cardinalidad* de una matriz es el número de elementos que contiene. Todos los elementos de una matriz tienen el mismo tipo de datos.

Una matriz *común* tiene un límite superior definido en relación con el número de elementos, que se conoce como cardinalidad máxima. A cada elemento de la matriz se le hace referencia mediante su posición ordinal como valor de índice. Si *N* es el número de elementos de una matriz común, la posición ordinal asociada con cada elemento es un valor entero mayor que o igual a 1 y menor que o igual a *N*.

Una *matriz asociativa* no tiene ningún límite superior específico respecto al número de elementos. A cada elemento se le hace referencia mediante su valor de índice asociado. El tipo de datos del valor de índice puede ser un valor entero o una serie de caracteres, pero es el mismo tipo de datos para toda la matriz.

La cardinalidad máxima de una matriz común no está relacionada con su representación física, a diferencia de la cardinalidad máxima de las matrices en los lenguajes de programación, como por ejemplo C. Por lo contrario, el sistema utiliza la cardinalidad máxima en tiempo de ejecución para garantizar que los subscripts se encuentren dentro de los límites. La cantidad de memoria necesaria para representar un valor de matriz común no es proporcional a la cardinalidad máxima del tipo correspondiente.

La cantidad de memoria necesaria para representar un valor de matriz suele ser proporcional a la cardinalidad máxima del tipo correspondiente. Cuando se hace referencia a una matriz, todos los valores de la matriz se almacenan en la memoria principal. Por lo tanto, las matrices que contienen una gran cantidad de datos consumirán grandes cantidades de memoria principal.

Tipos anclados

Un tipo anclado define un tipo de datos basado en otro objeto SQL como una columna, una variable global, una variable SQL, un parámetro SQL o la fila de una tabla o una vista.

Un tipo de datos definido mediante una definición de tipo anclado mantiene una dependencia sobre el objeto al que está anclado. Cualquier cambio en el tipo de datos del objeto de anclaje afectará al tipo de datos anclado. Si está anclado a la fila de una tabla o una vista, el tipo de datos anclado es ROW con los campos definidos por las columnas de la tabla o la vista de anclaje.

Tipos definidos por el usuario

Existen seis tipos de tipos de datos definidos por el usuario:

- Tipo diferenciado
- Tipo estructurado
- Tipo de referencia
- Tipo de matriz
- Tipo de fila
- Tipo de cursor

Cada uno de ellos se describe en los apartados siguientes.

Tipo diferenciado

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

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

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

Aunque AUDIO tenga la misma representación que el tipo de datos interno BLOB, se considera un tipo independiente; esto permite la creación de funciones escritas especialmente para AUDIO y asegura que dichas funciones no se aplicarán a valores de ningún otro tipo de datos (imágenes, texto, etc.)

Los tipos diferenciados tienen identificadores calificados. Si no se utiliza el nombre de esquema para calificar el nombre del tipo diferenciado cuando se utiliza en sentencias que no son CREATE TYPE (Diferenciado), DROP o COMMENT, en la vía de acceso de SQL se busca por orden el primer esquema con un tipo diferenciado que coincida.

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

Los tipos diferenciados originados en los tipos LOB están sujetos a las mismas restricciones que su tipo fuente.

Sin embargo, se puede especificar explícitamente que ciertas funciones y ciertos operadores del tipo fuente se apliquen al tipo diferenciado. Esto puede hacerse creando funciones definidas por el usuario que se deriven de funciones definidas en el tipo fuente del tipo diferenciado. Los operadores de comparación se generan automáticamente para tipos diferenciados definidos por el usuario, salvo aquellos que utilizan BLOB, CLOB o DBCLOB como tipo fuente. Además, se generan funciones para dar soporte a la conversión del tipo fuente al tipo diferenciado y del tipo diferenciado al tipo fuente.

Tipo estructurado

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

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

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

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

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

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

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

Un tipo no se puede descartar si ciertos otros objetos utilizan el tipo, ya sea directa o indirectamente. Por ejemplo, no se puede descartar un tipo si una columna de una tabla o vista hace uso directa o indirectamente del tipo.

Tipo de referencia

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

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

Tipo de matriz

Un *tipo de matriz* definido por el usuario es un tipo de datos que se define como matriz con elementos de otro tipo de datos. Cada tipo de matriz común tiene un índice con el tipo de datos INTEGER y tiene definida una cardinalidad máxima. Cada matriz asociativa tiene un índice con el tipo de datos INTEGER o VARCHAR y no tiene definida una cardinalidad máxima.

Tipo de fila

Un *tipo de fila* es un tipo de datos que se define como una secuencia ordenada de campos con nombre, cada uno con un tipo de datos asociado, que representa efectivamente una fila. Un tipo de fila se puede utilizar como tipo de datos para variables y parámetros en SQL PL a fin de proporcionar una manipulación sencilla de una fila de datos.

Tipo de datos del cursor

Un *tipo de cursor* definido por el usuario es un tipo de datos definido por el usuario con la palabra clave CURSOR y opcionalmente con un tipo de fila asociado. Un tipo de cursor definido por el usuario con un tipo de fila asociado es un *tipo de cursor de tipo firme*; de lo contrario, es un *tipo de cursor de tipo no firme*. El valor de un tipo de cursor definido por el usuario representa una referencia a un cursor subyacente.

Promoción de tipos de datos

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

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

- Para realizar la resolución de la función
- Para convertir tipos definidos por el usuario
- Para asignar tipos definidos por el usuario a tipos de datos internos

La Tabla 10 muestra la lista de prioridad (por orden) para cada tipo de datos y se puede utilizar para determinar los tipos de datos a los que se puede promover un tipo de datos determinado. La tabla muestra que la mejor elección siempre es el mismo tipo de datos en lugar de elegir la promoción a otro tipo de datos.

Tabla 10. Tabla de prioridades de tipos de datos

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)
SMALLINT	SMALLINT, INTEGER, BIGINT, decimal, real, doble, DECFLOAT
INTEGER	INTEGER, BIGINT, decimal, real, doble, DECFLOAT
BIGINT	BIGINT, decimal, real, doble, DECFLOAT
decimal	decimal, real, doble, DECFLOAT
real	real, doble, DECFLOAT
doble	doble, DECFLOAT
DECFLOAT	DECFLOAT
CHAR	CHAR, VARCHAR, CLOB
VARCHAR	VARCHAR, CLOB
CLOB	CLOB
GRAPHIC	GRAPHIC, VARGRAPHIC, DBCLOB
VARGRAPHIC	VARGRAPHIC, DBCLOB
DBCLOB	DBCLOB
BLOB	BLOB
DATE	DATE, TIMESTAMP
TIME	TIME
TIMESTAMP	TIMESTAMP
BOOLEAN	BOOLEAN
CURSOR	CURSOR
ARRAY	ARRAY
udt	udt (mismo nombre) o un supertipo de udt
REF(T)	REF(S) (en el caso de que S sea un supertipo de T)
ROW	ROW

Tabla 10. Tabla de prioridades de tipos de datos (continuación)

Tipo de datos	Lista de prioridad de tipos de datos (por orden de mejor a peor)

Nota:

1. Los tipos en minúsculas anteriores se definen de la siguiente manera:
   - decimal = DECIMAL(p,s) o NUMERIC(p,s)
   - real = REAL o FLOAT(*n*), donde *n* not es mayor que 24
   - doble = DOUBLE, DOUBLE-PRECISION, FLOAT o FLOAT(*n*), donde *n* es mayor que 24
   - udt = un tipo definido por el usuario

   Los sinónimos, más cortos o más lagos, de los tipos de datos listados se consideran iguales a la forma listada.
2. Para una base de datos Unicode, los siguientes se consideran tipos de datos equivalentes:
   - CHAR y GRAPHIC
   - VARCHAR y VARGRAPHIC
   - CLOB y DBCLOB

   Al resolver una función en una base de datos de Unicode, si pueden aplicarse a una invocación de función dada tanto una función definida por el usuario como una función incorporada, se invocará generalmente la función incorporada. La función definida por el usuario se invocará únicamente si su esquema es anterior a SYSIBM en el registro especial de CURRENT PATH y si sus tipos de datos de argumento coinciden con todos los tipos de datos de argumento de invocación de función, independientemente de la equivalencia del tipo de datos de Unicode.

Conversiones entre tipos de datos

En muchas ocasiones, un valor con un tipo de datos determinado, necesita *convertirse* a otro tipo de datos diferente o al mismo tipo de datos, aunque con otra longitud, precisión o escala. La promoción del tipo de datos es un ejemplo en el que la promoción de un tipo de datos a otro tipo de datos necesita que el valor se convierta al nuevo tipo de datos. Un tipo de datos que se puede convertir a otro tipo de datos es *convertible* de un tipo de datos fuente al tipo de datos de destino.

La conversión de un tipo de datos a otro tipo de datos puede producirse de forma implícita o explícita. Las funciones de conversión, la especificación CAST o la especificación XMLCAST pueden utilizarse para cambiar de forma explícita a un tipo de datos, según los tipos de datos implicados. Asimismo, cuando se crea una función con fuente definida por el usuario, los tipos de datos de los parámetros de la función fuente deben poder convertirse a los tipos de datos de la función que se está creando.

La Tabla 11 en la página 116 muestra las conversiones permitidas entre tipos de datos internos. La primera columna representa el tipo de datos del operando cast (tipo de datos fuente) y los tipos de datos de la parte superior representan el tipo de datos de destino de la operación de conversión. Una 'S' indica que la especificación CAST se puede utilizar para combinar tipos de datos fuente y de destino. Los casos en los que sólo se puede utilizar la especificación XMLCAST se anotan.

Si se produce un truncamiento al convertir cualquier tipo de datos en un tipo de datos gráficos o de caracteres, se devuelve un aviso si se truncan caracteres distintos de espacios en blanco. Este comportamiento de truncamiento es distinto de la asignación a un tipo de datos de caracteres o gráficos cuando se produce un error si se trunca algún carácter que no es un blanco.

Se da soporte a las siguientes conversiones en las que intervienen tipos diferenciados (con la especificación CAST, a menos que se indique lo contrario):

- Conversión del tipo diferenciado *DT* a su tipo de datos fuente *S*
- Conversión del tipo de datos fuente *S* del tipo diferenciado *DT* al tipo diferenciado *DT*
- Conversión del tipo diferenciado *DT* al mismo tipo diferenciado *DT*
- Conversión de un tipo de datos *A* al tipo diferenciado *DT* donde *A* se puede promocionar al tipo de datos fuente *S* del tipo diferenciado *DT*
- Conversión de INTEGER a un tipo diferenciado *DT* con un tipo de datos fuente SMALLINT
- Conversión de DOUBLE a un tipo diferenciado *DT* con un tipo de datos fuente REAL
- Conversión de DECFLOAT a un tipo diferenciado DT con un tipo de datos fuente de DECFLOAT
- Conversión de VARCHAR a un tipo diferenciado *DT* con un tipo de datos fuente CHAR
- Conversión de VARGRAPHIC a un tipo diferenciado *DT* con un tipo de datos fuente GRAPHIC
- Para una base de datos Unicode, conversión de VARCHAR o VARGRAPHIC a un tipo diferenciado *DT* con un tipo de datos fuente CHAR o GRAPHIC
- Conversión de un tipo diferenciado *DT* con un tipo de datos fuente *S* a XML utilizando la especificación XMLCAST

- Conversión de un XML a un tipo diferenciado *DT* con un tipo de datos fuente de cualquier tipo de datos incorporado, utilizando la especificación XMLCAST, en función del tipo de datos de esquema XML del valor XML

Los tipos de datos FOR BIT DATA no se pueden convertir a CLOB.

En las conversiones que implican un tipo de matriz como destino, el tipo de datos de los elementos del valor de la matriz fuente debe poderse convertir al tipo de datos de los elementos de datos de la matriz de destino (SQLSTATE 42846). Si el tipo de matriz de destino es una matriz común, el valor de la matriz fuente debe ser una matriz común (SQLSTATE 42821) y la cardinalidad del valor de la matriz fuente debe ser igual o menor que la cardinalidad máxima del tipo de datos de la matriz de destino (SQLSTATE 2202F). Si el tipo de matriz de destino es una matriz asociativa, el tipo de datos del índice del valor de matriz fuente se debe poder convertir al tipo de datos del índice del tipo de matriz de destino. Los valores de tipo de matriz definida por el usuario sólo se pueden convertir a otro tipo de matriz definida por el usuario que tenga el mismo nombre (SQLSTATE 42846).

Un tipo de cursor no puede ser el tipo de datos fuente ni el tipo de datos de destino de una especificación CAST, excepto para convertir un marcador de parámetro en un tipo de cursor.

En las conversiones que implican un tipo de fila como destino, el grado de la expresión de valor de la fila fuente debe coincidir con el grado del tipo de fila de destino, y cada campo de la expresión de valor de la fila fuente debe poderse convertir al campo de destino correspondiente. Los valores de tipo de fila definidos por el usuario sólo se pueden convertir a otro tipo de fila definido por el usuario que tenga el mismo nombre (SQLSTATE 42846).

No es posible convertir un valor de tipo estructurado en algo diferente. Un tipo estructurado *ST* no necesita convertirse a uno de sus supertipos, porque todos los métodos de los supertipos de *ST* son aplicables a *ST*. Si la operación deseada sólo es aplicable a un subtipo de *ST*, utilice la expresión de tratamiento de subtipos para tratar *ST* como uno de sus subtipos.

Cuando un tipo de datos definido por el usuario e implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre.

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

- Conversión del tipo de referencia *RT* a su tipo de datos de representación *S*
- Conversión del tipo de datos de representación *S* de un tipo de referencia *RT* al tipo de referencia *RT*
- Conversión de un tipo de referencia *RT* con un tipo de destino *T* a un tipo de referencia *RS* con un tipo de destino *S* donde *S* es un supertipo de *T*.
- Conversión de un tipo de datos *A* en un tipo de referencia *RT* donde *A* se puede promocionar al tipo de datos de representación *S* del tipo de referencia *RT*.

Cuando el tipo de destino de un tipo de datos de referencia implicado en una conversión no está calificado por un nombre de esquema, se utiliza la *vía de acceso de SQL* para buscar el primer esquema que incluya el tipo de datos definido por el usuario con este nombre.

Conversiones entre tipos de datos

Tabla 11. Conversiones soportadas entre tipos de datos internos

	Tipo de datos de destino																				
Tipo de datos fuente	**SMALLINT**	**INTEGER**	**BIGINT**	**DECIMAL**	**REAL**	**DOUBLE**	**DECFLOAT**	**CHAR**	**CHAR FBD[2]**	**VARCHAR**	**VARCHAR FBD[2]**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**DBCLOB**	**BLOB**	**DATE**	**TIME**	**TIMESTAMP**	**XML**	**BOOLEAN**
SMALLINT	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
INTEGER	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
BIGINT	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
DECIMAL	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
REAL	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
DOUBLE	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	S[3]	-
DECFLOAT	S	S	S	S	S	S	S	S	S	S	S	-	S[1]	S[1]	-	-	-	-	-	-	-
CHAR	S	S	S	S	S	S	S	S	S	S	S	S	S[1]	S[1]	S[1]	S	S	S	S	S[4]	-
CHAR FOR BIT DATA	S	S	S	S	S	S	S	S	S	S	S	-	-	-	-	S	S	S	S	S[3]	-
VARCHAR	S	S	S	S	S	S	S	S	S	S	S	S	S[1]	S[1]	S[1]	S	S	S	S	S[4]	-
VARCHAR FOR BIT DATA	S	S	S	S	S	S	S	S	S	S	S	-	-	-	-	S	S	S	S	S[3]	-
CLOB	-	-	-	-	-	-	-	S	-	S	-	S	S[1]	S[1]	S[1]	S	-	-	-	S[4]	-
GRAPHIC	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	-	S[1]	-	S[1]	S	S	S	S	S[1]	S[1]	S[1]	S[3]	-
VARGRAPHIC	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	S[1]	-	S[1]	-	S[1]	S	S	S	S	S[1]	S[1]	S[1]	S[3]	-
DBCLOB	-	-	-	-	-	-	-	S[1]	-	S[1]	-	S[1]	S	S	S	S	-	-	-	S[3]	-
BLOB	-	-	-	-	-	-	-	-	S	-	S	-	-	-	-	S	-	-	-	S[4]	-
DATE	-	S	S	S	-	-	-	S	S	S	S	-	S[1]	S[1]	-	-	S	-	S	S[3]	-
TIME	-	S	S	S	-	-	-	S	S	S	S	-	S[1]	S[1]	-	-	-	S	-	S[3]	-
TIMESTAMP	-	-	S	S	-	-	-	S	S	S	S	-	S[1]	S[1]	-	-	S	S	S	S[3]	-
XML	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S[5]	S	-
BOOLEAN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Y[7]

Tabla 11. Conversiones soportadas entre tipos de datos internos (continuación)

	Tipo de datos de destino																					
Tipo de datos fuente	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	DECFLOAT	CHAR	CHAR FBD[2]	VARCHAR	VARCHAR FBD[2]	CLOB	GRAPHIC	VARGRAPHIC	DBCLOB	BLOB	DATE	TIME	TIMESTAMP	XML	BOOLEAN
Notas																						

- Vea la descripción que precede a la tabla para conocer las conversiones soportadas donde intervienen tipos definidos por el usuario y tipos de referencia.
- No es posible convertir un valor de tipo estructurado en algo diferente.
- Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos, no son recomendables y se pueden eliminar en un release futuro.

[1] Conversión sólo soportada para bases de datos Unicode.

[2] FOR BIT DATA

[3] La conversión sólo se puede efectuar utilizando XMLCAST.

[4] Se procesa implícitamente una función XMLPARSE para convertir una serie a XML al asignarse (INSERT o UPDATE) una seria a una columna XML. La serie tiene que ser un documento XML bien formado para que la asignación tenga éxito.

[5] La conversión sólo se puede efectuar utilizando XMLCAST y depende del tipo de datos del esquema XML subyacente del valor XML. Para obtener detalles, consulte “XMLCAST”.

[6] Un tipo de cursor no puede ser el tipo de datos fuente ni el tipo de datos de destino de una especificación CAST, excepto para convertir un marcador de parámetro en un tipo de cursor.

[7] Sólo recibe soporte mediante la utilización de la especificación CAST. No existe ninguna función de conversión.

En la tabla Tabla 12 se muestra dónde puede encontrar información sobre las normas que se aplican durante la conversión a los tipos de datos de destino identificados.

Tabla 12. Normas para convertir a un tipo de datos

Tipo de datos de destino	Normas
SMALLINT	“Función escalar SMALLINT” en el manual *Consulta de SQL, Volumen 1*
INTEGER	“Función escalar INTEGER” en el manual *Consulta de SQL, Volumen 1*
BIGINT	“Función escalar BIGINT” en el manual *Consulta de SQL, Volumen 1*
DECIMAL	“Función escalar DECIMAL” en el manual *Consulta de SQL, Volumen 1*
NUMERIC	“Función escalar DECIMAL” en el manual *Consulta de SQL, Volumen 1*

Tabla 12. Normas para convertir a un tipo de datos (continuación)

Tipo de datos de destino	Normas
REAL	"Función escalar REAL" en el manual *Consulta de SQL, Volumen 1*
DOUBLE	"Función escalar DOUBLE" en el manual *Consulta de SQL, Volumen 1*
DECFLOAT	"Función escalar DECFLOAT" en el manual *Consulta de SQL, Volumen 1*
CHAR	"Función escalar CHAR" en el manual *Consulta de SQL, Volumen 1*
VARCHAR	"Función escalar VARCHAR" en el manual *Consulta de SQL, Volumen 1*
CLOB	"Función escalar CLOB" en el manual *Consulta de SQL, Volumen 1*
GRAPHIC	"Función escalar GRAPHIC" en el manual *Consulta de SQL, Volumen 1*
VARGRAPHIC	"Función escalar VARGRAPHIC" en el manual *Consulta de SQL, Volumen 1*
DBCLOB	"Función escalar DBCLOB" en el manual *Consulta de SQL, Volumen 1*
BLOB	"Función escalar BLOB" en el manual *Consulta de SQL, Volumen 1*
DATE	"Función escalar DATE" en el manual *Consulta de SQL, Volumen 1*
TIME	"Función escalar TIME" en el manual *Consulta de SQL, Volumen 1*
TIMESTAMP	Si el tipo de fuente es una serie de caracteres, consulte el apartado "Función escalar TIMESTAMP" en el manual *Consulta de SQL, Volumen 1* , donde se especifica un operando. Si el tipo de datos fuente es DATE, la indicación de fecha y hora estará compuesta por la fecha y hora especificadas de 00:00:00.

Conversiones de valores que no son XML a valores XML

Tabla 13. Conversiones soportadas desde valores que no son XML a valores XML

	Tipo de datos de destino	
Tipo de datos fuente	**XML**	**Tipo de esquema XML obtenido**
SMALLINT	S	xs:short
INTEGER	S	xs:int
BIGINT	S	xs:long
DECIMAL o NUMERIC	S	xs:decimal
REAL	S	xs:float
DOUBLE	S	xs:double
DECFLOAT	N	-
CHAR	S	xs:string

Tabla 13. Conversiones soportadas desde valores que no son XML a valores XML (continuación)

	Tipo de datos de destino	
Tipo de datos fuente	**XML**	**Tipo de esquema XML obtenido**
VARCHAR	S	xs:string
CLOB	S	xs:string
GRAPHIC	S	xs:string
VARGRAPHIC	S	xs:string
DBCLOB	S	xs:string
DATE	S	xs:date
TIME	S	xs:time
TIMESTAMP	S	xs:dateTime[1]
BLOB	S	xs:base64Binary
Tipo de caracteres FOR BIT DATA	S	xs:base64Binary
tipo diferenciado		utilice esta tabla con el tipo fuente del tipo diferenciado

Notas

[1] El tipo de datos fuente TIMESTAMP da soporte a la precisión de indicación de fecha y hora de 0 a 12. La precisión máxima de los segundos fraccionarios de xs:dateTime es 6. Si la precisión de la indicación de fecha y hora de un tipo de datos fuente TIMESTAMP sobrepasa 6, el valor se trunca cuando se convierte a xs:dateTime.

Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos, no son recomendables y se pueden eliminar en un release futuro.

Cuando los valores de una serie de caracteres se convierten a valores XML, el valor elemental xs:string obtenido no puede contener caracteres XML ilegales (SQLSTATE 0N002). Si la serie de caracteres de entrada no está en Unicode, dichos caracteres se convertirán a Unicode.

La conversión a tipos binarios de SQL da como resultado valores XQuery elementales con el tipo xs:base64Binary.

Conversión de valores XML a valores que no son XML

Un XMLCAST desde un valor XML a un valor que no es XML se puede describir como de dos conversiones: una conversión XQuery que convierte el valor XML fuente a un tipo XQuery correspondiente al tipo de destino SQL, seguido de una conversión desde el tipo XQuery correspondiente al tipo SQL real.

Se considera que XMLCAST está soportado si el tipo de destino tiene un tipo de destino XQuery correspondiente soportado y si hay una conversión XQuery soportada del tipo de valor fuente al tipo de destino XQuery correspondiente. El tipo de destino utilizado en la conversión XQuery está basado en el tipo de destino XQuery correspondiente y puede contener restricciones adicionales.

La tabla siguiente lista los tipos XQuery que se obtienen de dicha conversión.

Tabla 14. Conversiones soportadas de valores XML a valores que no son XML

	Tipo de datos fuente	
Tipo de datos de destino	**XML**	**Tipo de destino XQuery correspondiente**
SMALLINT	S	xs:short
INTEGER	S	xs:int
BIGINT	S	xs:long
DECIMAL o NUMERIC	S	xs:decimal
REAL	S	xs:float
DOUBLE	S	xs:double
DECFLOAT	S	sin tipo coincidente[1]
CHAR	S	xs:string
VARCHAR	S	xs:string
CLOB	S	xs:string
GRAPHIC	S	xs:string
VARGRAPHIC	S	xs:string
DBCLOB	S	xs:string
DATE	S	xs:date
TIME (sin huso horario)	S	xs:time
TIMESTAMP (sin huso horario)	S	xs:dateTime[2]
BLOB	S	xs:base64Binary
CHAR FOR BIT DATA	N	no convertible
VARCHAR FOR BIT DATA	S	xs:base64Binary
tipo diferenciado		utilice esta tabla con el tipo fuente del tipo diferenciado
fila, referencia, tipo de datos estructurado o abstracto (ADT), otros	N	no convertible

Notas

[1] DB2 da soporte a XML Schema 1.0, que no proporciona un tipo de esquema XML coincidente para DECFLOAT. El proceso del paso de conversión de XQuery de XMLCAST se maneja del siguiente modo:

- Si el valor fuente se escribe con un tipo numérico de esquema XML, utilice dicho tipo numérico.
- Si el valor fuente se escribe con un tipo de esquema XML xs:boolean, utilice xs:double.
- En caso contrario, utilice xs:string con comprobación adicional para un formato numérico válido.

[2] la precisión máxima de los segundos fraccionarios de xs:dateTime es 6. El tipo de datos fuente TIMESTAMP da soporte a la precisión de indicación de fecha y hora de 0 a 12. Si la precisión de la indicación de fecha y hora de un tipo de datos de destino TIMESTAMP es menor que 6, el valor se trunca cuando se convierte de xs:dateTime. Si la precisión de la indicación de fecha y hora de un tipo de datos de destino TIMESTAMP sobrepasa 6, el valor se rellena con ceros cuando se convierte de xs:dateTime.

En los casos de restricción siguientes, se utiliza un tipo de datos del esquema XML derivado por restricción, como tipo de datos de destino para la conversión XQuery.

- Los valores XML que se van a convertir en tipos de series deben caber dentro de los límites de longitud de dichos tipos de DB2 sin que los caracteres o bytes se trunquen. El nombre utilizado para el tipo de esquema XML derivado es el nombre del tipo SQL en mayúsculas seguido de un carácter subrayado y la longitud máxima de la serie; por ejemplo, VARCHAR_20 si el tipo de datos de destino XMLCAST es VARCHAR(20).
- Los valores XML que deben convertirse en valores DECIMAL deben ajustarse a la precisión de los valores DECIMAL especificados, y no deben contener, después de la coma decimal, un número de dígitos que no sean cero mayor que la escala. El nombre utilizado para el tipo de esquema XML derivado es DECIMAL_*precisión_escala*, donde *precisión* es la precisión de los tipos de datos SQL de destino y *escala* es la escala de los tipos de datos SQL de destino; por ejemplo, DECIMAL_9_2 si el tipo de datos de destino XMLCAST es DECIMAL(9,2).
- Los valores XML que deben convertirse a valores TIME no pueden contener un componente de segundos con dígitos que no sean cero después de la coma decimal. El nombre utilizado para el tipo de esquema XML derivado es TIME.

El nombre del tipo de esquema XML derivado sólo aparece en un mensaje si un valor XML no cumple alguna de estas restricciones. Este nombre de tipo ayuda a comprender el mensaje de error y no corresponde a ningún tipo de XQuery definido. Si el valor de entrada no cumple el tipo de base del tipo de esquema XML derivado (el tipo de destino XQuery correspondiente), el mensaje de error puede indicar en su lugar este tipo. Dado que este formato de nombre de tipo de esquema XML puede cambiar en el futuro, no debería utilizarse como interfaz de programación.

Antes de que la conversión XQuery procese un valor XML, se eliminarán todos los nodos de documento de la secuencia y cada hijo directo de nodo del documento eliminado se convertirá en un elemento de la secuencia. Si el nodo del documento tiene varios nodos hijo directos, la secuencia revisada tendrá más elementos que la secuencia original. El valor XML sin ningún tipo de nodo de documento se atomiza mediante la función XQuery fn:data y se utiliza el valor de secuencia atomizada resultante en la conversión XQuery. Si la secuencia atomizada es una secuencia vacía, la conversión devuelve un valor nulo sin ningún proceso posterior. Si el valor de la secuencia atomizada contiene varios elementos, se devolverá un error (SQLSTATE 10507).

Si el tipo de destino de XMLCAST es el tipo de datos DATE, TIME o TIMESTAMP de SQL, el valor XML obtenido de la conversión XQuery también se ajustará a UTC y se eliminará el componente de huso horario del valor.

Cuando el valor del tipo de destino XQuery correspondiente se convierte al tipo de destino SQL; los tipos de datos XML binarios como xs:base64Binary o xs:hexBinary, se convierten del formato de carácter a datos binarios reales.

Si un valor xs:double o xs:float de INF, -INF o NaN se convierte (mediante XMLCAST) a un valor DOUBLE o REAL de tipo de datos SQL, se devuelve un error (SQLSTATE 22003). Un valor xs:double o xs:float de -0 se convierte a +0.

El tipo de destino puede ser un tipo diferenciado definido por el usuario, si el operando fuente no es un tipo diferenciado definido por el usuario. En dicho caso, el valor fuente se convierte en el tipo fuente del tipo diferenciado definido por el

usuario (es decir, el tipo de destino) mediante la especificación XMLCAST y, a continuación, este valor se convierte en el tipo diferenciado definido por el usuario mediante la especificación CAST.

En una base de datos no Unicode, la conversión de un valor XML a un tipo de destino no XML implica la conversión de página de códigos de un formato UTF-8 interno a la página de códigos de la base de datos. La conversión dará como resultado la entrada de los caracteres de sustitución en caso de que algún elemento de código del valor XML no esté presente en la página de códigos de la base de datos.

Asignaciones y comparaciones

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

Una norma básica para las dos operaciones es que el tipo de datos de los operandos implicados debe ser compatible. La norma de compatibilidad también se aplica a las operaciones de conjuntos.

Otra norma básica para las operaciones de asignación es que no pueda asignarse un valor nulo a una columna que no pueda contener valores nulos, ni a una variable del lenguaje principal que no tenga una variable indicadora asociada.

A continuación se presenta una matriz de compatibilidad que muestra las compatibilidades de tipos de datos definidas por el sistema para operaciones de asignación y comparación.

Tabla 15. Compatibilidad de tipos de datos para asignaciones y comparaciones

Operandos	Entero binario	Número decimal	Coma flotante	Coma flotante decimal	Serie de caracteres	Serie gráfica	Booleano	Fecha	Hora	Indicación de fecha y hora	Serie binaria	UDT
Entero binario	Sí	Sí	Sí	Sí	Sí	Sí [5]	No	No	No	No	No	[2]
Número decimal	Sí	Sí	Sí	Sí	Sí	Sí [5]	No	No	No	No	No	[2]
Coma flotante	Sí	Sí	Sí	Sí	Sí	Sí [5]	No	No	No	No	No	[2]
Coma flotante decimal	Sí	Sí	Sí	Sí	Sí	Sí [5]	No	No	No	No	No	[2]
Serie de caracteres	Sí	Sí	Sí	Sí	Sí	Sí [5,6]	No	Sí	Sí	Sí	No [3]	[2]
Serie gráfica	Sí [5]	Sí [5]	Sí [5]	Sí [5]	Sí [5,6]	Sí	No	Sí [5]	Sí [5]	Sí [5]	No	[2]
Serie binaria	No	No	No	No	Sí[3]	No	No	No	No	No	Sí	[2]
Fecha	No	No	No	No	Sí	Sí [5]	No	Sí	No	Sí	No	[2]
Hora	No	No	No	No	Sí	Sí [5]	No	No	Sí	[1]	No	[2]
Indicación de la hora	No	No	No	No	Sí	Sí [5]	No	Sí	[1]	Sí	No	[2]
Booleano	No	No	No	No	No	No	Sí[7]	No	No	No	No	[2]
UDT	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	Sí

Tabla 15. Compatibilidad de tipos de datos para asignaciones y comparaciones (continuación)

Operandos	Entero binario	Número decimal	Coma flotante	Coma flotante decimal	Serie de caracteres	Serie gráfica	Booleano	Fecha	Hora	Indicación de fecha y hora	Serie binaria	UDT

[1] Puede asignarse un valor TIMESTAMP a un valor TIME; sin embargo, no puede asignarse un valor TIME a un valor TIMESTAMP, y éste, a su vez, no puede compararse con un valor TIME.

[2] Un valor de tipo diferenciado definido por el usuario sólo se puede comparar con un valor definido con el mismo tipo diferenciado definido por el usuario. En general, se da soporte a las asignaciones entre un valor de tipo diferenciado y su tipo de datos fuente. Un tipo estructurado definido por el usuario no es comparable y sólo se puede asignar a un operando del mismo tipo estructurado o a uno de sus subtipos. Algunas normas de asignación adicionales se aplican a tipos de filas, cursores y matrices. Para obtener más información, vea "Asignaciones de tipos definidos por el usuario" en la página 131.

[3] Soporte sólo para asignación (no para comparación) y sólo para series de caracteres definidas como FOR BIT DATA.

[4] Para obtener información sobre la asignación y comparación de tipos de referencia, consulte "Asignación de tipos de referencia" en la página 133 y "Comparaciones de tipos de referencia" en la página 139.

[5] Sólo soportado para bases de datos Unicode.

[6] Las series de datos de bits y gráficas no son compatibles.

[7] Las variables del tipo de datos booleanos no pueden compararse directamente; la comparación sólo puede realizarse con los valores literales TRUE, FALSE, NULL.

Asignaciones numéricas

Para las asignaciones numéricas, no se permite el desbordamiento.

- Cuando se realizan asignaciones a un tipo de datos numéricos exacto, se produce el desbordamiento si se elimina cualquier dígito de la parte entera del número. Si es necesario, se trunca la parte de fracción de un número.
- Cuando se realizan asignaciones a un tipo de datos numéricos aproximado o coma flotante decimal, se produce el desbordamiento si se elimina el dígito más significativo de la parte entera del número. Para números de coma flotante y de coma flotante decimal, la parte entera del número es el número que se obtendrá si el número de coma flotante o de coma flotante decimal se convierte a un número decimal con precisión ilimitada. Si es necesario, el redondeo puede hacer que se eliminen los dígitos menos significativos del número.

  Para los números de coma flotante decimal, está permitido truncar toda la parte del número y el resultado es un valor infinito con un aviso.

  Para números de coma flotante, tampoco se permite el subdesbordamiento. El subdesbordamiento se produce para números entre 1 y -1 si se elimina el dígito más significativo distinto de cero. Para la coma flotante decimal, el subdesbordamiento se permite y, en función de la modalidad de redondeo, produce cero o el número positivo más pequeño o el número negativo más grande que se puede representar junto con un aviso.

  Se devuelve un aviso de desbordamiento o subdesbordamiento en lugar de un error si se produce un desbordamiento o un subdesbordamiento en la asignación a una variable del lenguaje principal con una variable de indicador. En este caso, el número no se asigna a la variable de lenguaje principal y la variable de indicador se establece en 2 negativo.

Para números de coma flotante decimal, el registro especial CURRENT DECFLOAT ROUNDING MODE indica la modalidad de redondeo en vigor.

Asignaciones a entero

Cuando se asigna un número decimal, de coma flotante o de coma flotante decimal a una columna o variable de enteros, se elimina la parte de fracción del número. Como resultado, un número entre 1 y -1 se reduce a 0.

Asignaciones a decimal

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

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

Cuando se asigna un número de coma flotante a una columna o variable decimal, primero el número se convierte a un número decimal temporal de precisión 31 y, a continuación, si es necesario, se trunca a la precisión y escala del destino. En esta conversión, el número se redondea (utilizando la aritmética de coma flotante) a una precisión de 31 dígitos decimales. Como resultado, un número entre 1 y -1 menor que el número positivo más pequeño o mayor que el número negativo más grande que se pueda representar en la columna o variable decimal se reduce a 0. Se da a la escala el valor más grande posible que permita representar la parte entera del número sin pérdida de significación.

Cuando se asigna un número de coma flotante decimal a una columna o variable decimal, el número se redondea a la precisión y escala de la columna o variable decimal. Como resultado, un número entre 1 y -1 que sea menor que el número positivo más pequeño o mayor que el número negativo más grande que se pueda representar en la columna o variable decimal se reduce a 0 o se redondea al valor negativo más grande o positivo más pequeño que se pueda representar en la columna o variable decimal, en función de la modalidad de redondeo.

Asignaciones a coma flotante

Los números de coma flotante son aproximaciones de números reales. Por lo tanto, cuando se asigna un número entero, decimal, de coma flotante o de coma flotante decimal a una columna o variable de coma flotante, es posible que el resultado no sea idéntico al número original. El número se redondea a la precisión de la columna o variable de coma flotante utilizando aritmética de coma flotante. Un valor de coma flotante decimal se convierte en primer lugar a una representación de serie y, a continuación, se convierte a un número de coma flotante.

Asignaciones a coma flotante decimal

Cuando se asigna un número entero a una columna o variable de coma flotante decimal, el número se convierte primero a un número decimal temporal y, a continuación, a un número de coma flotante decimal. La precisión y escala del número decimal temporal es 5,0 para un entero pequeño, 11,0 para un entero grande o 19,0 para un entero superior. El redondeo se puede producir al asignar un BIGINT a una variable o columna DECFLOAT(16).

Cuando se asigna un número decimal a una variable o columna de coma flotante decimal, el número se convierte a la precisión (16 o 34) del destino. Se eliminan los ceros iniciales. En función de la precisión y escala del número decimal y la precisión del destino, es posible que se redondee el valor.

Cuando se asigna un número de coma flotante a una variable o columna de coma flotante decimal, el número se convierte primero a una representación de serie temporal del número de coma flotante. La representación de serie del número se convierte a continuación a coma flotante decimal.

Cuando se asigna un número DECFLOAT(16) a una variable o columna DECFLOAT(34), el valor resultante es idéntico al número DECFLOAT(16).

Cuando se asigna un número DECFLOAT(34) a una columna o variable DECFLOAT(16), el exponente de la fuente se convierte al exponente correspondiente del formato de resultados. La mantisa del número DECFLOAT(34) se redondea a la precisión del destino.

Asignaciones de series a numérico

Cuando se asigna una serie a un tipo de datos numérico, se convierte al tipo de datos numérico de destino utilizando las normas de una especificación CAST. Para obtener más información, consulte "Especificaciones CAST" en la publicación *Consulta de SQL, Volumen 1*.

Asignaciones de series

Existen dos tipos de asignaciones:

- En la *asignación de almacenaje*, se asigna un valor y no es conveniente el truncamiento de datos significativos; por ejemplo, cuando se asigna un valor a una columna
- En la *asignación de recuperación*, se asigna un valor y se permite truncamiento; por ejemplo, cuando se recuperan datos de la base de datos

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

Asignación de almacenamiento

La norma básica es que la longitud de la serie asignada al destino no debe ser mayor que el atributo de longitud del destino. Si la longitud de la serie es mayor que el atributo de longitud del destino, se pueden producir las siguientes situaciones:

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

Si se asigna una serie a un destino de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos de un solo byte, de doble byte o UCS-2. El carácter de relleno siempre es el blanco, incluso para las columnas definidas con el atributo FOR BIT DATA. (UCS-2 define varios caracteres SPACE con distintas propiedades. Para una base de datos Unicode, el gestor de bases de datos siempre utilizará ASCII SPACE en la posición x'0020' como blanco UCS-2. Para una base de datos EUC, se utiliza IDEOGRAPHIC SPACE en la posición x'3000' para rellenar las series GRAPHIC.)

Asignación de recuperación

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

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

Si se asigna una serie de caracteres a un destino de longitud fija y la longitud de la serie es menor que el atributo de longitud del destino, la serie se rellena por la derecha con el número necesario de blancos de un solo byte, de doble byte o UCS-2. El carácter de relleno siempre es un blanco, incluso para las series definidas con el atributo FOR BIT DATA. (UCS-2 define varios caracteres SPACE con distintas propiedades. Para una base de datos Unicode, el gestor de bases de datos siempre utilizará ASCII SPACE en la posición x'0020' como blanco UCS-2. Para una base de datos EUC, se utiliza IDEOGRAPHIC SPACE en la posición x'3000' para rellenar las series GRAPHIC.)

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C se maneja en base a las opciones especificadas con los mandatos PREP o BIND.

Normas de conversión para asignación de series

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

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

Para las bases de datos Unicode, las series de caracteres pueden asignarse a una columna gráfica y las series gráficas pueden asignarse a una columna de caracteres.

Consideraciones de MBCS para la asignación de las series de caracteres

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

- El relleno con blancos siempre se realiza utilizando el carácter blanco de un solo byte (X'20').
- El truncamiento de blancos siempre se realiza en base al carácter blanco de un solo byte (X'20'). El carácter blanco de doble byte se trata como cualquier otro carácter con respecto al truncamiento.
- La asignación de una serie de caracteres a una variable del lenguaje principal puede dar como resultado la fragmentación de caracteres MBCS si la variable del lenguaje principal de destino no es lo suficientemente grande para contener

toda la serie fuente. Si se fragmenta un carácter MBCS, cada byte del fragmento del carácter MBCS destino se establece en el destino en un carácter blanco de un solo byte (X'20'), no se mueven más bytes de la fuente y SQLWARN1 se establece en 'W' para indicar el truncamiento. Observe que se aplica el mismo manejo de fragmentos de caracteres MBCS incluso cuando la serie de caracteres está definida como FOR BIT DATA.

Consideraciones de DBCS para la asignación de las series gráficas

Las asignaciones de series gráficas se procesan de manera análoga a la de las series de caracteres. Para las bases de datos que no son Unicode, los tipos de datos de serie gráfica sólo son compatibles con otros tipos de datos de serie gráfica y nunca con tipos de datos numéricos, de serie de caracteres o de indicación de fecha y hora. Para las bases de Unicode, los tipos de datos de serie gráfica son compatibles con tipos de datos de serie de caracteres. No obstante, los tipos de datos de serie de caracteres o de serie gráfica no se pueden intercambiar en la sentencia SELECT INTO o en VALUES INTO.

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

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

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

La asignación de recuperación de las variables del lenguaje principal terminadas en nulo en C (declaradas utilizando wchar_t) se maneja sobre la base de las opciones especificadas con los mandatos PREP o BIND.

Asignaciones de numérico a series

Cuando se asigna un número a un tipo de datos de serie, se convierte al tipo de datos de serie de destino utilizando las normas de una especificación CAST. Para obtener más información, consulte "Especificaciones CAST" en la publicación *Consulta de SQL, Volumen 1*.

Si durante la conversión de un valor numérico a un tipo de datos gráficos o de caracteres se trunca un carácter que no es un blanco, se devuelve un aviso. Este comportamiento de truncamiento es distinto de la asignación a un tipo de datos de

caracteres o de gráficos que sigue las normas de asignación de almacenamiento, ya que si durante la asignación se trunca un carácter que no es un blanco, se devuelve un error.

Asignaciones de fecha y hora

Un valor TIME sólo puede asignarse a una columna TIME o a una variable de serie o columna de serie.

Puede asignarse un valor de DATE a un tipo de datos DATE, TIMESTAMP o de serie. Cuando se asigna un valor de DATE a un tipo de datos TIMESTAMP, se asume que la información de hora que falta son todo ceros.

Puede asignarse un valor de TIMESTAMP a un tipo de datos DATE, TIME, TIMESTAMP o de serie. Cuando un valor TIMESTAMP se asigna a un tipo de datos DATE, se extrae la parte correspondiente a la fecha y se trunca la parte correspondiente a la hora. Cuando se asigna un valor TIMESTAMP a un tipo de datos TIME, la parte correspondiente a la fecha se pasa por alto y la parte correspondiente a la hora se extrae, pero con los segundos fraccionarios truncados. Cuando se asigna un valor de TIMESTAMP a una TIMESTAMP con una precisión inferior, los segundos fraccionarios sobrantes se truncan. Cuando se asigna un valor de TIMESTAMP a una TIMESTAMP con una precisión superior, se asume que los dígitos que faltan son ceros.

La asignación no debe hacerse a una variable o columna CLOB, DBCLOB o BLOB.

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

Cuando el destino no es una variable del lenguaje principal y tiene un tipo de datos de caracteres, no se permite el truncamiento. El atributo de longitud de la columna debe ser, como mínimo, 10 para una fecha, 8 para una hora, 19 para TIMESTAMP(0) y 20+*p* para TIMESTAMP(*p*).

Si el destino es una variable del lenguaje principal de serie, se aplican las normas siguientes:

- **Para DATE:** Si la longitud de la variable del lenguaje principal es de menos de 10 caracteres, se devuelve un error.
- **Para TIME:** Si se utiliza el formato USA, la longitud de la variable de lenguaje principal no debe tener menos de 8 caracteres; en otros formatos, la longitud no debe tener menos de 5 caracteres.

  Si se utilizan los formatos ISO o JIS, y la longitud de la variable del lenguaje principal es menor que 8 caracteres, la parte correspondiente a los segundos de la hora se omite del resultado y se asigna a la variable indicadora, si se proporciona. El campo SQLWARN1 de la SQLCA se establece de manera que indica la omisión.
- **Para TIMESTAMP:** Si la longitud de la variable del lenguaje principal es de menos de 19 caracteres, se devuelve un error. Si la longitud es inferior a 32 caracteres, pero superior o igual a 19 caracteres, se omiten los dígitos finales de

la parte de los segundos fraccionarios del valor. El campo SQLWARN1 de SQLCA se establece para indicar la omisión.

Si se asigna un valor de DATE a TIMESTAMP, la hora y los componentes fraccionarios de la indicación de fecha y hora se establecen en la medianoche y en 0, respectivamente. Si se asigna un valor de TIMESTAMP a DATE, se extrae la parte correspondiente a la fecha y la hora y los componentes fraccionarios se truncan.

Si se asigna un valor de TIMESTAMP a TIME, la parte correspondiente a DATE se pasa por alto y los componentes fraccionarios se truncan.

Asignaciones de XML

La norma general que rige las asignaciones de XML es que sólo se puede asignar un valor XML a las columnas o a las variables XML. A continuación, indicamos las excepciones a la norma.

- ***Proceso de las variables de lenguaje principal XML de entrada:*** Caso especial de la norma de asignación de XML, ya que la variable de lenguaje principal está basada en el valor de una serie. Para realizar la asignación de XML dentro de SQL, el valor de la serie se analiza implícitamente en un valor XML utilizando el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION. Esto permite determinar si se conservarán o se eliminarán los espacios en blanco, a menos que la variable del lenguaje principal sea un argumento de la función XMLVALIDATE, que siempre elimina los espacios en blanco innecesarios.
- ***Asignación de series a marcadores de parámetro de entrada de tipo de datos XML:*** si un marcador de parámetro de entrada tiene un tipo de datos implícito o explícito de XML, el valor vinculado (asignado) a ese marcador de parámetro puede ser una variable de serie de caracteres, una variable de serie de gráficos o una variable de serie binaria. En dicho caso, el valor de serie se analiza implícitamente en un valor XML utilizando el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION para determinar si se conservarán o se eliminarán los espacios en blanco, a menos que el marcador de parámetros sea un argumento de la función XMLVALIDATE, que siempre elimina los espacios en blanco innecesarios.
- ***Asignación directa de series a columnas XML en sentencias de cambio de datos:*** Si se realizan asignaciones directas a una columna del tipo XML en una sentencia de cambio de formato, la expresión asignada podrá ser también una serie de caracteres o una serie binaria. En dicho caso, el resultado de `XMLPARSE (DOCUMENT` *`expresión`* `STRIP WHITESPACE)` se asignará a la columna de destino. Los tipos de datos de serie soportados se definen mediante los argumentos soportados para la función XMLPARSE. Tenga en cuenta que esta excepción de la asignación de XML no permite asignar valores de serie de caracteres o binarios a variables SQL o a parámetros de SQL del tipo de datos XML.
- ***Asignación de XML de recuperación a series:*** Si se están recuperando valores XML en variables del lenguaje principal utilizando una sentencia FETCH INTO o una sentencia EXECUTE INTO en SQL incorporado, los tipos de datos de la variable del lenguaje principal pueden ser CLOB, DBCLOB o BLOB. Si se utilizan otras interfaces de programación de aplicaciones (como CLI, JDBC o .NET), los valores XML se podrán recuperar en tipos de series binarias, gráficas o de caracteres soportados por la interfaz de programación de aplicaciones. En todos estos casos, el valor XML se serializa implícitamente en una serie codificada en UTF-8 y, en el caso de las variables de series de caracteres o gráficas, se convierte en la página de códigos del cliente.

Los valores de las series binarias o de caracteres no se pueden recuperar en variables del lenguaje principal XML. Los valores de las variables del lenguaje principal XML no pueden asignarse a columnas, variables SQL ni parámetros SQL de un tipo de datos de serie de caracteres ni de un tipo de datos de serie binaria.

La asignación a variables y parámetros XML en UDF incorporadas a SQL en línea y procedimientos de SQL se realiza por referencia. Los parámetros del tipo de datos XML para invocar una UDF SQL en línea o un procedimiento de SQL también se pasan por referencia. 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.

Asignaciones de tipos definidos por el usuario

Para los tipos diferenciados y los tipos estructurados, se aplican normas para las asignaciones a variables del lenguaje principal diferentes a las normas que se utilizan para todas las demás asignaciones.

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

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

Si el destino de la asignación es una columna basada en un tipo diferenciado, el tipo de datos fuente debe ser convertible al tipo de datos de destino.

Tipos estructurados: La asignación realizada con variables del lenguaje principal está basada en el tipo declarado de la variable del lenguaje principal; es decir, sigue la norma siguiente:

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

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

Para los tipos de matriz, se aplican normas diferentes para las asignaciones a variables de SQL y parámetros. La validez de una asignación a una variable SQL o parámetro se determina en función de las normas siguientes:

- Si la parte derecha de la asignación es una variable SQL o un parámetro, una llamada a la función TRIM_ARRAY, una llamada a la función ARRAY_DELETE o una expresión CAST, su tipo debe ser el mismo que el tipo de la variable o el parámetro SQL situado a la izquierda de la asignación.
- Si la parte derecha de la asignación es un constructor de matriz o a una llamada a la función ARRAY_AGG, se convierte implícitamente al tipo de la variable o parámetro SQL de la parte izquierda.

Por ejemplo, suponiendo que el tipo de variable *V* es MYARRAY, la sentencia:

```
SET V = ARRAY[1,2,3];
```

es equivalente a:

```
SET V = CAST(ARRAY[1,2,3] AS MYARRAY);
```

Y la sentencia:

```
SELECT ARRAY_AGG(C1) INTO V FROM T
```

es equivalente a:

```
SELECT CAST(ARRAY_AGG(C1) AS MYARRAY) INTO V FROM T
```

En los apartados siguientes encontrará información adicional sobre los tipos específicos definidos por el usuario:

Asignaciones de tipo de matriz

El valor de un elemento de una matriz se debe poder asignar al tipo de datos de los elementos de la matriz. Las normas de asignación de este tipo de datos se aplican a la asignación de valores. El valor especificado para un índice de la matriz se debe poder asignar al tipo de datos del índice de la matriz. Las normas de asignación de este tipo de datos se aplican a la asignación de valores. En una matriz común, el tipo de datos del índice es INTEGER; en una matriz asociativa, el tipo de datos es INTEGER o VARCHAR(n), siendo n cualquier atributo de longitud válido para el tipo de datos VARCHAR. Si el valor de índice de una asignación a una matriz común es mayor que la cardinalidad actual de la matriz, la cardinalidad de la matriz aumentará al nuevo valor de índice, siempre que el valor no exceda el valor máximo de un tipo de datos INTEGER. La asignación de un elemento nuevo a una matriz asociativa aumenta la cardinalidad en 1 exactamente, dado que los valores del índice pueden ser sparse.

A continuación se muestran asignaciones válidas relacionadas con valores de tipo de matriz:

- Variable de matriz a otra variable de matriz con el mismo tipo de matriz como variable fuente.
- Una expresión de matriz de tipos a una variable de matriz, donde el tipo de elemento de matriz de la expresión fuente puede asignarse al tipo de elemento de matriz de la variable de matriz de destino.

Asignaciones de tipos de fila

Las asignaciones a campos de una variable de fila deben respetar las mismas normas que si el propio campo fuera una variable del mismo tipo de datos que el campo. Una variable de fila sólo puede asignarse a una variable de fila con el mismo tipo de fila definido por el usuario. Cuando se utiliza FETCH, SELECT o VALUES INTO para asignar valores a una variable de fila, los tipos de valor fuente deben poder asignarse a los campos de fila de destino.Si la variable fuente o de destino (o ambas) de una asignación está anclada a la fila de una tabla o de una vista, el número de campos debe ser el mismo y los tipos de campo del valor fuente deben poder asignarse a los tipos de campo del valor de destino.

Asignaciones de tipos de cursor

Las asignaciones a cursores dependen del tipo de cursor. Los valores siguientes pueden asignarse a una variable o a un parámetro del tipo incorporado CURSOR:

- Un constructor de valor de cursor
- Un valor de tipo incorporado CURSOR
- Un valor de cualquier tipo de cursor definido por el usuario

A continuación se indican los valores que pueden asignarse a una variable o a un parámetro de un tipo de cursor definido por el usuario y de tipo no firme:

- Un constructor de valor de cursor
- Un valor de tipo incorporado CURSOR
- Un valor de un tipo de cursor definido por el usuario con el mismo nombre de tipo

A continuación se indican los valores que pueden asignarse a una variable o a un parámetro de un tipo de cursor definido por el usuario y de tipo firme:

- Un constructor de valor de cursor
- Un valor de un tipo de cursor definido por el usuario con el mismo nombre de tipo

Asignaciones de tipos booleanos

Los valores siguientes definidos por el sistema pueden asignarse a una variable, a un parámetro o a un tipo de retorno del tipo incorporado BOOLEAN:

- TRUE
- FALSE
- NULL

También puede asignarse el resultado de la evaluación de una condición de búsqueda. Si la evaluación de la condición de búsqueda da un resultado desconocido, se asigna el valor NULL.

Asignación de tipos de referencia

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

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

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

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

Comparaciones numéricas

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

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

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

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

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

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

Si un número es DECFLOAT(16) y el otro número es DECFLOAT(34), el valor DECFLOAT(16) se convierte a DECFLOAT(34) antes de que se efectúe la comparación.

El tipo de datos de coma flotante decimal da soporte tanto al cero positivo como al cero negativo. El cero positivo y negativo tiene representaciones binarias diferentes, pero el predicado = (equal) devolverá verdadero (true) para las comparaciones de cero negativo y positivo.

Las funciones escalares COMPARE_DECFLOAT y TOTALORDER pueden utilizarse para realizar comparaciones a nivel binario si, por ejemplo, se necesita una comparación de 2,0 <> 2,00.

El tipo de datos de coma flotante decimal da soporte a la especificación de un NaN negativo y positivo (quiet y signalling) e infinito negativo y positivo. Desde una perspectiva SQL, INFINITY = INFINITY, NAN = NAN, SNAN = SNAN y -0 = 0.

Las normas de ordenación y comparación para los valores especiales son los siguientes:

- (+/-) INFINITY se compara como igual únicamente con (+/-) INFINITY del mismo signo.
- (+/-) NAN se compara como igual únicamente con (+/-) NAN del mismo signo.
- (+/-) SNAN se compara como igual únicamente con (+/-) SNAN del mismo signo.

El orden de los distintos valores especiales es el siguiente:

- -NAN < -SNAN < -INFINITY < 0 < INFINITY < SNAN < NAN

Cuando se comparan los tipos de datos de serie y numéricos, la serie se convierte en DECFLOAT(34) utilizando las normas de una especificación CAST. Para obtener más información, consulte “Especificaciones CAST” en la publicación *Consulta de SQL, Volumen 1*. La serie debe contener una representación de serie válida de un número.

Comparaciones de series

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

Cuando se comparan series de caracteres de longitud desigual, la comparación se efectúa utilizando una copia lógica de la serie más corta, rellenada por el lado derecho con espacios en blanco para alargar dicha serie y alcanzar de esta manera la serie más larga. Esta extensión lógica se realiza para todas las series de caracteres, incluidas las que tienen el distintivo FOR BIT DATA.

Las series de caracteres (excepto las series de caracteres con el distintivo FOR BIT DATA) se comparan de acuerdo con el orden de clasificación especificado al crear la base de datos. Por ejemplo, el orden de clasificación por omisión proporcionado por el gestor de bases de datos puede dar el mismo peso a la versión en minúsculas y en mayúsculas del mismo carácter. El gestor de bases de datos realiza una comparación en dos pasos para asegurarse de que sólo se consideran iguales las series idénticas. En el primer paso, las series se comparan de acuerdo con el orden de clasificación de la base de datos. Si los pesos de los caracteres de las series son iguales, se realiza un segundo paso de "desempate" para comparar las series en base a sus valores de elemento de código real.

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

No se da soporte a las series LOB en las operaciones de comparación que utilizan operadores de comparación básicos (=, <>, <, >, <= y >=). Están soportadas en comparaciones que utilizan el predicado LIKE y la función POSSTR.

Los fragmentos de series se pueden comparar utilizando las funciones escalares SUBSTR y VARCHAR. Por ejemplo, dadas las columnas:

```
MI_SHORT_CLOB   CLOB(300)
MY_LONG_VAR     VARCHAR(8000)
```

entonces lo siguiente será válido:

```
WHERE VARCHAR(MY_SHORT_CLOB) > VARCHAR(SUBSTR(MY_LONG_VAR,1,300))
```

Ejemplos:

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

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

```
'a' < 'A' < 'á' < 'Á'
```

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

```
D4 < D3 < D2 < D1
```

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

```
 'A' < 'a' < 'Á' < 'á'
```

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

```
D1 < D2 < D3 < D4
```

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

```
'A' < 'a' < 'Á' < 'á'
```

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

```
D1 < D2 < D3 < D4
```

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

```
D1 < D2 < D3 < D4
```

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

Normas de conversión para la comparación

Cuando se comparan dos series, primero una de ellas se convierte, si es necesario, al esquema de codificación y a la página de códigos de la otra serie.

Clasificación de los resultados

Los resultados que necesitan clasificarse se ordenan en base a las normas de comparación de series que se tratan en "Comparaciones de series" en la página 135. La comparación se efectúa en el servidor de bases de datos. Al devolver los resultados a la aplicación cliente, se puede llevar a cabo una conversión de la página de códigos. Esta conversión de la pagina de códigos subsiguiente no afecta al orden del conjunto resultante determinado por el servidor.

Consideraciones de MBCS para la comparación de series

Las series de caracteres SBCS/MBCS mixtas se comparan de acuerdo con el orden de clasificación especificado cuando se ha creado la base de datos. Para las bases

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

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

Ejemplo:

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

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

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

y

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

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

Las comparaciones de series gráficas son válidas entre todos los tipos de datos de series gráficas excepto DBCLOB.

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

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

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

y

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

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

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

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

Comparaciones de fecha y hora

Un valor DATE, TIME o TIMESTAMP puede compararse con otro valor del mismo tipo de datos o con una representación de serie de dicho tipo de datos. Una DATE o una representación de serie de una fecha también pueden compararse con una TIMESTAMP, donde la información de hora que falta para el valor de la fecha se presupone que son ceros. Todas las comparaciones son cronológicas, lo que significa que cuanto más alejado en el tiempo esté del 1 de enero de 0001, mayor es el valor de dicho punto en el tiempo.

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

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

Ejemplo:

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

Comparaciones de tipos definidos por el usuario

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

Ejemplo:

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

```
CREATE TYPE YOUTH AS INTEGER WITH COMPARISONS

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

La comparación siguiente es válida:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > HIGH_SCHOOL_LEVEL
```

No será válida la siguiente comparación:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > ATTENDEE_NUMBER
```

Sin embargo, AGE se puede comparar con ATTENDEE_NUMBER utilizando una función o una especificación CAST para convertir entre el tipo diferenciado y el tipo fuente. Todas las comparaciones siguientes son válidas:

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE INTEGER(AGE) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
   WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > YOUTH(ATTENDEE_NUMBER)

SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

Los valores con un tipo estructurado definido por el usuario no se pueden comparar con ningún otro valor (se pueden utilizar los predicados NULL y TYPE).

No están soportadas las comparaciones de los valores de tipo de matriz. Los elementos de las matrices pueden compararse según las normas de comparación de tipos de datos de los elementos.

Comparaciones de tipos de filas

Una variable de fila no puede compararse con otra variable de fila, incluso si el nombre de tipo de fila es el mismo. Los campos individuales de un tipo de fila pueden compararse con otros valores, y se aplican las normas de comparación del tipo de datos del campo.

Comparaciones de tipos de cursor

Una variable de cursor no puede compararse con otra variable de cursor aunque el nombre del tipo de cursor sea el mismo.

Comparaciones de tipos booleanos

Un valor booleano se puede comparar con un valor booleano literal. Un valor TRUE es mayor que un valor FALSE.

Comparaciones de tipos de referencia

Los valores de un tipo de referencia sólo pueden compararse si sus tipos de destino tienen un supertipo común. Sólo se encontrará la función de comparación adecuada si el nombre de esquema del supertipo común es la vía de acceso a SQL. Se realiza la comparación si se utiliza el tipo de representación de los tipos de referencia. El ámbito de la referencia no se tiene en cuenta en la comparación.

Comparaciones XML en una base de datos que no es Unicode

Cuando se realiza en una base de datos no Unicode, las comparaciones entre datos XML y valores de serie de gráficos o caracteres requiere una conversión de página de códigos de uno de los dos conjuntos de datos que está comparándose. Valores de carácter o gráficos utilizados en una sentencia de SQL o XQuery, como predicado de consultas o una variable del lenguaje principal con tipo de datos de serie gráfica o de caracteres, se convierte a la página de códigos de base de datos anterior a la comparación. Si los caracteres incluidos en estos datos tienen elementos de código que no formen parte de la página de códigos de la base de

datos, se añadirán caracteres de sustitución en su lugar, ocasionando potencialmente resultados inesperados para la consulta.

Por ejemplo, un cliente con una página de códigos UTF-8 se utilizará para conectarse a un servidor de base de datos creado con la codificación en griego ISO8859-7. La expresión S_GS_M se envía como predicado de una sentencia XQuery, donde S_G representa el carácter sigma en griego en Unicode (U+03A3) y S_M representa el símbolo matemático sigma en Unicode (U+2211). Esta expresión se convierte en primer lugar a la página de códigos de la base de datos, de modo que ambos caracteres "S" se convierten al punto de códigos equivalente para sigma en la página de códigos de base de datos en griego, 0xD3. Podemos indicar este elemento de código como S_A. La expresión recién convertida S_AS_A se convierte a continuación de nuevo a UTF-8 para la comparación con los datos XML de destino. Puesto que la distinción entre estos dos puntos de código se ha perdido como consecuencia de la conversión de página de códigos que se necesita para pasar la expresión del predicado a la base de datos, los dos valores diferenciados inicialmente S_G y S_M se pasan al analizador XML como la expresión S_GS_G. A continuación, esta expresión no puede coincidir cuando se compara con el valor S_GS_M en un documento XML.

Una forma de evitar resultados de consulta inesperados que puedan ocasionar problemas de conversión de página de códigos es la de asegurar que todos los caracteres utilizados en una expresión de consulta tengan puntos de código coincidentes en la página de códigos de caracteres. Los caracteres que no tengan puntos de código coincidentes pueden incluirse por medio del uso de la referencia de entidad de caracteres de Unicode. Una referencia de entidad de caracteres siempre elude la conversión de la página de códigos. Por ejemplo, utilizando la referencia de entidad de caracteres ࢣ en vez del carácter S_M se asegura que se utilice el punto de código de Unicode correcto para la comparación, sin tener en cuenta la página de códigos de la base de datos.

Normas para tipos de datos de resultados

Los tipos de datos de un resultado los determinan las normas que se aplican a los operandos de una operación. Esta sección explica dichas normas.

Estas normas se aplican a:

- Las columnas correspondientes en selecciones completas de operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Las expresiones resultantes de una expresión CASE y la función escalar DECODE
- Argumentos de la función escalar COALESCE (también NVL y VALUE)
- Argumentos de las funciones escalares GREATEST, LEAST, MAX y MIN
- Los valores de expresiones de la lista de entrada de un predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas
- Valores de expresiones para los elementos de un constructor de matrices
- Argumentos de un predicado BETWEEN (salvo si los tipos de datos de todos los operandos son numéricos)
- Argumentos para los rangos de grupos de agregación en las especificaciones OLAP

Estas normas se aplican, sujetas a otras restricciones, sobre series largas para las distintas operaciones.

A continuación encontrará las normas que se refieren a los distintos tipos de datos. En algunos casos, se utiliza una tabla para mostrar los posibles tipos de datos resultantes. Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos y no son recomendables.

Estas tablas identifican el tipo de datos resultante, incluida la longitud aplicable o precisión y la escala. El tipo resultante se determina teniendo en cuenta los operandos. Si hay más de un par de operandos, empiece considerando el primer par. Esto da un tipo resultante que es el que se examina con el siguiente operando para determinar el siguiente tipo resultante, etcétera. El último tipo resultante intermedio y el último operando determinan el tipo resultante para la operación. El proceso de operaciones se realiza de izquierda a derecha, por lo tanto los tipos del resultado intermedios son importantes cuando se repiten operaciones. Por ejemplo, examinemos una situación que implique:

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

El primer par da como resultado un tipo CHAR(4). Los valores del resultado siempre tienen 4 bytes. El tipo resultante final es VARCHAR(4). Los valores del resultado de la primera operación UNION siempre tendrán una longitud de 4.

Series de caracteres

Un valor de serie de caracteres es compatible con otro valor de serie de caracteres. Las series de caracteres incluyen los tipos de datos CHAR, VARCHAR y CLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
CHAR(x)	CHAR(y)	CHAR(z) donde z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z) donde z = max(x,y)

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
VARCHAR(x)	CHAR(y) o VARCHAR(y)	VARCHAR(z) donde z = max(x,y)
CLOB(x)	CHAR(y), VARCHAR(y) o CLOB(y)	CLOB(z) donde z = max(x,y)

La página de códigos de la serie de caracteres del resultado se derivará en base a las normas de conversión de series.

Series gráficas

Un valor de serie gráfica es compatible con otro valor de serie gráfica. Las series gráficas incluyen los tipos de datos GRAPHIC, VARGRAPHIC y DBCLOB.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z) donde z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) o VARGRAPHIC(y)	VARGRAPHIC(z) donde z = max(x,y)
DBCLOB(x)	GRAPHIC(y), VARGRAPHIC(y) o DBCLOB(y)	DBCLOB(z) donde z = max (x,y)

La página de códigos de la serie gráfica del resultado se derivará en base a las normas de conversión de series.

Series de caracteres y gráficas en una base de datos Unicode

En una base de datos Unicode, un valor de serie de caracteres es compatible con un valor de serie gráfica.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
GRAPHIC(x)	CHAR(y) o GRAPHIC(y)	GRAPHIC(z) donde z = max(x,y)
VARGRAPHIC(x)	CHAR(y) o VARCHAR(y)	VARGRAPHIC(z) donde z = max(x,y)
VARCHAR(x)	GRAPHIC(y) o VARGRAPHIC	VARGRAPHIC(z) donde z = max(x,y)
DBCLOB(x)	CHAR(y) o VARCHAR(y) o CLOB(y)	DBCLOB(z) donde z = max (x,y)
CLOB(x)	GRAPHIC(y) o VARGRAPHIC(y)	DBCLOB(z) donde z = max (x,y)

Objeto grande binario (BLOB)

Un valor de serie binaria (BLOB) sólo es compatible con otro valor de serie binaria (BLOB). La función escalar BLOB puede utilizarse para convertir desde otros tipos si éstos deben tratarse como tipos BLOB. La longitud del resultado BLOB es la longitud mayor de todos los tipos de datos.

Numérico

Los tipos numéricos son compatibles con otros tipos de datos numéricos, tipos de datos de serie de caracteres (salvo CLOB) y en una base de datos Unicode, tipos de datos de serie gráfica (salvo DBCLOB). Los tipos numéricos incluyen SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE y DECFLOAT.

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
SMALLINT	SMALLINT	SMALLINT
SMALLINT	Serie	DECFLOAT(34)
INTEGER	SMALLINT o INTEGER	INTEGER
INTEGER	Serie	DECFLOAT(34)
BIGINT	SMALLINT, INTEGER o BIGINT	BIGINT
BIGINT	Serie	DECFLOAT(34)
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) donde p = x+max(w-x,5)[1]
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) donde p = x+max(w-x,11)[1]
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) donde p = x+max(w-x,19)[1]
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s) donde p = max(x,z)+max(w-x,y-z)[1]s = max(x,z)
DECIMAL(w,x)	Serie	DECFLOAT(34)
REAL	REAL	REAL
REAL	SMALLINT, INTEGER, BIGINT o DECIMAL	DOUBLE
REAL	Serie	DECFLOAT(34)
DOUBLE	SMALLINT, INTEGER, BIGINT, DECIMAL, REAL o DOUBLE	DOUBLE
DOUBLE	Serie	DECFLOAT(34)
DECFLOAT(n)	SMALLINT, INTEGER, DECIMAL (<=16,s), REAL o DOUBLE	DECFLOAT(n)
DECFLOAT(n)	BIGINT o DECIMAL (>16,s)	DECFLOAT(34)
DECFLOAT(n)	DECFLOAT(m)	DECFLOAT(MAX(n,m))
DECFLOAT(n)	Serie	DECFLOAT(34)

[1] La precisión no puede exceder de 31.

Fecha y hora

Los tipos de datos de fecha y hora son compatibles con otros operandos del mismo tipo de datos o con cualquier expresión CHAR o VARCHAR que contenga una representación de serie válida del mismo tipo de datos. Además, DATE es compatible con TIMESTAMP y el otro operando de un TIMESTAMP puede ser una representación de serie de una fecha y hora o una fecha. En una base de datos

Unicode, las series de caracteres y gráficas son compatibles, lo que implica que las representaciones de serie GRAPHIC o VARGRAPHIC de valores de fecha y hora son compatibles con otros operandos de fecha y hora.

Tabla 16. Tipos de datos de resultados con operandos de fecha y hora

Si un operando es...	Y el otro operando es...	El tipo de datos del resultado es...
DATE	DATE, CHAR(*y*) o VARCHAR(*y*)	DATE
TIME	TIME, CHAR(*y*) o VARCHAR(*y*)	TIME
TIMESTAMP(*x*)	TIMESTAMP(*y*)	TIMESTAMP(*max(x,y)*)
TIMESTAMP(*x*)	DATE, CHAR(*y*) o VARCHAR(*y*)	TIMESTAMP(*x*)

XML

Un valor XML es compatible con otro valor XML. El tipo de datos del resultado es XML.

Booleano

Un valor booleano es compatible con otro valor booleano. El tipo de datos del resultado es BOOLEAN.

Tipos diferenciados

Un valor de tipo diferenciado definido por el usuario sólo es compatible con otro valor del mismo tipo diferenciado definido por el usuario. El tipo de datos del resultado es el tipo diferenciado definido por el usuario.

Tipos de datos de matriz
Un valor de tipo de datos de matriz definido por el usuario sólo es compatible con otro valor del mismo tipo de datos de matriz definido por el usuario. El tipo de datos del resultado es el tipo de datos de matriz definido por el usuario.

Tipos de datos de cursor
Un valor de CURSOR es compatible con otro valor de CURSOR. El tipo de datos del resultado es CURSOR. Un valor de tipo de datos de cursor definido por el usuario sólo es compatible con otro valor del mismo tipo de datos de cursor definido por el usuario. El tipo de datos del resultado es el tipo de datos de cursor definido por el usuario.

Tipos de datos de fila
Un valor de tipo de datos de fila definido por el usuario sólo es compatible con otro valor del mismo tipo de datos de fila definido por el usuario. El tipo de datos del resultado es el tipo de datos de fila definido por el usuario.

Tipos de referencia

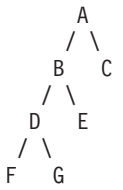
Un valor de tipo de referencia es compatible con otro valor del mismo tipo de referencia siempre y cuando sus tipos de destino tengan un supertipo común. El tipo de datos del resultado es un tipo de referencia que tiene un supertipo común como tipo de destino. Si todos los operandos tienen la tabla de ámbito idéntica, el

resultado tiene esta tabla de ámbito. De lo contrario, el resultado no tiene ámbito.

Tipos estructurados

Un valor de tipo estructurado es compatible con otro valor del mismo tipo estructurado siempre que tengan un supertipo común. El tipo de datos estático de la columna del tipo estructurado resultante es el tipo estructurado que es el supertipo menos común de cualquiera de las dos columnas.

Por ejemplo, considere la siguiente jerarquía de tipos estructurados:



Los tipos estructurados del tipo estático E y F son compatibles con el tipo estático resultante de B, que es el supertipo menos común de E y F.

Atributo con posibilidad de nulos del resultado

A excepción de INTERSECT y EXCEPT, el resultado permite nulos a menos que ambos operandos no permitan nulos.

- Para INTERSECT, si cualquier operando no permite nulos, el resultado no permite nulos (la intersección nunca sería nula).
- Para EXCEPT, si el primer operando no permite nulos, el resultado no permite nulos (el resultado sólo puede ser valores del primer operando).

Normas para la conversión de series

La página de códigos utilizada para realizar una operación se determina por las normas que se aplican a los operandos en dicha operación. Esta sección explica dichas normas.

Estas normas se aplican a:

- Las columnas de serie correspondientes en selecciones completas con operaciones de conjuntos (UNION, INTERSECT y EXCEPT)
- Los operandos de concatenación
- Los operandos de predicados (a excepción de LIKE)
- Las expresiones resultantes de una expresión CASE y la función escalar DECODE
- Los argumentos de la función escalar COALESCE (también NVL y VALUE)
- Argumentos de las funciones escalares GREATEST, LEAST, MAX y MIN
- Los argumentos *serie-fuente* y *serie-inserción* de la función escalar OVERLAY (e INSERT)
- Los valores de expresiones de la lista de entrada de un predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas.

En cada caso, la página de códigos del resultado se determina en el momento del enlace, y la ejecución de la operación puede implicar la conversión de series a la página de códigos identificada por dicha página de códigos. Un carácter que no tenga una conversión válida se correlaciona con el carácter de sustitución del juego de caracteres y SQLWARN10 se establece en 'W' en la SQLCA.

La página de códigos del resultado se determina por las páginas de códigos de los operandos. Las páginas de códigos de los dos primeros operandos determinan una página de códigos del resultado intermedia, esta página de códigos y la del siguiente operando determinan una nueva página de códigos del resultado intermedia (si se aplica), etcétera. La última página de códigos del resultado intermedia y la página de códigos del último operando determinan la página de códigos de la serie o columna del resultado. En cada par de páginas de códigos, el resultado se determina por la aplicación secuencial de las normas siguientes:

- Si las páginas de códigos son iguales, el resultado es dicha página de códigos.
- Si cualquiera de las dos páginas de códigos es BIT DATA (página de códigos 0), la página de códigos del resultado es BIT DATA.
- En una base de datos Unicode, si una página de códigos denota datos en un esquema de codificación que es distinto de la otra página de códigos, el resultado es UCS-2 sobre UTF-8 (es decir, los datos de tipo gráfico sobre los datos de tipo carácter). (En una base de datos que no sea Unicode, no se permite la conversión entre distintos esquemas de codificación.
- Para los operandos que son variables del lenguaje principal (cuya página de códigos no es BIT DATA), la página de códigos resultante es la página de códigos de la base de datos. Los datos de entrada de este tipo de variables del lenguaje principal se convierten de la página de códigos de la aplicación a la página de códigos de la base de datos antes de utilizarse.

Las conversiones a la página de códigos del resultado se realizan, si es necesario, para:

- Un operando del operador de concatenación

- El argumento seleccionado de la función escalar COALESCE (también NVL y VALUE)
- El argumento seleccionado de las funciones escalares GREATEST, LEAST, MAX y MIN
- Los argumentos *serie-fuente* y *serie-inserción* de la función escalar OVERLAY (e INSERT)
- La expresión de resultado seleccionada de la expresión CASE y la función escalar DECODE
- Las expresiones de la lista in del predicado IN
- Las expresiones correspondientes de una cláusula VALUES de múltiples filas
- Las columnas correspondientes que hacen referencia en operaciones de conjuntos.

La conversión de los caracteres es necesaria si son ciertas todas las afirmaciones siguientes:

- Las páginas de códigos son diferentes
- Ninguna serie es BIT DATA
- La serie no es nula ni está vacía

Ejemplos

Ejemplo 1: Supongamos lo siguiente en una base de datos creada con la página de códigos 850:

Expresión	Tipo	Página de códigos
COL_1	columna	850
HV_2	variable del lenguaje principal	437

Cuando se evalúa el predicado:

```
COL_1 CONCAT :HV_2
```

la página de códigos del resultado de los dos operandos es 850, porque los datos de variables del lenguaje principal se convertirán a página de códigos de la base de datos antes de utilizarse.

Ejemplo 2: Utilizando la información del ejemplo anterior para evaluar el predicado:

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

la página de códigos del resultado es 850. Por lo tanto, la página de códigos del resultado para la función escalar COALESCE será la página de códigos 850.

Comparaciones de series en una base de datos Unicode

La coincidencia de patrón es un área en la que el comportamiento de las bases de datos MBCS existentes es ligeramente diferente del comportamiento de una base de datos Unicode.

Para bases de datos MBCS de DB2 Database para Linux, UNIX y Windows, el comportamiento actual es el siguiente: Si la expresión de coincidencia contiene datos MBCS, el patrón puede incluir caracteres SBCS y no SBCS. Los caracteres especiales del patrón se interpretan del modo siguiente:

- Un signo de subrayado de media anchura SBCS hace referencia a un carácter SBCS.
- Un carácter de subrayado de anchura completa no SBCS hace referencia a un carácter no SBCS.
- Un signo de porcentaje (SBCS de media anchura o no SBCS de anchura completa) hace referencia a cero o más caracteres SBCS o no SBCS.

En una base de datos Unicode, no se suele hacer distinción entre caracteres de ″un solo byte″ y ″no de un solo byte″. Aunque el formato UTF-8 es una codificación de ″bytes mixtos″ de caracteres Unicode, no existe ninguna distinción real entre caracteres SBCS y no SBCS en UTF-8. Cada carácter es un carácter Unicode, independientemente del número de bytes en formato UTF-8. En una serie gráfica Unicode, cada carácter no suplementario, incluido el carácter de subrayado de media anchura (U+005F) y el signo de porcentaje de media anchura (U+0025), tiene dos bytes de anchura. Para bases de datos Unicode, los caracteres especiales del patrón se interpretan del modo siguiente:

- Para series de caracteres, un carácter de subrayado de media anchura (X'5F') o un carácter de subrayado de anchura completa (X'EFBCBF') hace referencia a un carácter Unicode. Un signo de porcentaje de media anchura (X'25') o un signo de porcentaje de anchura completa (X'EFBC85') hace referencia a cero o más caracteres Unicode.
- Para series gráficas, un carácter de subrayado de media anchura (U+005F) o un carácter de subrayado de anchura completa (U+FF3F) hace referencia a un carácter Unicode. Un signo de porcentaje de media anchura (U+0025) o un signo de porcentaje de anchura completa (U+FF05) hace referencia a cero o más caracteres Unicode.

Nota: Se necesitan dos caracteres de subrayado para la coincidencia de un carácter gráfico Unicode suplementario, pues, en una serie gráfica, un carácter de este tipo se representa mediante dos caracteres UCS-2. En una serie de caracteres, sólo se necesita un único carácter de subrayado para la coincidencia de un carácter Unicode suplementario.

Para la ″expresión de escape″ opcional, que especifica un carácter que se debe utilizar para modificar el significado especial de los caracteres de subrayado y signo de porcentaje, la expresión se puede especificar mediante cualquiera de los siguientes:

- Una constante
- Un registro especial
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene cualquiera de las anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento en la expresión puede ser del tipo CLOB o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.
- Para las series de caracteres, el resultado de la expresión debe ser un carácter o una serie FOR BIT DATA que contenga exactamente un (1) byte (SQLSTATE 22019). Para las series gráficas, el resultado de la expresión debe ser un carácter (SQLSTATE 22019).

Resolución del objeto de anclaje para un tipo anclado

El objeto de anclaje de un tipo anclado que no especifica ROW se especifica con un nombre que podría representar una variable de SQL, un parámetro de SQL, una variable global, una variable de módulo, una columna de una tabla o una columna de una vista.

La forma en la que se resolverá el objeto de anclaje dependerá del número de identificadores que contenga su nombre y del contexto de la sentencia que utiliza la cláusula ANCHOR.

- Si el nombre del objeto de anclaje está especificado con un identificador, el nombre podría representar una variable de SQL, un parámetro de SQL, una variable de módulo o una variable global.
- Si el nombre del objeto de anclaje está especificado con dos identificadores, el nombre podría representar una variable de SQL calificada por etiqueta, un parámetro de SQL calificado por rutina, una variable global calificada por esquema, una variable de módulo, la columna de una tabla o la columna de una vista.
- Si el nombre del objeto de anclaje está especificado con tres identificadores, el nombre representa una columna de una tabla calificada por esquema, una columna de una vista calificada por esquema, una variable global calificada por el nombre del servidor actual y un esquema, o una variable de módulo calificada por esquema.

Una variable de SQL sólo puede ser candidata para nombre de objeto de anclaje si se utiliza la cláusula ANCHOR en una declaración de variable de SQL dentro de una sentencia compuesta. Un parámetro de SQL sólo puede ser candidato para nombre de objeto de anclaje si se utiliza la cláusula ANCHOR en una declaración de variable de SQL dentro de una sentencia compuesta empleada en un cuerpo de rutina de SQL.

La resolución del nombre de objeto de anclaje que tiene un identificador se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta, busque un nombre de variable de SQL coincidente comenzando por la sentencia compuesta anidada más interna hasta llegar a la sentencia compuesta más externa.
2. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta que forma parte de un cuerpo de rutina, busque un nombre de parámetro de SQL coincidente para la rutina.
3. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, busque un nombre de variable de módulo coincidente dentro del módulo.
4. Si la búsqueda todavía no ha resultado satisfactoria, busque una tabla o una vista utilizando el primer identificador como nombre de esquema y el segundo identificador como nombre de vista o tabla.
5. Si la búsqueda todavía no ha resultado satisfactoria, busque una variable global con un nombre de variable global coincidente en la vía de acceso de SQL.

La resolución del nombre de objeto de anclaje que tiene dos identificadores se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta, busque un nombre de variable de SQL calificado coincidente comenzando por la sentencia compuesta anidada más interna hasta llegar a la sentencia compuesta más externa.

2. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta que forma parte de un cuerpo de rutina, busque un nombre de parámetro de SQL coincidente para la rutina si el primer identificador del nombre del objeto de anclaje coincide con el nombre de la rutina.
3. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, y si el primer identificador coincide con el nombre de módulo de dicho módulo, busque un nombre de variable de módulo dentro del módulo que coincida con el segundo identificador.
4. Si la búsqueda todavía no ha resultado satisfactoria, busque una columna de vista o tabla en el esquema actual utilizando el primer identificador como nombre de tabla o vista y el segundo identificador como nombre de columna.
5. Si la búsqueda todavía no ha resultado satisfactoria, busque una variable global utilizando el primer identificador como nombre de esquema y el segundo identificador como nombre de variable global.
6. Si la búsqueda no ha resultado satisfactoria y no ha buscado un módulo en el paso 3, busque un módulo en la vía de acceso de SQL con un nombre que coincida con el primer identificador. Si lo encuentra, utilice el segundo identificador para buscar un nombre de variable de módulo publicado coincidente en el módulo.
7. Si no encuentra un módulo utilizando la vía de acceso de SQL en el paso 6, consulte un alias público de módulo que coincida con el nombre del primer identificador. Si lo encuentra, utilice el segundo identificador para buscar un nombre de variable de módulo publicado coincidente en el módulo identificado por el alias público de módulo.

La resolución del nombre de objeto de anclaje que tiene tres identificadores se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, y si los dos primeros identificadores coinciden con el nombre de esquema y el nombre de módulo de dicho módulo, busque una variable de módulo cuyo nombre coincida con el último identificador.
2. Si no la ha encontrado en el paso anterior o éste no es aplicable, busque una columna de vista o tabla utilizando el primer identificador como nombre de esquema, el segundo identificador como nombre de vista o tabla y el tercer identificador como nombre de columna.
3. Si no la ha encontrado en el paso anterior y el primer identificador es igual al nombre del servidor actual, busque una variable global utilizando el segundo identificador como nombre de esquema y el tercer identificador como nombre de variable global.
4. Si no la ha encontrado y no se buscó un módulo en el paso 1, busque una variable de módulo publicada utilizando el primer identificador como nombre de esquema, el segundo identificador como nombre de módulo y, si existe dicho módulo, utilice el tercer identificador para buscar un nombre de variable de módulo publicado coincidente en el módulo.

Resolución del objeto de anclaje para un tipo de fila anclado

El objeto de anclaje de un tipo anclado que incluye la palabra clave ROW se especifica con un nombre que podría representar una variable de SQL, un parámetro de SQL, una variable global, una variable de módulo, una tabla o una vista, dependiendo del contexto y del número de identificadores contenidos en el nombre y del contexto de la cláusula ANCHOR.

La forma en la que se resolverá el objeto de anclaje dependerá del número de identificadores que contenga su nombre y del contexto de la sentencia que utiliza la cláusula ANCHOR.

- Si el nombre del objeto de anclaje está especificado con un identificador, el nombre podría representar una variable de SQL, un parámetro de SQL, una variable de módulo, una variable global, una tabla o una vista.
- Si el nombre del objeto de anclaje está especificado con dos identificadores, el nombre podría representar una variable de SQL calificada por etiqueta, un parámetro de SQL calificado por rutina, una variable global calificada por esquema, una variable de módulo, una tabla calificada por esquema o una vista calificada por esquema.
- Si el nombre del objeto de anclaje está especificado con tres identificadores, el nombre podría representar una variable global calificada por el nombre del servidor actual y un esquema, una tabla calificada por el nombre del servidor actual y un esquema, una vista calificada por el nombre del servidor actual y un esquema o una variable de módulo calificada por esquema.

Una variable de SQL sólo puede ser candidata para nombre de objeto de anclaje si se utiliza la cláusula ANCHOR en una declaración de variable de SQL dentro de una sentencia compuesta. Un parámetro de SQL sólo puede ser candidato para nombre de objeto de anclaje si se utiliza la cláusula ANCHOR en una declaración de variable de SQL dentro de una sentencia compuesta empleada en un cuerpo de rutina de SQL. La resolución del nombre de objeto de anclaje que tiene un identificador se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta, busque un nombre de variable de SQL coincidente comenzando por la sentencia compuesta anidada más interna hasta llegar a la sentencia compuesta más externa.
2. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta que forma parte de un cuerpo de rutina, busque un nombre de parámetro de SQL coincidente para la rutina.
3. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, busque un nombre de variable de módulo coincidente dentro del módulo.
4. Si la búsqueda todavía no ha resultado satisfactoria, busque una tabla o una vista con un nombre coincidente en el esquema actual.
5. Si la búsqueda todavía no ha resultado satisfactoria, busque una variable global de esquema con un nombre de variable global coincidente en la vía de acceso de SQL.

La resolución del nombre de objeto de anclaje que tiene dos identificadores se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta, busque un nombre de variable de SQL calificado coincidente comenzando por la sentencia compuesta anidada más interna hasta llegar a la sentencia compuesta más externa.

2. Si la cláusula ANCHOR es una declaración de variable de SQL de una sentencia compuesta que forma parte de un cuerpo de rutina, busque un nombre de parámetro de SQL coincidente para la rutina si el primer identificador del nombre del objeto de anclaje coincide con el nombre de la rutina.
3. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, y si el primer identificador coincide con el nombre de módulo de dicho módulo, busque un nombre de variable de módulo dentro del módulo que coincida con el segundo identificador.
4. Si la búsqueda todavía no ha resultado satisfactoria, busque una tabla o una vista utilizando el primer identificador como nombre de esquema y el segundo identificador como nombre de vista o tabla.
5. Si la búsqueda todavía no ha resultado satisfactoria, busque una variable global utilizando el primer identificador como nombre de esquema y el segundo identificador como nombre de variable global.
6. Si la búsqueda no ha resultado satisfactoria y no ha buscado un módulo en el paso 3, busque un módulo en la vía de acceso de SQL con un nombre que coincida con el primer identificador. Si lo encuentra, utilice el segundo identificador para buscar un nombre de variable de módulo publicado coincidente en el módulo.
7. Si no encuentra un módulo utilizando la vía de acceso de SQL en el paso 6, consulte un alias público de módulo que coincida con el nombre del primer identificador. Si lo encuentra, utilice el segundo identificador para buscar un nombre de variable de módulo publicado coincidente en el módulo identificado por el alias público de módulo.

La resolución del nombre de objeto de anclaje que tiene tres identificadores se realiza mediante los pasos siguientes:

1. Si la cláusula ANCHOR se utiliza en la definición de un objeto de módulo, y si los dos primeros identificadores coinciden con el nombre de esquema y el nombre de módulo de dicho módulo, busque una variable de módulo cuyo nombre coincida con el último identificador.
2. Si no la ha encontrado y el primer identificador es igual al nombre del servidor actual, busque una tabla o una vista utilizando el segundo identificador como nombre de esquema y el tercer identificador como nombre de tabla o vista.
3. Si no la ha encontrado y el primer identificador es igual al nombre del servidor actual, busque una variable global utilizando el segundo identificador como nombre de esquema y el tercer identificador como nombre de variable global.
4. Si no la ha encontrado y no se buscó un módulo en el paso 1, busque una variable de módulo publicada utilizando el primer identificador como nombre de esquema, el segundo identificador como nombre de módulo y, si existe dicho módulo, utilice el tercer identificador para buscar un nombre de variable publicada coincidente en el módulo.

Tipos de datos compatibles entre particiones de base de datos

La *compatibilidad entre particiones de base de datos* se define entre los tipos de base de datos de las columnas correspondientes de las claves de distribución. Los tipos de datos compatibles entre particiones de base de datos tienen la propiedad de dos variables, una de cada tipo, con el mismo valor, se correlacionan con el mismo índice de correlación de distribución por la misma función de partición.

La Tabla 17 en la página 154 muestra la compatibilidad de los tipos de datos en las particiones de base de datos.

La compatibilidad entre particiones de base de datos tiene las características siguientes:

- Se utilizan formatos internos para DATE, TIME y TIMESTAMP. No son compatibles entre sí y ninguno es compatible con los tipos de datos gráficos o de caracteres.
- La compatibilidad de la partición no resulta afectada por la capacidad de nulos de una columna.
- La compatibilidad de la partición resulta afectada por la clasificación. Las clasificaciones basadas en la UCA sensibles a la configuración local requieren una correspondencia exacta en la clasificación, excepto que se ignora el atributo de fuerza (S) de la clasificación. Todas las demás clasificaciones se consideran equivalentes a efectos de determinar la compatibilidad de la partición.
- Las columnas de caracteres definidas con FOR BIT DATA sólo resultan compatibles con las columnas de caracteres sin FOR BIT DATA cuando se utiliza una clasificación que no sea la clasificación basada en la UCA sensible a la configuración local.
- Los valores NULL de los tipos de datos compatibles se tratan de manera idéntica. Se pueden generar resultados diferentes para los valores NULL de tipos de datos no compatibles.
- Se utiliza el tipo de datos del UDT para analizar la compatibilidad entre particiones de base de datos.
- Las indicaciones de fecha y hora del mismo valor de la clave de distribución se tratan de manera idéntica, incluso si difieren en su precisión.
- Los decimales del mismo valor de la clave de distribución se tratan de manera idéntica, incluso si difieren su escala y precisión.
- La función de generación aleatoria proporcionada por el sistema ignora los blancos de cola de las series de caracteres (CHAR, VARCHAR, GRAPHIC o VARGRAPHIC).
- Cuando se utiliza una clasificación basada en la UCA sensible a la configuración local, CHAR, VARCHAR, GRAPHIC y VARGRAPHIC son tipos de datos compatibles. Cuando se utilizan otras clasificaciones, CHAR y VARCHAR son tipos compatibles y GRAPHIC y VARGRAPHIC son tipos compatibles, pero CHAR y VARCHAR no son tipos compatibles con GRAPHIC y VARGRAPHIC. CHAR o VARCHAR de diferentes longitudes son tipos de datos compatibles.
- Los valores DECFLOAT que sean iguales se tratan de manera idéntica incluso si su precisión es diferente. Los valores DECFLOAT que sean numéricamente iguales se tratan de manera idéntica incluso si tienen un número diferente de dígitos significativos.
- Los tipos de datos que no están soportados como parte de una clave de distribución no son aplicables para la compatibilidad de partición de base de

datos. Esto incluye columnas cuyos tipos de datos son BLOB, CLOB, DBCLOB, XML, tipo diferenciado en función de cualquiera de estos tipos de datos o, tipo estructurado.

Tabla 17. Compatibilidades de partición de base de datos

Operandos	Entero binario	Número decimal	Coma flotante	Coma flotante decimal	Serie de caracteres	Serie gráfica	Fecha	Hora	Indicación de fecha y hora	Tipo diferenciado
Entero binario	Sí	No	No	No	No	No	No	No	No	[1]
Número decimal	No	Sí	No	No	No	No	No	No	No	[1]
Coma flotante	No	No	Sí	No	No	No	No	No	No	[1]
Coma flotante decimal	No	No	No	Sí	No	No	No	No	No	[1]
Serie de caracteres	No	No	No	No	Sí[2]	2, 3	No	No	No	[1]
Serie gráfica	No	No	No	No	2, 3	Sí[2]	No	No	No	[1]
Fecha	No	No	No	No	No	No	Sí	No	No	[1]
Hora	No	No	No	No	No	No	No	Sí	No	[1]
Indicación de fecha y hora	No	No	No	No	No	No	No	No	Sí	[1]
Tipo diferenciado	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

Nota:

[1] Un valor de tipo diferenciado es compatible desde el punto de vista de partición de base de datos con el tipo de datos fuente del tipo diferenciado o con cualquier otro tipo diferenciado con el mismo tipo de datos fuente. El tipo de datos fuente del tipo diferenciado debe ser un tipo de datos soportado como parte de una clave de distribución. El valor de un tipo diferenciado definido por el usuario (UDT) es compatible desde el punto de vista de partición de base de datos con el tipo fuente del UDT o cualquier otro UDT con un tipo fuente compatible en el nivel de partición de base de datos. Un tipo diferenciado no puede basarse en BLOB, CLOB, DBCLOB o XML.

[2] Los tipos de serie gráfica y de caracteres son compatibles cuando tienen clasificaciones compatibles.

[3] Los tipos de serie gráfica y de caracteres son compatibles cuando está vigente una clasificación basada en la UCA sensible a la configuración local. De lo contrario, no son tipos compatibles.

Constantes

Una *constante* (a veces llamada un *literal*) especifica un valor. Las constantes se clasifican en constantes de tipo serie y constantes numéricas. Las constantes numéricas pueden, a su vez, ser constantes enteras, de coma flotante y decimales.

Todas las constantes tienen el atributo NOT NULL.

Un valor de cero negativo en una constante numérica (-0) indica el mismo valor que un cero sin el signo (0).

Los tipos definidos por el usuario son tipos firmes. Esto significa que un tipo definido por el usuario sólo es compatible con su propio tipo. Sin embargo, una constante tiene un tipo interno. Por lo tanto, una operación que implique un tipo definido por el usuario y una constante sólo es posible si el tipo definido por el usuario se ha convertido al tipo interno de la constante o si la constante se ha convertido al tipo definido por el usuario. Por ejemplo, si se utiliza la tabla y el tipo diferenciado del apartado “Comparaciones de tipos definidos por el usuario” en la página 138, serán válidas las siguientes comparaciones con la constante 14:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > 14
```

No será válida la siguiente comparación:

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > 14
```

Constantes enteras

Una *constante entera* especifica un entero en forma de número, con signo o sin signo, con un máximo de 19 dígitos que no incluye ninguna coma decimal. El tipo de datos de una constante entera es entero grande si su valor está comprendido en el rango de un entero grande. El tipo de datos de una constante entera es un entero superior si su valor se encuentra fuera del rango de un entero grande, pero está comprendido en el rango de un entero superior. Una constante definida fuera del rango de valores enteros superior se considera una constante decimal.

Observe que la representación literal más pequeña de una constante entera grande es -2 147 483 647 y no -2 147 483 648, que es el límite para los valores enteros. De manera similar, la representación literal más pequeña de una constante entera superior es -9 223 372 036 854 775 807 y no -9 223 372 036 854 775 808, que es el límite para los valores enteros superiores.

Ejemplos:

```
64     -15     +100     32767     720176      12345678901
```

En los diagramas de sintaxis, el término ’entero’ se utiliza para una constante entera grande que no debe incluir un signo.

Constantes de coma flotante

Una *constante de coma flotante* especifica un número de coma flotante en forma de dos números separados por una E. El primer número puede incluir un signo y una coma decimal; el segundo número puede incluir un signo, pero no una coma

decimal. El tipo de datos de una constante de coma flotante es de precisión doble. El valor de la constante es el producto del primer número y la potencia de 10 especificada por el segundo número; este valor debe estar dentro del rango de los números de coma flotante. El número de bytes de la constante no debe exceder de 30.

Ejemplos:

```
15E1    2,E5    2,2E-1    +5,E+2
```

Constantes decimales

Una *constante decimal* es un número con o sin signo de 31 dígitos de longitud como máximo y que incluye una coma decimal o no está comprendido dentro del rango de enteros binarios. Debe estar comprendido en el rango de números decimales. La precisión es el número total de dígitos (incluyendo los ceros iniciales y de cola); la escala es el número de dígitos situados a la derecha de la coma decimal (incluyendo los ceros de cola).

Ejemplos:

```
25.5    1000.    -15.    +37589.3333333333
```

Constantes de coma flotante decimal

No hay ninguna constante de coma flotante decimal excepto los valores especiales de coma flotante decimal, que se interpreta como DECFLOAT(34).

Estos valores especiales son: INFINITY, NAN y SNAN. INFINITY representa infinito, un número cuya magnitud es infinitamente grande. INFINITY puede ir precedido por un signo opcional. Puede especificarse INF en lugar de INFINITY. NAN significa No es un número (NaN) y a veces se denomina Quiet NaN. Es un valor que representa resultados no definidos que no produce ningún aviso ni ninguna excepción.. SNAN representa Signalling NaN (sNaN). Es un valor que representa resultados no definidos que producirá un aviso o una excepción si se utiliza en cualquier operación que esté definida en cualquier operación numérica. Tanto NAN como SNAN pueden ir precedidos por un signo opcional, pero el signo no es significativo para las operaciones aritméticas. Se puede utilizar SNAN en operaciones no numéricas sin producir un aviso o una excepción, por ejemplo en la lista VALUES de INSERT o como constante comparada en un predicado.

```
SNAN   -INFINITY
```

Cuando se utiliza uno de los valores especiales (INFINITY, NAN y SNAN) en un contexto en el que se podría interpretar como un nombre, convierta explícitamente el valor a coma flotante decimal. Por ejemplo:

```
CAST ('SNAN' AS DECFLOAT)
```

Todos los valores no especiales se interpretan como constantes decimales, de coma flotante o entero, con arreglo a las normas especificadas anteriormente. Para obtener un valor de coma flotante decimal numérico, utilice la función de conversión DECFLOAT con una constante de serie de caracteres. No es recomendable utilizar constantes de coma flotante como argumentos para la función DECFLOAT, ya que la coma flotante no es exacta y el valor de coma flotante decimal puede resultar diferente de los caracteres de dígito decimales que componen el argumento. En su lugar, utilice constantes de caracteres como argumentos para la función DECFLOAT.

Por ejemplo, DECFLOAT('6.0221415E23', 34) devuelve el valor de coma flotante decimal 6.0221415E+23, pero DECFLOAT(6.0221415E23, 34) devuelve el valor de coma flotante decimal 6.0221415000000003E+23.

Constantes de series de caracteres

Una *constante de serie de caracteres* especifica una serie de caracteres de longitud variable. Existen tres formatos de constante de serie de caracteres:

- Una secuencia de caracteres que empieza y termina con un delimitador de serie, que es un apóstrofo ('). El número de bytes entre los delimitadores de serie no puede ser superior a 32.672. Se utilizan dos delimitadores de serie consecutivos para representar un delimitador de serie en la serie de caracteres. Dos delimitadores de serie consecutivos que no están contenidos en una serie representan la serie vacía.
- X seguida de una secuencia de caracteres que empieza y termina con un delimitador de serie. Este formato de constante de serie de caracteres también se denomina *constante hexadecimal*. Los caracteres entre los delimitadores de serie deben ser un número par de dígitos hexadecimales. Los espacios en blanco entre los delimitadores de serie se ignoran. El número de dígitos hexadecimales no debe exceder de 32.672. Un dígito hexadecimal es un dígito o cualquiera de las letras A a F (mayúsculas o minúsculas). Bajo los convenios de notación hexadecimal, cada par de dígitos hexadecimales representa un carácter. Este formato de constante de serie de caracteres le permite especificar caracteres que no tienen representación de teclado.
- U& seguida de una secuencia de caracteres que empieza y termina con un delimitador de serie y que está seguida opcionalmente de la cláusula UESCAPE. Este formato de constante de serie de caracteres también se denomina *constante de serie Unicode*. El número de bytes entre los delimitadores de serie no puede ser superior a 32.672. La constante de serie Unicode se convierte de UTF-8 a la página de códigos de sección durante la compilación de sentencia. Se utilizan dos delimitadores de series consecutivos para representar un delimitador de series dentro de la serie de caracteres. Se utilizan dos caracteres de escape de Unicode consecutivos para representar un carácter de escape de Unicode en la serie de caracteres, pero estos caracteres cuentan como un carácter cuando se calculan las longitudes de las constantes de tipo carácter. Dos delimitadores de serie consecutivos que no están contenidos en una serie representan la serie vacía. Dado que un carácter en UTF-8 puede estar en el rango de 1 a 4 bytes, una constante de tipo serie de Unicode de la longitud máxima puede representar en realidad menos de 32.672 caracteres.

  Un carácter se puede expresar mediante su carácter tipográfico (*glifo*) o su elemento de código Unicode. El elemento de código de un carácter Unicode está en el rango de X'000000' a X'10FFFF'. Para expresar un carácter Unicode mediante el elemento de código, utilice el carácter de escape de Unicode seguido de 4 dígitos hexadecimales o el carácter de escape de Unicode seguido de un signo más (+) y 6 dígitos hexadecimales. El carácter de escape de Unicode por omisión es la barra inclinada invertida (\), pero se puede especificar un carácter diferente con la cláusula UESCAPE. La cláusula UESCAPE se especifica como la palabra clave UESCAPE seguida de un carácter individual entre delimitadores de serie. El carácter de escape de Unicode no puede ser un signo más (+), unas comillas dobles ("), unas comillas simples ('), un espacio en blanco o ninguno de los caracteres de 0 a 9 o de A a F, en mayúsculas o minúsculas (SQLSTATE 42604). Un ejemplo de los dos modos en que se puede especificar la letra latina A mayúscula como elemento de código de Unicode es \0041 y \+000041.

El valor de una constante se convierte siempre en la página de códigos de la base de datos cuando se vincula con la base de datos. Se considera que está en la página de códigos de la base de datos. Por lo tanto, si se utiliza en una expresión que combina una constante con una columna FOR BIT DATA y cuyo resultado es FOR BIT DATA, el valor de la constante no se convertirá desde su representación de página de códigos de base de datos.

Ejemplos:

```
'12/14/1985'    '32'    'DON''T CHANGE'    ''
X'FFFF'    X'46 72 61 6E 6B'
U&'\0141\00F6d\017A es una ciudad de Polonia'   U&'c:\\temp'   U&'@+01D11E' UESCAPE '@'
```

La serie situada más a la derecha en la segunda línea del ejemplo representa el patrón VARCHAR de la serie ASCII 'Frank'. La última línea corresponde a: 'Lódz es una ciudad de Polonia', 'c:\temp' y un carácter individual que representa el símbolo musical de clave de sol.

Constantes de series gráficas

Una *constante de serie gráfica* especifica una serie gráfica de longitud variable formada por una secuencia de caracteres de doble byte que empieza y termina por un apóstrofe de un byte ('), y que está precedida por un carácter G o N de un solo byte. Los caracteres que se encuentran entre los apóstrofes deben representar un número par de bytes, y la longitud de la serie gráfica no debe exceder los 16.336 bytes.

Ejemplos:

```
G'serie de caracteres de doble byte'
N'serie de caracteres de doble byte'
```

El apóstrofo no debe aparecer como parte de un carácter MBCS para que se considere como delimitador.

En una base de datos Unicode, constante de serie gráfica hexadecimal que especifica que también se da soporte a una serie gráfica de longitud variable. El formato de una constante de serie gráfica hexadecimal es: GX seguido por una secuencia de caracteres que empieza y termina por un apóstrofe. Los caracteres que se encuentran entre los apóstrofes deben ser un múltiplo par de cuatro dígitos hexadecimales. El número de dígitos hexadecimales no debe ser superior a 16.336; de lo contrario, se devuelve un error (SQLSTATE 54002). Si el formato de la constante de serie gráfica hexadecimal no es correcto, se devuelve un error (SQLSTATE 42606). Cada grupo de cuatro dígitos representa un único carácter gráfico. En una base de datos Unicode, esto sería un único carácter gráfico UCS-2.

Ejemplos:

```
GX'FFFF'
```

representa el patrón de bits '1111111111111111' en una base de datos Unicode.

```
GX'0052006900063006B'
```

representa el patrón VARGRAPHIC de la serie ASCII 'Rick' en una base de datos a Unicode.

Constantes de fecha y hora

Una *constante de fecha y hora* especifica una fecha, una hora o una indicación de fecha y hora.

Por lo general, las constantes de serie de caracteres se utilizan para representar valores de fecha y hora constantes en asignaciones y comparaciones. Sin embargo, se puede utilizar el nombre de tipo de datos asociado antes de los formatos específicos de la constante de serie de caracteres para denotar específicamente la constante como constante de fecha y hora en vez de una constante de serie de caracteres. El formato de estas tres constantes de fecha y hora es el siguiente:

DATE '*aaaa-mm-dd***'**
El tipo de datos del valor es DATE.

TIME '*hh:mm:ss***'**
o
TIME '*hh:mm***'**
El tipo de datos del valor es TIME.

TIMESTAMP '*aaaa-mm-dd hh:mm:ss.nnnnnnnnnnnn***'**
o bien
TIMESTAMP '*aaaa-mm-dd-hh.mm.ss.nnnnnnnnnnnn***'**
donde el número de dígitos de segundos fraccionarios puede variar entre 0 y 12 y el carácter de punto se puede omitir si no hay segundos fraccionarios. El tipo de datos del valor es TIMESTAMP(*p*), donde *p* es el número de dígitos de los segundos fraccionarios.

Pueden omitirse los ceros iniciales de los elementos correspondientes al mes, día y hora de la parte de la constante de serie de caracteres, cuando sea aplicable, en todas las constantes de fecha y hora. Deben incluirse ceros iniciales para los elementos de minutos y segundos de las constantes TIME o TIMESTAMP. Se pueden incluir y omitir los espacios en blanco finales.

Constantes de series gráficas UCS-2

En una base de datos Unicode, serie gráfica UCS-2 hexadecimal que especifica que se da soporte a una constante de serie gráfica UCS-2 de longitud variable. El formato de una constante de serie gráfica UCS-2 hexadecimal es: UX seguido por una secuencia de caracteres que empieza y termina por un apóstrofe. Los caracteres que se encuentran entre los apóstrofes deben ser un múltiplo par de cuatro dígitos hexadecimales. El número de dígitos hexadecimales no debe ser superior a 16.336; de lo contrario, se devuelve un error (SQLSTATE 54002). Si el formato de la constante de serie gráfica UCS-2 hexadecimal no es correcto, se devuelve un error (SQLSTATE 42606). Cada grupo de cuatro dígitos representa un único carácter gráfico UCS-2.

Ejemplo:

```
UX'0042006F006200620079'
```

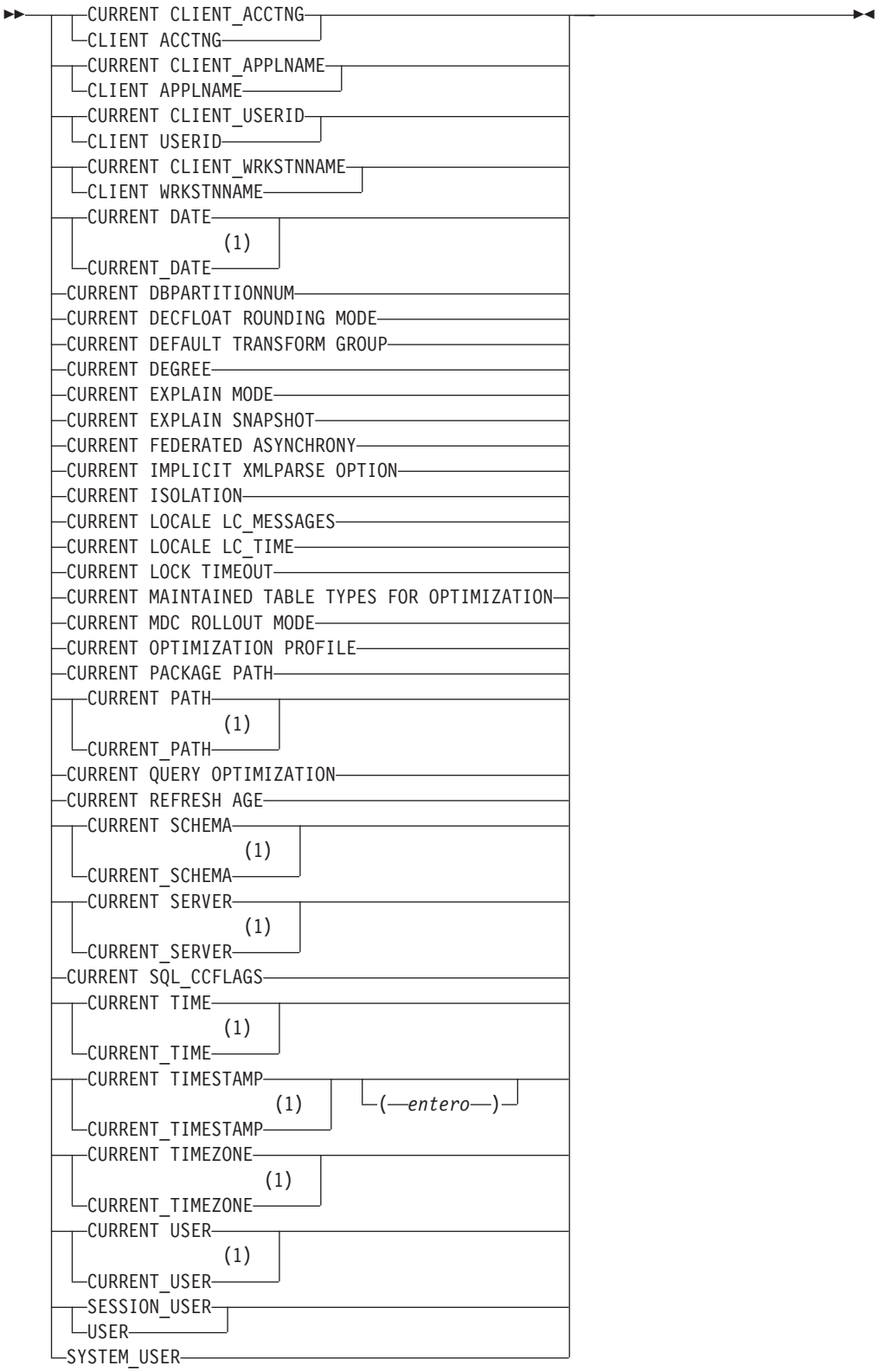
representa el patrón VARGRAPHIC de la serie ASCII 'Bobby'.

Constantes booleanas

Una constante booleana especifica la palabra clave TRUE o FALSE, lo que representa los valores verdaderos true o false respectivamente. El valor verdadero desconocido puede especificarse mediante CAST(NULL AS BOOLEAN).

Registros especiales

Un *registro especial* es un área de almacenamiento que el gestor de bases de datos define para un proceso de aplicación. Se utiliza para almacenar información a la que se puede hacer referencia en sentencias de SQL. Una referencia a un registro especial es una referencia a un valor proporcionado por el servidor actual. Si el valor es una serie, su CCSID es el CCSID por omisión del servidor actual. Se hace referencia a los registros especiales de la forma siguiente:



Notas:

1 El estándar básico de SQL2003 utiliza el formato con el subrayado.

Algunos registros especiales puede actualizarse utilizando la sentencia SET. En la tabla siguiente se muestran los registros especiales que se pueden actualizar así como indicar qué registro especial puede ser el valor nulo.

Tabla 18. Registros especiales actualizables y anulables

Registro especial	Actualizable	Posibil. de nulos
CURRENT CLIENT_ACCTNG	No	No
CURRENT CLIENT_APPLNAME	No	No
CURRENT CLIENT_USERID	No	No
CURRENT CLIENT_WRKSTNNAME	No	No
CURRENT DATE	No	No
CURRENT DBPARTITIONNUM	No	No
CURRENT DECFLOAT ROUNDING MODE	No	No
CURRENT DEFAULT TRANSFORM GROUP	Sí	No
CURRENT DEGREE	Sí	No
CURRENT EXPLAIN MODE	Sí	No
CURRENT EXPLAIN SNAPSHOT	Sí	No
CURRENT FEDERATED ASYNCHRONY	Sí	No
CURRENT IMPLICIT XMLPARSE OPTION	Sí	No
CURRENT ISOLATION	Sí	No
"CURRENT LOCALE LC_MESSAGES" en la página 178	Sí	No
CURRENT LOCALE LC_TIME	Sí	No
CURRENT LOCK TIMEOUT	Sí	Sí
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	Sí	No
CURRENT MDC ROLLOUT MODE	Sí	No
CURRENT OPTIMIZATION PROFILE	Sí	Sí
CURRENT PACKAGE PATH	Sí	No
CURRENT PATH	Sí	No
CURRENT QUERY OPTIMIZATION	Sí	No
CURRENT REFRESH AGE	Sí	No
CURRENT SCHEMA	Sí	No
CURRENT SERVER	No	No
CURRENT SQL_CCFLAGS	Sí	No
CURRENT TIME	No	No
CURRENT TIMESTAMP	No	No
CURRENT TIMEZONE	No	No
CURRENT USER	No	No
SESSION_USER	Sí	No
SYSTEM_USER	No	No
USER	Sí	No

Cuando se hace referencia a un registro especial en una rutina, el valor del registro especial de la rutina depende de si el registro especial es actualizable o no. Para

registros especiales no actualizables, el valor se define en el valor por omisión del registro especial. Para registros especiales actualizables, el valor inicial se hereda del invocador de la rutina y puede modificarse con una sentencia SET posterior dentro de la rutina.

CURRENT CLIENT_ACCTNG

El registro especial CURRENT CLIENT_ACCTNG (o CLIENT ACCTNG) contiene el valor de la serie de contabilidad a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor de la serie de contabilidad puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Obtenga el valor actual de la serie de contabilidad para esta conexión.

```
VALUES (CURRENT CLIENT_ACCTNG)
  INTO :ACCT_STRING
```

CURRENT CLIENT_APPLNAME

El registro especial CLIENT_APPLNAME (o CLIENT APPLNAME) contiene el valor del nombre de la aplicación a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de aplicación puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Seleccione los departamentos a los que se les permite utilizar la aplicación que se está usando en esta conexión.

```
SELECT DEPT
  FROM DEPT_APPL_MAP
  WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

CURRENT CLIENT_USERID

El registro especial CLIENT_USERID (o CLIENT USERID) contiene el valor del ID de usuario de cliente a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del ID de usuario de cliente puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Averigüe en qué departamento funciona el ID de usuario de cliente actual.

```
SELECT DEPT
  FROM DEPT_USERID_MAP
  WHERE USER_ID = CURRENT CLIENT_USERID
```

CURRENT CLIENT_WRKSTNNAME

El registro especial CLIENT_WRKSTNNAME (o CLIENT WRKSTNNAME) contiene el valor del nombre de la estación de trabajo a partir de la información de cliente especificada para esta conexión. El tipo de datos del registro es VARCHAR(255). El valor por omisión de este registro es una serie vacía.

El valor del nombre de estación de trabajo puede cambiarse utilizando la API para Establecer información de cliente (sqleseti).

Tenga en cuenta que el valor proporcionado a través de la API sqleseti está en la página de códigos de aplicación y el valor del registro especial se almacena en la página de códigos de base de datos. En función de los valores de datos utilizados al establecer la información de cliente, puede que, durante la conversión de la página de códigos, se produzca el truncamiento del valor de datos almacenado en el registro especial.

Ejemplo: Obtenga el nombre de estación de trabajo que se está utilizando para esta conexión.

```
VALUES (CURRENT CLIENT_WRKSTNNAME)
  INTO :WS_NAME
```

CURRENT DATE

El registro especial CURRENT DATE (o CURRENT_DATE) especifica una fecha basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT TIME o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT DATE no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT DATE se puede utilizar en una consulta prevista para fuentes de datos. Cuando se procesa la consulta, la fecha devuelta se obtendrá del registro CURRENT DATE, en el servidor federado, no de las fuentes de datos.

Ejemplo: ejecutar el mandato siguiente desde CLP de DB2 para obtener la fecha actual.

```
db2 values CURRENT DATE
```

Ejemplo: Utilizando la tabla PROJECT, establezca la fecha final del proyecto (PRENDATE) del proyecto MA2111 (PROJNO) en la fecha actual.

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DBPARTITIONNUM

El registro especial CURRENT DBPARTITIONNUM especifica un valor INTEGER que identifica el número de nodo coordinador de la sentencia. Para las sentencias emitidas desde una aplicación, el coordinador es la partición de base de datos a la que se conecta la aplicación. Para las sentencias emitidas desde una rutina, el coordinador es la partición de base de datos desde la que se invoca la rutina.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT DBPARTITIONNUM nunca se hereda de la sentencia que la invoca.

CURRENT DBPARTITIONNUM devuelve 0 si la instancia de base de datos no está definida para soportar el particionamiento. (En otras palabras, si no hay ningún archivo db2nodes.cfg. Para las bases de datos particionadas, el archivo db2nodes.cfg existe y contiene las definiciones de partición de base de datos).

Es posible cambiar CURRENT DBPARTITIONNUM mediante la sentencia CONNECT, pero sólo bajo ciertas condiciones.

Para compatibilidad con versiones anteriores a la Versión 8, la palabra clave NODE puede sustituirse por DBPARTITIONNUM.

Ejemplo: Establecer la variable del lenguaje principal APPL_NODE (entero) en el número de la partición de base de datos a la que está conectada la aplicación.

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

CURRENT DECFLOAT ROUNDING MODE

El registro especial CURRENT DECFLOAT ROUNDING MODE especifica la modalidad de redondeo que se utiliza para los valores DECFLOAT.

El tipo de datos es VARCHAR(128). Se soportan las siguientes modalidades de redondeo:

- 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 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.

El valor de la modalidad de redondeo DECFLOAT en un cliente se puede confirmar para que coincida con el del servidor invocando la sentencia SET CURRENT DECFLOAT ROUNDING MODE. Sin embargo, esta sentencia no se puede utilizar para cambiar la modalidad de redondeo del servidor. El valor inicial de CURRENT DECFLOAT ROUNDING MODE lo determina el parámetro de configuración de base de datos **decflt_rounding** y sólo se puede cambiar modificando el valor de este parámetro de configuración de base de datos.

CURRENT DEFAULT TRANSFORM GROUP

El registro especial CURRENT DEFAULT TRANSFORM GROUP especifica un valor VARCHAR(18) que identifica el nombre del grupo de transformación utilizado por las sentencias de SQL dinámicas para intercambiar valores de tipo estructurado definidos por el usuario con programas de lenguaje principal. Este registro especial no especifica los grupos de transformación utilizados en las sentencias de SQL dinámico o en el intercambio de parámetros y resultados con funciones externas o métodos.

Su valor puede definirse mediante la sentencia SET CURRENT DEFAULT TRANSFORM GROUP. Si no se define ningún valor, el valor inicial del registro especial es la serie vacía (VARCHAR con una longitud de cero).

En una sentencia de SQL dinámico (es decir, una sentencia que interacciona con variables del lenguaje principal), el nombre del grupo de transformación utilizado para intercambiar valores es el mismo que el nombre de este registro especial, a menos que el registro contenga la serie vacía. Si el registro contiene la serie vacía (no se ha definido ningún valor utilizando la sentencia SET CURRENT DEFAULT TRANSFORM GROUP), se utiliza el grupo de transformación DB2_PROGRAM para la transformación. Si el grupo de transformación DB2_PROGRAM no está definido para el tipo estructurado indicado, se emite un error durante la ejecución (SQLSTATE 42741).

Ejemplos:

Establezca el grupo de transformación por omisión en MYSTRUCT1. Las funciones TO SQL y FROM SQL definidas en la transformación MYSTRUCT1 se utilizan para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa de lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

Recupere el nombre del grupo de transformación por omisión asignado a este registro especial.

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

El registro especial CURRENT DEGREE especifica el grado de paralelismo intrapartición para la ejecución de sentencias de SQL dinámico. (Para SQL estático, la opción de vinculación de DEGREE proporciona el mismo control.) El tipo de datos del registro es CHAR(5). Los valores válidos son ANY o la representación de serie de un entero entre 1 y 32.767, inclusive.

Si el valor de CURRENT DEGREE representado como un entero es 1 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia no utilizará el paralelismo intrapartición.

Si el valor de CURRENT DEGREE representado como un entero es mayor que 1 y menor que o igual a 32.767 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia puede implicar el paralelismo intrapartición con el grado especificado.

Si el valor de CURRENT DEGREE es ANY cuando una sentencia de SQL se prepara dinámicamente, la ejecución de dicha sentencia puede implicar el paralelismo intrapartición que utiliza un grado determinado por el gestor de bases de datos.

El grado de paralelismo real durante la ejecución será el menor de los valores siguientes:

- El valor del parámetro de configuración del grado de consulta máximo(**max_querydegree**)
- El grado de ejecución de la aplicación
- El grado de compilación de la sentencia de SQL.

Si el parámetro de configuración del gestor de bases de datos **intra_parallel** se establece en NO, el valor del registro especial CURRENT DEGREE se ignorará con el fin de optimizar y la sentencia no utilizará el paralelismo intrapartición.

El valor puede cambiarse invocando la sentencia SET CURRENT DEGREE.

El valor inicial de CURRENT DEGREE lo determina el parámetro de configuración de la base de datos **dft_degree**.

CURRENT EXPLAIN MODE

El registro especial CURRENT EXPLAIN MODE contiene un valor VARCHAR (254) que controla la actuación del recurso Explain con respecto a sentencias de SQL dinámico admisibles. Este recurso genera e inserta información de Explain en las tablas de Explain. Esta información no incluye la instantánea de Explain. Los valores posibles son YES, EXPLAIN, NO, REOPT, RECOMMEND INDEXES y EVALUATE INDEXES. (Para SQL estático, la opción de vinculación de EXPLAIN proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLAIN son: YES, NO y ALL.)

YES Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico se capture al compilar la sentencia.

EXPLAIN
Se habilita el recurso pero no se ejecutan las sentencias dinámicas.

NO Inhabilita el recurso Explain.

REOPT
Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico (o de vinculación incremental) sólo se captura cuando se reoptimice la sentencia utilizando valores reales para las variables de entrada (variables del lenguaje principal, registros especiales, variables globales o marcadores de parámetro).

RECOMMEND INDEXES
Recomienda un conjunto de índices para cada consulta dinámica. Llena la tabla ADVISE_INDEX con el conjunto de índices.

EVALUATE INDEXES
Explica las consultas dinámicas como si existieran los índices recomendados. Los índices se seleccionan de la tabla ADVISE_INDEX.

El valor inicial es NO. Este valor puede cambiarse invocando la sentencia SET CURRENT EXPLAIN MODE.

Los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan al invocar el recurso Explain. El registro especial CURRENT EXPLAIN MODE también interactúa con la opción de vinculación EXPLAIN. RECOMMEND INDEXES y EVALUATE INDEXES sólo se pueden establecer para el registro CURRENT EXPLAIN MODE y deben establecerse utilizando la sentencia SET CURRENT EXPLAIN MODE.

Ejemplo: Establezca la variable del lenguaje principal EXPL_MODE (VARCHAR (254)) en el valor que hay actualmente en el registro especial CURRENT EXPLAIN MODE.

```
VALUES CURRENT EXPLAIN MODE
  INTO :EXPL_MODE
```

CURRENT EXPLAIN SNAPSHOT

El registro especial CURRENT EXPLAIN SNAPSHOT contiene un valor CHAR(8) que controla el comportamiento del recurso de instantáneas de Explain. Este recurso genera información comprimida que incluye información sobre planes de acceso, costes del operador y estadísticas en tiempo de vinculación.

Sólo las sentencias siguientes tienen en cuenta el valor de este registro: CALL, SQL compuesto (dinámico), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES o VALUES INTO. Los valores posibles son YES, EXPLAIN, NO y REOPT. (Para SQL estático, la opción de vinculación EXPLSNAP proporciona el mismo control. En el caso de los mandatos PREP y BIND, los valores de la opción EXPLSNAP son: YES, NO y ALL.)

YES Habilita el recurso de instantáneas de Explain y realiza una instantánea de la representación interna de una sentencia de SQL dinámico al compilar la sentencia.

EXPLAIN
Habilita el recurso de instantáneas de Explain pero no se ejecutan las sentencias dinámicas.

NO Inhabilita el recurso de instantáneas de Explain.

REOPT
Habilita el recurso Explain y hace que la información de Explain para una sentencia de SQL dinámico (o de vinculación incremental) sólo se captura cuando se reoptimice la sentencia utilizando valores reales para las variables de entrada (variables del lenguaje principal, registros especiales, variables globales o marcadores de parámetro).

El valor inicial es NO. Este valor puede cambiarse invocando la sentencia SET CURRENT EXPLAIN SNAPSHOT.

Los valores de los registros especiales CURRENT EXPLAIN SNAPSHOT y CURRENT EXPLAIN MODE interactúan al invocar el recurso Explain. El registro especial CURRENT EXPLAIN SNAPSHOT también interactúa con la opción de vinculación EXPLSNAP.

Ejemplo: Establezca la variable del lenguaje principal EXPL_SNAP (char(8)) en el valor que contiene actualmente el registro especial CURRENT EXPLAIN SNAPSHOT.

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

CURRENT FEDERATED ASYNCHRONY

El registro especial CURRENT FEDERATED ASYNCHRONY especifica el grado de asincronía para la ejecución de sentencias de SQL dinámico. (La opción de vinculación FEDERATED_ASYNCHRONY ofrece el mismo control para el SQL estático.) El tipo de datos del registro es INTEGER. Los valores válidos son ANY (representa -1) o un entero entre 0 y 32.767, incluido. Si, cuando la sentencia de SQL está dinámicamente preparada, el valor de CURRENT FEDERATED ASYNCHRONY es:

- 0, la ejecución de esa sentencia no utilizará la asincronía
- mayor que 0 o menor o igual que 32.767, la ejecución de esa sentencia puede implicar asincronía utilizando el grado especificado
- ANY (representa -1), la ejecución de esa sentencia puede implicar que la asincronía utilice un grado que esté determinado por el gestor de bases de datos

El valor del registro especial CURRENT FEDERATED ASYNCHRONY puede modificarse invocando la sentencia SET CURRENT FEDERATED ASYNCHRONY.

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_ASYNCHRONY determina el valor inicial si la sentencia dinámica forma parte de una aplicación que se está vinculando.

Ejemplo: Establecer la variable del lenguaje principal FEDASYNC (INTEGER) en el valor del registro especial CURRENT FEDERATED ASYNCHRONY.

```
VALUES CURRENT FEDERATED ASYNCHRONY INTO :FEDASYNC
```

CURRENT IMPLICIT XMLPARSE OPTION

El registro especial CURRENT IMPLICIT XMLPARSE OPTION especifica las opciones de tratamiento de los espacios en blanco que se utilizarán cuando el servidor DB2 analice implícitamente los datos XML serializados. Se produce una operación de análisis implícito no validante cuando una sentencia SQL procesa una variable del lenguaje principal XML o un marcador de parámetro XML escrito implícita o explícitamente que no es un argumento de la función XMLVALIDATE. El tipo de datos del registro es VARCHAR(19).

El valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION puede modificarse invocando la sentencia SET CURRENT IMPLICIT XMLPARSE OPTION. Su valor inicial es 'STRIP WHITESPACE'.

Ejemplos:

Recuperar el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION en la variable del lenguaje principal CURXMLPARSEOPT:

```
EXEC SQL VALUES (CURRENT IMPLICIT XMLPARSE OPTION) INTO :CURXMLPARSEOPT;
```

Establezca el registro especial CURRENT IMPLICIT XMLPARSE OPTION en 'PRESERVE WHITESPACE'.

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

Se conserva el espacio en blanco cuando se ejecuta la siguiente sentencia SQL:

```
INSERT INTO T1 (XMLCOL1) VALUES (?)
```

CURRENT ISOLATION

El registro especial CURRENT ISOLATION mantiene un valor CHAR(2) que identifica el nivel de aislamiento (en relación a otras sesiones simultáneas) correspondiente a las sentencias de SQL dinámico emitidas desde la sesión actual.

Los valores posibles son:

(blancos)
Sin definir; se utiliza el atributo de aislamiento del paquete.

UR Lectura no confirmada

CS Estabilidad del cursor

RR Lectura repetible

RS Estabilidad de lectura

El valor del registro especial CURRENT ISOLATION se puede cambiar mediante la sentencia SET CURRENT ISOLATION.

Hasta que se emita una sentencia SET CURRENT ISOLATION en una sesión o después de especificar RESET para SET CURRENT ISOLATION, el registro especial CURRENT ISOLATION está establecido en blancos y no se aplica a sentencias de SQL dinámico; el nivel de aislamiento utilizado se toma del atributo de aislamiento del paquete que ha emitido la sentencia de SQL dinámico. Una vez emitida una sentencia SET CURRENT ISOLATION, el registro especial CURRENT ISOLATION proporciona el nivel de aislamiento correspondiente a cualquier sentencia de SQL dinámico siguiente compilada dentro de la sesión, con independencia de los valores del paquete que emita la sentencia. Esto permanecerá vigente hasta que finalice la sesión o hasta que se emita una sentencia SET CURRENT ISOLATION con la opción RESET.

Ejemplo: Establecer para la variable del lenguaje principal ISOLATION_MODE (CHAR(2)) el valor actualmente almacenado en el registro especial CURRENT ISOLATION.

```
VALUES CURRENT ISOLATION
  INTO :ISOLATION_MODE
```

CURRENT LOCALE LC_MESSAGES

El registro especial CURRENT LOCALE LC_MESSAGES identifica el entorno local que se utiliza supervisando las rutinas en el módulo **monreport**.

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.

El tipo de datos es VARCHAR(128).

El valor inicial de CURRENT LOCALE LC_MESSAGES es ″en_US″ para el inglés (Estados Unidos). El valor puede cambiarse invocando la sentencia SET CURRENT LOCALE LC_MESSAGES.

CURRENT LOCALE LC_TIME

El registro especial CURRENT LOCALE LC_TIME identifica el entorno local que se utiliza para las sentencias de SQL que implican las funciones incorporadas relacionadas con la fecha y hora DAYNAME, MONTHNAME, NEXT_DAY, ROUND, ROUND_TIMESTAMP, TIMESTAMP_FORMAT, TRUNCATE, TRUNC_TIMESTAMP y VARCHAR_FORMAT. Estas funciones utilizan el valor de CURRENT LOCALE LC_TIME si el argumento *nombre-entorno-local* no está especificado de forma explícita.

El tipo de datos es VARCHAR(128).

El valor inicial de CURRENT LOCALE LC_TIME es “en_US” para el inglés (Estados Unidos). El valor puede cambiarse invocando la sentencia SET CURRENT LOCALE LC_TIME.

CURRENT LOCK TIMEOUT

El registro especial CURRENT LOCK TIMEOUT especifica el número de segundos que debe esperarse un bloqueo antes de devolver un error que indique que no es posible obtener un bloqueo. Este registro especial afecta a los bloqueos de fila, tabla, clave de índice, bloque MDC y vía de acceso a XML (XPath). El tipo de datos del registro es INTEGER.

Los valores válidos para el registro especial CURRENT LOCK TIMEOUT son los enteros comprendidos entre -1 y 32767, ambos inclusive. Este registro especial también puede establecerse en un valor nulo. Un valor de -1 especifica que no deben producirse tiempos de espera excedidos y que la aplicación debe esperar hasta que se libere el bloqueo o se detecte un punto muerto. Un valor de 0 especifica que la aplicación no debe esperar un bloqueo; si no es posible obtener un bloqueo, debe devolverse un error inmediatamente.

El valor del registro especial CURRENT LOCK TIMEOUT puede modificarse invocando la sentencia SET CURRENT LOCK TIMEOUT. Su valor inicial es nulo; en este caso, se utiliza el valor actual del parámetro de configuración de la base de datos **locktimeout** al esperar un bloqueo y se devolverá este valor para el registro especial.

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

El registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION especifica un valor VARCHAR (254) que identifica los tipos de tablas que pueden tenerse en cuenta al optimizar el proceso de las series de SQL dinámico. Las consultas de SQL incorporado estático nunca tienen en cuenta las tablas de consultas materializadas.

El valor inicial de CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION es SYSTEM. Su valor puede cambiarse con la sentencia SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

CURRENT MDC ROLLOUT MODE

El registro especial CURRENT MDC ROLLOUT MODE especifica el comportamiento de las tablas de clústers de varias dimensiones (MDC) de las sentencias DELETE calificadas para el proceso de lanzamiento..

El valor por omisión de este registro se determina por medio de la variable de registro DB2_MDC_ROLLOUT. El valor puede cambiarse invocando la sentencia SET CURRENT MDC ROLLOUT MODE. Cuando el registro especial CURRENT MDC ROLLOUT MODE se establece en un valor particular, resultará afectado el comportamiento de ejecución de las sucesivas sentencias DELETE calificadas para el lanzamiento. La sentencia DELETE no ha de volverse a compilar para que cambie el comportamiento.

CURRENT OPTIMIZATION PROFILE

El registro especial CURRENT OPTIMIZATION PROFILE especifica el nombre calificado del perfil de optimización que debe ser utilizado por las sentencias DML que se preparan dinámicamente para la optimización.

El valor inicial es el valor nulo. El valor se puede cambiar invocando la sentencia SET CURRENT OPTIMIZATION PROFILE. Un perfil de optimización que no está calificado con un nombre de esquema se calificará de forma implícita con el valor del registro especial CURRENT DEFAULT SCHEMA.

Ejemplo 1: Establezca el perfil de optimización en 'JON.SALES'.

```
SET CURRENT OPTIMIZATION PROFILE = JON.SALES
```

Ejemplo 2: Obtenga el valor actual del nombre de perfil optimización para esta conexión.

```
VALUES (CURRENT OPTIMIZATION PROFILE) INTO :PROFILE
```

CURRENT PACKAGE PATH

El registro especial CURRENT PATH especifica un valor VARCHAR(4096) que identifica la vía de acceso que se ha de utilizar para resolver las referencias a paquetes que son necesarias al ejecutar sentencias de SQL.

El valor puede ser una serie vacía o en blanco o una lista de uno o varios nombres de esquema delimitados por comillas dobles y separados por comas. Las comillas dobles que aparezcan como parte de la serie deberán representarse como dos comillas dobles, como suele hacerse con los identificadores delimitados. Los delimitadores y las comas contribuyen a la longitud del registro especial.

Este registro especial se aplica a sentencias tanto estáticas como dinámicas.

El valor inicial de CURRENT PACKAGE PATH en una función, un método o un procedimiento definido por el usuario se hereda de la aplicación que lo invoca. En otros contextos, el valor inicial de CURRENT PACKAGE PATH es una serie vacía. El valor sólo es una lista de esquemas si el proceso de la aplicación ha especificado de forma explícita una lista de esquemas mediante la sentencia SET CURRENT PACKAGE PATH.

Ejemplos:

Una aplicación utilizará varios paquetes SQLJ(en los esquemas SQLJ1 y SQLJ2) y paquete JDBC (en el esquema DB2JAVA). El registro especial CURRENT PACKAGE PATH debe establecerse para comprobar SQLJ1, SQLJ2 y DB2JAVA, en este orden.

```
SET CURRENT PACKAGE PATH = "SQLJ1", "SQLJ2", "DB2JAVA"
```

La variable del lenguaje principal HVPKLIST debe establecerse en el valor almacenado actualmente en el registro especial CURRENT PACKAGE PATH.

```
VALUES CURRENT PACKAGE PATH INTO :HVPKLIST
```

CURRENT PATH

El registro especial CURRENT PATH (o CURRENT_PATH) especifica un valor VARCHAR(2048) que identifica la vía de acceso de SQL que se utiliza al resolver nombres de función no calificados, nombres de procedimiento, nombres de tipo de datos, nombres de variable global y nombres de objeto de módulo en sentencias de SQL preparadas de forma dinámica. CURRENT FUNCTION PATH es sinónimo de CURRENT PATH. El valor inicial es el valor por omisión que se especifica más abajo. Para SQL estático, la opción de vinculación FUNCPATH proporciona una vía de acceso de SQL que se utiliza para la resolución de funciones y tipos de datos.

El registro especial CURRENT PATH contiene una lista de uno o varios nombres de esquema escritos entre comillas dobles y separados por comas. Por ejemplo, una vía de acceso de SQL que especifica que el gestor de bases de datos primero debe mirar en el esquema FERMAT, luego en el esquema XGRAPHIC y por último en el esquema SYSIBM se devuelve en el registro especial CURRENT PATH de la siguiente manera:

```
"FERMAT","XGRAPHIC","SYSIBM"
```

El valor por omisión es “SYSIBM”,“SYSFUN”,“SYSPROC”,“SYSIBMADM”,*X*, donde *X* es el valor del registro especial USER, delimitado por comillas dobles. El valor puede cambiarse invocando la sentencia SET CURRENT PATH. No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema. SYSIBM no toma ninguno de los 2048 bytes si se asume implícitamente.

Un tipo de datos que no está calificado con un nombre de esquema se calificará implícitamente con el primer esquema de la vía de acceso de SQL que contenga un tipo de datos con el mismo nombre no calificado. Existen excepciones a esta norma, como se indica en las descripciones de las sentencias siguientes: CREATE TYPE (Diferenciado), CREATE FUNCTION, COMMENT y DROP.

Ejemplo: Utilizando la vista de catálogos SYSCAT.ROUTINES, busque todas las rutinas definidas por el usuario que pueden invocarse sin calificar el nombre de rutina, ya que el registro especial CURRENT PATH contiene el nombre de esquema.

```
SELECT ROUTINENAME, ROUTINESCHEMA FROM SYSCAT.ROUTINES
  WHERE POSITION (ROUTINESCHEMA, CURRENT PATH, CODEUNITS16) <> 0
```

CURRENT QUERY OPTIMIZATION

El registro especial CURRENT QUERY OPTIMIZATION especifica un valor INTEGER que controla la clase de optimización de consulta que realiza el gestor de bases de datos al vincular sentencias de SQL dinámico. La opción de vinculación QUERYOPT controla la clase de optimización de consulta para las sentencias de SQL estático. Los valores posibles oscilan entre 0 y 9. Por ejemplo, si la clase de optimización de consulta se establece en 0 (optimización mínima), el valor del registro especial es 0. El valor por omisión está determinado por el parámetro de configuración de la base de datos **dft_queryopt**. El valor puede cambiarse invocando la sentencia SET CURRENT QUERY OPTIMIZATION.

Ejemplo: Utilizando la vista de catálogo SYSCAT.PACKAGES, busque todos los planes que se han vinculado con el mismo valor que el valor actual del registro especial CURRENT QUERY OPTIMIZATION.

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

CURRENT REFRESH AGE

El registro especial CURRENT REFRESH AGE especifica un valor de duración de indicación de fecha y hora con un tipo de datos de DECIMAL(20,6). Es la duración máxima desde que se produjo un suceso de indicación de fecha y hora concreto en un objeto de datos de la antememoria (por ejemplo, una sentencia REFRESH TABLE procesada en una tabla de consultas materializadas REFRESH DEFERRED mantenida por el sistema) de forma que el objeto de datos de la antememoria pueda utilizarse para optimizar el proceso de la consulta. Si CURRENT REFRESH AGE tiene un valor de 99 999 999 999 999 y la clase de optimización de consulta es 5 o superior, se tienen en cuenta los tipos de tablas especificadas en CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION al optimizar el proceso de una consulta de SQL dinámico.

El valor de CURRENT REFRESH AGE debe ser 0 o 99 999 999 999 999. El valor inicial es 0. Se puede cambiar el valor invocando la sentencia SET CURRENT REFRESH AGE.

CURRENT SCHEMA

El registro especial CURRENT SCHEMA (o CURRENT_SCHEMA) especifica un valor VARCHAR(128) que identifica el nombre de esquema utilizado para calificar las referencias a objetos de base de datos, donde corresponda, en sentencias de SQL preparadas dinámicamente. Para mantener la compatibilidad con DB2 para z/OS, se puede especificar CURRENT SQLID (o CURRENT_SQLID) en lugar de CURRENT SCHEMA.

El valor inicial de CURRENT SCHEMA es el ID de autorización del usuario de la sesión actual. El valor puede cambiarse invocando la sentencia SET SCHEMA.

La opción de vinculación QUALIFIER controla el nombre de esquema utilizado para calificar las referencias a objetos de base de datos, donde corresponda, para sentencias de SQL estático.

Ejemplo: Establezca el esquema para la calificación de objetos en 'D123'.

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

El registro especial CURRENT SERVER (o CURRENT_SERVER) especifica un valor VARCHAR(18) que identifica el servidor de aplicaciones actual. El registro contiene el nombre real del servidor de aplicaciones, no un alias.

Es posible cambiar CURRENT SERVER mediante la sentencia CONNECT, pero sólo bajo ciertas condiciones.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT SERVER no se hereda de la sentencia que la invoca.

Ejemplo: Establezca la variable del lenguaje principal APPL_SERVE (VARCHAR(18)) en el nombre del servidor de aplicaciones al que está conectada la aplicación.

```
VALUES CURRENT SERVER INTO :APPL_SERVE
```

CURRENT SQL_CCFLAGS

El registro especial CURRENT SQL_CCFLAGS especifica las constantes con nombre de compilación condicional que se definen para utilizarse durante la compilación de las sentencias SQL.

El tipo de datos del registro especial es VARCHAR(1024).

El registro especial CURRENT SQL_CCFLAGS contiene una lista de pares de nombres y valores separados mediante una coma y un espacio en blanco. En un par, el nombre se separa del valor mediante la utilización de un carácter de dos puntos. Los valores de la lista son una constante de tipo BOOLEAN, una constante de tipo INTEGER o la palabra clave NULL. Los nombres pueden especificarse mediante la utilización de cualquier combinación de caracteres en mayúsculas o en minúsculas, que se convierten todos en caracteres en mayúsculas. Por ejemplo, en el registro especial, los valores de compilación condicional que se han definido para la depuración y el rastreo podrían aparecer como el valor de serie:

```
CC_DEBUG:TRUE, CC_TRACE_LEVEL:2
```

El valor inicial del registro especial es el valor que tiene el parámetro de configuración de base de datos **sql_ccflags** cuando se utiliza por primera vez el registro especial. El primer uso puede tener lugar como resultado de procesar una sentencia con una directiva de consulta o como referencia directa al registro especial. Si el valor asignado al parámetro de configuración **sql_ccflags** no es válido, se devuelve un error en el primer uso (SQLSTATE 42815 o 428HV).

El valor del registro especial puede cambiarse mediante la ejecución de la sentencia SET CURRENT SQL_CCFLAGS.

CURRENT TIME

El registro especial CURRENT TIME (o CURRENT_TIME) especifica una hora basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIMESTAMP en una sola sentencia, todos los valores se basan en la misma lectura del reloj.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIME no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT TIME se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la hora devuelta se obtendrá del registro CURRENT TIME del servidor federado, no de las fuentes de datos.

Ejemplo: ejecutar el mandato siguiente desde CLP de DB2 para obtener la hora actual.

```
db2 values CURRENT TIME
```

Ejemplo: Utilizando la tabla CL_SCHED, seleccione todas las clases (CLASS_CODE) que empiezan (STARTING) más tarde hoy. Las clases de hoy tienen un valor 3 en la columna DAY.

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

El registro especial CURRENT TIMESTAMP (o CURRENT_TIMESTAMP) especifica una indicación de fecha y hora basada en la lectura del reloj cuando se ejecuta la sentencia de SQL en el servidor de aplicaciones. Si este registro especial se utiliza más de una vez en la misma sentencia de SQL o bien con CURRENT DATE o CURRENT TIME en una sola sentencia, todos los valores se basan en la misma lectura del reloj. Para las peticiones del registro especial CURRENT TIMESTAMP separadas, es posible devolver el mismo valor; si se necesitan valores exclusivos, considere la utilización de la función GENERATE_UNIQUE, de una secuencia o de una columna de identidad.

Si se desea una indicación de fecha y hora con una precisión específica, se puede hacer referencia al registro especial como CURRENT TIMESTAMP(*entero*), donde el valor *entero* puede estar comprendido entre 0 y 12. La precisión por omisión es 6. La precisión de la lectura de reloj varía en función de la plataforma, y el valor resultante se rellena con ceros en los casos en los que la precisión de la lectura de reloj recuperada es inferior a la precisión de la petición.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIMESTAMP no se hereda de la sentencia que la invoca.

En un sistema federado, CURRENT TIMESTAMP se puede utilizar en una consulta destinada a fuentes de datos. Cuando se procesa la consulta, la indicación de fecha y hora se obtendrá del registro CURRENT TIMESTAMP del servidor federado, no de las fuentes de datos.

También se puede especificar SYSDATE como sinónimo de CURRENT TIMESTAMP(0).

Ejemplo: Inserte una fila en la tabla IN_TRAY. El valor de la columna RECEIVED mostrará la indicación de fecha y hora en la que se ha añadido la fila. Los valores de las otras tres columnas se obtienen de las variables del lenguaje principal SRC (char(8)), SUB (char(64)) y TXT (VARCHAR(200)).

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

El registro especial CURRENT TIMEZONE (o CURRENT_TIMEZONE) especifica la diferencia entre UTC (Hora coordinada universal, conocida anteriormente como GMT) y la hora local del servidor de aplicaciones. La diferencia se representa por un período de tiempo (un número decimal cuyos dos primeros dígitos corresponden a las horas, los dos siguientes a los minutos y los dos últimos a los segundos). El número de horas está entre -24 y 24, exclusive. Al restar CURRENT TIMEZONE de la hora local se convierte esa hora a UTC. La hora se calcula a partir de la hora del sistema operativo en el momento en que se ejecuta la sentencia de SQL. (El valor de CURRENT TIMEZONE se determina a partir de las funciones de tiempo de ejecución de C).

El registro especial CURRENT TIMEZONE se puede utilizar allí donde se utilice una expresión de tipo de datos DECIMAL(6,0); por ejemplo, en operaciones aritméticas de hora e indicación de fecha y hora.

Cuando se utiliza en una sentencia de SQL incluida en una rutina, CURRENT TIMEZONE no se hereda de la sentencia que la invoca.

Ejemplo: Inserte un registro en la tabla IN_TRAY utilizando una indicación de fecha y hora UTC para la columna RECEIVED.

```
INSERT INTO IN_TRAY VALUES (
  CURRENT TIMESTAMP - CURRENT TIMEZONE,
  :source,
  :subject,
  :notetext )
```

CURRENT USER

El registro especial CURRENT USER (o CURRENT_USER) especifica el ID de autorización que se va a utilizar para autorización de sentencia. Para sentencias de SQL estático, el valor representa el ID de autorización que se utilizó cuando se vinculó el paquete. Para sentencias de SQL dinámico, el valor es el mismo que el valor del registro especial SESSION_USER para paquetes vinculados con la opción de vinculación DYNAMICRULES(RUN). El tipo de datos del registro es VARCHAR(128).

Ejemplo: Seleccione nombres de tablas cuyo esquema coincida con el valor del registro especial CURRENT USER.

```
SELECT TABNAME FROM SYSCAT.TABLES
  WHERE TABSCHEMA = CURRENT USER AND TYPE = 'T'
```

Si esta sentencia se ejecuta como una sentencia de SQL estático, devuelve las tablas cuyo nombre de esquema coincide con el vinculador del paquete que incluye la sentencia. Si esta sentencia se ejecuta como una sentencia de SQL dinámico, devuelve las tablas cuyo nombre de esquema coincide con el valor actual del registro especial SESSION_USER.

SESSION_USER

El registro especial SESSION_USER especifica el ID de autorización que se debe utilizar para la sesión actual. El valor de este registro se utiliza para la comprobación de autorización de sentencias de SQL dinámico cuando el comportamiento de ejecución DYNAMICRULES está en vigor para el paquete. El tipo de datos del registro es VARCHAR(128).

El valor inicial de SESSION_USER para una nueva conexión es el mismo que el valor del registro especial SYSTEM_USER. Su valor se puede modificar invocando la sentencia SET SESSION AUTHORIZATION.

SESSION_USER es sinónimo del registro especial USER.

Ejemplo: Determine qué rutinas se pueden ejecutar utilizando SQL dinámico. Supongamos que el comportamiento de ejecución DYNAMICRULES está en vigor para el paquete que emitirá la sentencia de SQL dinámico que invoca la rutina.

```
SELECT SCHEMA, SPECIFICNAME FROM SYSCAT.ROUTINEAUTH
  WHERE GRANTEE = SESSION_USER
  AND EXECUTEAUTH IN ('Y', 'G')
```

SYSTEM_USER

El registro especial SYSTEM_USER especifica el ID de autorización del usuario que se ha conectado a la base de datos. El valor de este registro sólo se puede modificar conectándose como un usuario con otro ID de autorización. El tipo de datos del registro es VARCHAR(128).

Consulte “Ejemplo” en la descripción de la sentencia SET SESSION AUTHORIZATION.

USER

El registro especial USER especifica el ID de autorización de tiempo de ejecución que se pasa al gestor de bases de datos cuando se inicia una aplicación en una base de datos. El tipo de datos del registro es VARCHAR(128).

Cuando se utiliza en una sentencia de SQL incluida en una rutina, USER no se hereda de la sentencia que lo invoca.

Ejemplo: Seleccione todas las notas de la tabla IN_TRAY que el usuario haya colocado ahí.

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

Variables globales

Las variables globales son variables de memoria con nombre a las que se puede acceder y que pueden modificarse por medio de sentencias de SQL.

Las variables globales le permiten compartir datos relacionales entre sentencias de SQL sin que sea necesario que la lógica de la aplicación dé soporte esta transferencia de datos. Puede controlar el acceso a las variables globales por medio de las sentencias GRANT (privilegios de variables globales) y REVOKE (privilegios de variables globales).

DB2 da soporte a las variables globales de sesión creadas. Una *variable global de sesión* está asociada a una sesión específica y contiene un valor que es exclusivo para cada sesión. Una variable global de sesión creada está disponible para cada sentencia de SQL activa que se ejecuta frente a la base de datos en la que se ha definido la variable. Una variable global de sesión puede asociarse con más de una sesión, pero su valor será específico para cada sesión. Las variables globales de sesión y los privilegios asociados con las mismas se definen en el catálogo del sistema.

Las *variables globales de sesión definidas por el usuario* son variables globales que se crean mediante una sentencia de definición de datos SQL y que se registran en el gestor de bases de datos en el catálogo. Una variable global reside en el esquema donde se ha creado o en el módulo donde se ha añadido o publicado. Las *variables de esquema* se crean mediante la sentencia CREATE VARIABLE. Para obtener más información, consulte "CREATE VARIABLE". Las *variables de módulo* se crean mediante la cláusula ADD *definición-variable-módulo* o PUBLISH *definición-variable-módulo* de la sentencia ALTER MODULE. Para obtener más información, consulte "ALTER MODULE".

La resolución de una referencia de variable global depende del contexto en el que se hace referencia a la variable global y el modo en que el nombre de la variable global se haya calificado. Una referencia de variable que pretenda ser una variable global podría resolverse en una variable de SQL, un parámetro de SQL o un nombre de columna según el contexto de la referencia y cómo se califique la referencia en ese contexto. Los pasos de resolución siguientes presuponen que la referencia de variable global no se resuelve en una variable de SQL, un parámetro de SQL ni un nombre de columna:

- Si el nombre de la variable global se califica, la resolución se lleva a cabo mediante el gestor de bases de datos siguiendo estos pasos:
  1. Si la referencia de variable global procede de un módulo y el calificador coincide con el nombre del módulo en el que se hace referencia a la variable global, se busca en el módulo una variable de módulo coincidente. 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 una variable de módulo coincide con el nombre de variable global no calificado de la referencia, la resolución ha finalizado. Si el calificador no coincide o no hay ninguna variable de módulo coincidente, la resolución continúa con el paso siguiente.
  2. El calificador se considera un nombre de esquema y se busca en dicho esquema una variable de esquema coincidente. Si una variable de esquema coincide con el nombre de variable global no calificado de la referencia, la resolución ha finalizado. Si el esquema no existe o no hay variables de

esquema coincidentes en el esquema, y el calificador ha coincidido con el nombre del módulo del primer paso, se devolverá un error (SQLSTATE 42703). De lo contrario, la resolución continúa con el paso siguiente.

3. El calificador se considera un nombre de módulo.
   - Si el nombre de módulo está calificado con un nombre de esquema, se busca en ese módulo una variable de módulo publicada coincidente.
   - 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, se busca en ese módulo una variable de módulo publicada coincidente.
   - Si el módulo no se encuentra utilizando la vía de acceso de SQL, se tendrá en cuenta la existencia de un alias público de módulo que coincida con el calificador de la variable global. Si se encuentra, en el módulo asociado con el alias público de módulo se busca una variable de módulo publicada coincidente.

   Si una variable de módulo publicada coincide con el nombre de variable global no calificado de la referencia de variable global, la resolución ha finalizado. Si no se encuentra un módulo coincidente o no hay ninguna variable de módulo coincidente en el módulo coincidente, se devuelve un error (SQLSTATE 42703).

- Si el nombre de la variable global no se califica, la resolución se lleva a cabo mediante el gestor de bases de datos siguiendo estos pasos:
  1. Si la referencia de variable global no calificada procede de un objeto de módulo, se busca en el módulo una variable de módulo coincidente. Si una variable de módulo coincide con el nombre de variable global de la referencia, la resolución ha finalizado. Si no hay ninguna variable de módulo coincidente, la resolución continúa con el paso siguiente.
  2. En los esquemas de la vía de acceso de SQL se busca, de izquierda a derecha, una variable de esquema coincidente. Si una variable de esquema coincide con el nombre de variable global de la referencia, la resolución ha finalizado.

  Si no se encuentra ninguna variable global coincidente tras realizar el paso 2, se devuelve un error (SQLSTATE 42703).

Cuando se hace referencia a una variable global en una sentencia de SQL o en un activador, vista o rutina, se registra una dependencia del nombre de variable global totalmente calificado para la sentencia u objeto. La autorización necesaria para una variable global depende de dónde esté definida y cómo se utilice.

- El ID de autorización de una sentencia de SQL que hace referencia a una variable de esquema y recupera el valor debe tener el privilegio READ sobre la variable global.
- El ID de autorización de una sentencia de SQL que hace referencia a una variable de esquema y asigna un valor debe tener el privilegio WRITE sobre la variable global.
- El ID de autorización de una sentencia de SQL que hace referencia a una variable de módulo y recupera el valor o asigna un valor debe tener el privilegio EXECUTE sobre el módulo de la variable global.

Puede hacerse referencia a las variables globales de cualquier expresión que no tenga que ser determinante. Las expresiones determinantes se necesitan en las situaciones siguientes que impiden el uso de variables globales:

- Restricciones de comprobación
- Definiciones de columnas generadas

- Renovar tablas de consulta materializada (MQT) inmediatas

El valor de una variable global puede cambiarse utilizando la sentencia EXECUTE, FETCH, SET, SELECT INTO o VALUES INTO. También puede cambiarse si es un argumento de un parámetro OUT o INOUT en una sentencia CALL o invocación de función.

La tabla siguiente muestra en qué punto el valor de una variable global se lee para la referencia indicada de la variable global.

Tabla 19. Cuándo se lee el valor de una variable global según el contexto

Contexto de una referencia de variable global:	**La referencia utiliza el valor de la variable global al principio de:**
Una sentencia en una sentencia de SQL compuesto (en línea)	La sentencia de SQL compuesto SQL (en línea)
Una sentencia en una sentencia de SQL compuesto (compilado)	La sentencia en la sentencia de SQL compuesto (compilado)
Una sentencia, que posiblemente invoque una función o active un activador[1]	La sentencia de SQL
Una sentencia en una función de SQL en línea invocada	La sentencia de SQL que invoca la función de SQL en línea
Una sentencia en un activador en línea activado	La sentencia de SQL que activa el activador en línea
Una sentencia en un método de SQL en línea invocado	La sentencia de SQL que invoca el método de SQL
Una sentencia en una función de SQL compilado invocado	La sentencia de SQL en la función de SQL compilado
Una sentencia en un activador compilado activado	La sentencia de SQL en el activador compilado
Una sentencia en una rutina externa invocada	La sentencia de SQL en el programa externo
Nota: En esta tabla, la sentencia de SQL, que puede invocar una función o activar un activador, no incluye la sentencia de SQL compuesto (en línea) ni la sentencia de SQL compuesto (compilado).	

Si el tipo de datos de la variable global es un tipo de cursor, se podrá hacer referencia al cursor subyacente de la variable de cursor global en cualquier lugar en el se pueda especificar un *nombre-variable-cursor*.

Si el tipo de datos de la variable global es un tipo de fila, se podrá hacer referencia a un campo de la variable de fila global en cualquier lugar en el se pueda hacer referencia a una variable global con el mismo tipo que el campo. El nombre de la variable global que califica al nombre de campo se resuelve de la misma manera que el resto de las variables globales.

Funciones

Una *función* es una operación denotada por un nombre de función seguido por uno o más operandos que se incluyen entre paréntesis. Por ejemplo, se pueden pasar a la función TIMESTAMP valores de datos de entrada del tipo DATE y TIME y el resultado será TIMESTAMP. Las funciones pueden estar incorporadas o puede definirlas el usuario.

- Con el gestor de bases de datos se proporcionan *funciones incorporadas*. Devuelven un solo valor del resultado y se identifican como parte del esquema SYSIBM. Tales funciones son las funciones agregadas (por ejemplo, AVG), las funciones de operador (por ejemplo, +), las funciones de conversión (por ejemplo, DECIMAL) y las funciones escalares (por ejemplo, CEILING).
- Las *funciones definidas por el usuario* son funciones que se crean mediante una sentencia de definición de datos SQL y que se registran en el gestor de bases de datos en el catálogo. Las *funciones de esquema definidas por el usuario* se crean mediante la sentencia CREATE FUNCTION. Para obtener más información, consulte "CREATE FUNCTION". Se proporciona un conjunto de *funciones de esquema definidas por el usuario* con el gestor de bases de datos en un esquema denominado SYSFUN. Las *funciones de módulo definidas por el usuario* se crean mediante las sentencias ALTER MODULE ADD FUNCTION o ALTER MODULE PUBLISH FUNCTION. Para obtener más información, consulte "ALTER MODULE". Se proporciona un conjunto de *funciones de módulo definidas por el usuario* con el gestor de bases de datos en un conjunto de módulos en un esquema denominado SYSIBMADM. Una *función definida por el usuario* reside en el esquema donde se ha creado o en el módulo donde se ha añadido o publicado.

  Las funciones definidas por el usuario amplían las posibilidades del sistema de bases de datos añadiendo definiciones de funciones (proporcionadas por usuarios o proveedores) que pueden aplicarse en el propio núcleo de la base de datos. La ampliación de las funciones de la base de datos permite que la base de datos explote las mismas funciones en su núcleo que las que utiliza una aplicación, proporcionando más sinergia entre la aplicación y la base de datos.

Funciones definidas por el usuario externas, de SQL y con fuente

Una función definida por el usuario puede ser una función externa, una función de SQL o una función con fuente. Una *función externa* se define en la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque la función. Las funciones externas no pueden ser funciones agregadas. Una función SQL se define en la base de datos sólo a través de sentencias de SQL, incluida, como mínimo, una sentencia RETURN. Puede devolver un valor escalar, una fila o una tabla. Las funciones SQL no pueden ser funciones agregadas. Una *función con fuente* se define para la base de datos con una referencia a otra función incorporada o definida por el usuario que ya se conoce en la base de datos. Las funciones con fuente pueden ser funciones escalares o funciones agregadas. Son útiles para dar soporte a funciones existentes con tipos definidos por el usuario.

Funciones definidas por el usuario escalares, agregadas, de fila y tabla

Cada función definida por el usuario también se clasifica como escalar, agregada, de fila o de tabla. Una *función escalar* es una función que devuelve una respuesta

de un solo valor cada vez que se invoca. Por ejemplo, la función incorporada SUBSTR() es una función escalar. Las UDF escalares pueden ser externas o con fuente.

Una *función agregada* es aquella a la que se pasa conceptualmente un conjunto de valores similares (una columna) y que devuelve una respuesta con un solo valor. Un ejemplo de una función agregada es la función incorporada AVG(). Una UDF de columna externa no puede definirse en DB2, pero puede definirse una UDF de columna, que se origine en una de las funciones agregadas incorporadas. Es útil para tipos diferenciados. Por ejemplo, si hay definido un tipo diferenciado SHOESIZE con el tipo base INTEGER, se puede definir una UDF AVG(SHOESIZE), que se origine en la función incorporada AVG(INTEGER) y se trataría de una función agregada.

Una *función de fila* es una función que devuelve una fila de valores. Puede utilizarse en un contexto que da soporte a una expresión de fila. También se puede utilizar como función de transformación, que correlaciona valores de atributos de un tipo estructurado con valores de una fila. Las funciones de fila deben estar definidas como funciones de SQL.

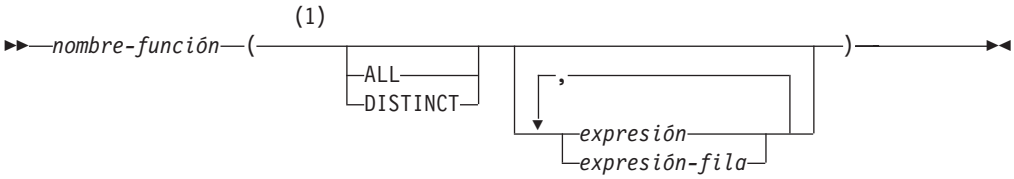
Una *función de tabla* es una función que devuelve una tabla a la sentencia de SQL donde se invoca la función. Sólo se puede hacer referencia a la función en la cláusula FROM de una sentencia SELECT. Una función así puede utilizarse para aplicar la potencia de proceso del lenguaje SQL a datos que no son de o para DB2 convertir tales datos a una tabla DB2. Una función de tabla puede leer un archivo, obtener datos de Internet o acceder a una base de datos de Lotus Notes y devolver una tabla de resultados. Esta información puede unirse a otras tablas de la base de datos. Una función de tabla se puede definir como una función externa o como una función de SQL. (Una función de tabla no puede ser una función con fuente).

Signaturas de función

Una función de esquema se identifica por su nombre de esquema, un nombre de función, el número de parámetros y los tipos de datos de sus parámetros. Una función de módulo se identifica por su nombre de esquema, su nombre de módulo, un nombre de función, el número de parámetros y los tipos de datos de sus parámetros. Esta identificación de una función de esquema o una función de módulo se denomina *signatura de función*, que debe ser exclusiva en la base de datos; por ejemplo, TEST.RISK(INTEGER). Varias funciones pueden tener el mismo nombre dentro de un esquema o un módulo, siempre que el número de parámetros o bien los tipos de datos de los parámetros sean diferentes. Un nombre de función para el que existen múltiples instancias de función con el mismo número de parámetros se llama función *sobrecargada*. Un nombre de función puede estar sobrecargado dentro de un esquema, en cuyo caso hay más de una función con el mismo nombre y con el mismo número de parámetros en el esquema. De igual forma, un nombre de función puede estar sobrecargado dentro de un módulo, en cuyo caso hay más de una función con el mismo nombre y con el mismo número de parámetros en el módulo. Estas funciones deben tener tipos de datos de parámetros distintos. Las funciones también pueden estar sobrecargadas dentro de los esquemas de una vía de acceso de SQL, en cuyo caso hay más de una función con el mismo nombre y con el mismo número de parámetros en los distintos esquemas de la vía de acceso de SQL. Estas funciones no deben tener necesariamente tipos de datos de parámetros distintos.

Invocación de funciones

Las referencias a una función deben ser conformes con la sintaxis siguiente:



Notas:

1 La palabra clave ALL o DISTINCT puede especificarse únicamente para una función agregada o para una función definida por el usuario cuyo origen es una función agregada.

En la sintaxis anterior, *expresión* y*expresión-fila* no pueden incluir una función agregada. Consulte "Expresiones" para conocer otras normas de *expresión*.

Una función se invoca haciendo referencia (en un contexto que lo permita) a su nombre calificado o no calificado, seguido por la lista de argumentos, entre paréntesis. Los calificadores posibles para un nombre de función son:

- Un nombre de esquema
- Uno nombre de módulo no calificado
- Un nombre de módulo calificado mediante esquema

El calificador que se utiliza al invocar una función determina el ámbito empleado para buscar una función coincidente.

- Si se utiliza un nombre de módulo calificado mediante esquema como calificador, el ámbito será el módulo especificado.
- Si se utiliza un único identificador como calificador, el ámbito incluye:
  - El esquema que coincide con el calificador
  - Uno de los módulos siguientes:
    - El módulo que realiza la invocación, si el nombre del módulo que realiza la invocación coincide con el calificador
    - El primer módulo de un esquema en la vía de acceso de SQL que coincide con el calificador
- Si no se emplea calificador, el ámbito incluye los esquemas de la vía de acceso de SQL y, si la función se invoca desde un objeto de módulo, el mismo módulo desde el que se invoca la función.

Para las sentencias de SQL estático, la vía de acceso de SQL se especifica mediante la utilización de la opción de vinculación **FUNCPATH**. Para las sentencias de SQL dinámico, la vía de acceso de SQL es el valor del registro especial CURRENT PATH.

Cuando se invoca una función, el gestor de bases de datos debe determinar la función que se debe ejecutar. Este proceso se denomina *resolución de funciones* y se aplica tanto a las funciones incorporadas como a las definidas por el usuario. Se recomienda que las invocaciones de funciones que pretendan invocar una función definida por el usuario sean totalmente calificadas. De esta forma, se mejora el rendimiento de la resolución de funciones y se impide que se produzcan resoluciones de funciones inesperadas al añadir funciones nuevas u otorgar privilegios.

Un *argumento* es un valor que se pasa a una función en una invocación. Cuando se invoca una función en SQL, se pasa una lista de cero o más argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. Un *parámetro* es una definición formal de una entrada para una función o una salida de una función. Cuando una función está definida en la base de datos, ya sea internamente (una función incorporada) o por el usuario (una función definida por el usuario), se especifican sus parámetros (cero o más) y el orden de sus definiciones define su posición y su semántica. Por lo tanto, cada parámetro es una entrada posicional particular para una función o una salida de una función. En la invocación, un argumento corresponde a un parámetro determinado en virtud de la posición que éste ocupe en la lista de argumentos. En los casos donde los argumentos de la invocación de la función no coinciden exactamente con los tipos de los parámetros de la función seleccionada, los argumentos se convierten al tipo de datos del parámetro durante la ejecución, utilizando las mismas normas que para la asignación a columnas. También se incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro.

El acceso a las funciones de esquema se controla mediante el privilegio EXECUTE sobre las funciones de esquema. Si el ID de autorización de la sentencia que invoca la función no tiene el privilegio EXECUTE, el algoritmo de resolución de función no tendrá en cuenta la función de esquema aunque ésta se corresponda mucho mejor. Las funciones incorporadas (funciones SYSIBM) y las funciones del esquema SYSFUN tienen otorgado el privilegio EXECUTE en PUBLIC de forma implícita.

El acceso a las funciones de módulo se controla mediante el privilegio EXECUTE sobre el módulo para todas las funciones pertenecientes al módulo. El ID de autorización de la sentencia que invoca la función puede no tener el privilegio EXECUTE sobre un módulo. En estos casos, y a diferencia de las funciones de esquema, el algoritmo de resolución de función continúa teniendo en cuenta las funciones de módulo de dicho módulo, aunque no pueden ejecutarse.

Cuando se invoca la sentencia definida por el usuario, el valor de cada uno de sus argumentos se asigna, utilizando la asignación de almacenamiento, al parámetro correspondiente de la función. Se pasa el control a las funciones externas de acuerdo con los convenios de llamada del lenguaje principal. Una vez finalizada la ejecución de una función escalar definida por el usuario o una función agregada definida por el usuario, el resultado de la función se asigna, utilizando la asignación de almacenamiento, al tipo de datos del resultado. Para obtener detalles sobre las normas de asignación, consulte “Asignaciones y comparaciones”.

Sólo se puede hacer referencia a las funciones de tabla en la cláusula FROM de una subselección. Para obtener más información sobre cómo hacer referencia a una función de tabla, consulte “referencia-tabla”.

Resolución de función

Tras invocar una función, el gestor de bases de datos debe determinar la función que se debe ejecutar. Este proceso se denomina resolución de funciones y se aplica tanto a las funciones incorporadas como a las definidas por el usuario.

La resolución de función consta de dos pasos.

1. En el primer paso, el gestor de bases de datos determina el conjunto de *funciones candidatas* basándose en la calificación del nombre de la función invocada, en el contexto que invoca la función, en el nombre no calificado de la función invocada y en el número de argumentos.

2. En el segundo paso, el gestor de bases de datos determina la mejor opción del conjunto de funciones candidatas basándose en los tipos de datos de los argumentos de la función invocada, en comparación con los tipos de datos de los parámetros de las funciones pertenecientes al conjunto de funciones candidatas.

Determinación del conjunto de funciones candidatas

Las funciones seleccionadas para el conjunto de funciones candidatas proceden de uno o varios de los espacios de búsqueda siguientes.

1. El *módulo de contexto*, es decir, el módulo que contiene el objeto de módulo que invocó la función
2. Un conjunto de uno o varios esquemas
3. Un módulo distinto del módulo de contexto

Los espacios de búsqueda específicos considerados se ven afectados por la calificación del nombre de la función invocada.

- *Resolución de función calificada*: cuando se invoca una función con un nombre de función y un calificador, el gestor de bases de datos utiliza el calificador y, en algunos casos, el contexto de la función invocada para determinar el conjunto de funciones candidatas.
  1. Si se invoca una función desde un objeto de módulo mediante un nombre de función con un calificador, el gestor de bases de datos considera si el calificador coincide con el nombre del módulo de contexto. Si el calificador es un solo identificador, el nombre de esquema del módulo se ignora al determinar una coincidencia. Si el calificador es un identificador de dos partes, se compara con el nombre de módulo calificado mediante esquema al determinar una coincidencia. Si el calificador coincide con el nombre de módulo de contexto, el gestor de bases de datos busca en el módulo de contexto las funciones candidatas, basándose en los criterios siguientes:
     – El nombre de la instancia de función debe coincidir con el nombre de la invocación de función.
     – El número de parámetros de la instancia de función debe coincidir con el número de argumentos de la invocación de función.

     Si se encuentran una o varias funciones candidatas en el módulo de contexto, se procesa este conjunto de funciones candidatas para determinar cuál es la mejor opción, sin tener en consideración las otras funciones candidatas posibles de otros espacios de búsqueda (véase "Determinación de la mejor opción"). En caso contrario, se continúa con el siguiente espacio de búsqueda.
  2. Si el calificador es un identificador único, el gestor de bases de datos considera el calificador como un nombre de esquema y busca en dicho esquema las funciones candidatas, basándose en los criterios siguientes:
     – El nombre de la instancia de función debe coincidir con el nombre de la invocación de función.
     – El número de parámetros de la instancia de función debe coincidir con el número de argumentos de la invocación de función.
     – El ID de autorización de la sentencia debe tener el privilegio EXECUTE para la instancia de la función.

     Si se encuentran una o varias funciones candidatas en el esquema, se procesa este conjunto de funciones candidatas para determinar cuál es la mejor opción, sin tener en consideración las otras funciones candidatas posibles de

otros espacios de búsqueda (véase "Determinación de la mejor opción"). En caso contrario, se continúa con el siguiente espacio de búsqueda, si procede.

3. Si se invoca la función desde fuera de un módulo o el calificador no coincide con el nombre del módulo de contexto cuando se le invoca desde un objeto de módulo, el gestor de bases de datos considera el calificador como un nombre de módulo. Dejando de lado el privilegio EXECUTE sobre los módulos, el gestor de bases de datos selecciona a continuación el primer módulo coincidente, según los criterios siguientes:
   - Si el nombre de módulo está calificado con un nombre de esquema, selecciona el módulo con dicho nombre de esquema y nombre de módulo.
   - Si el nombre del módulo no está calificado con un nombre de esquema, selecciona el módulo con dicho nombre de módulo que se encuentra en el primer esquema de la vía de acceso de SQL.
   - Si no se encuentra el módulo mediante la vía de acceso de SQL, selecciona el alias público de módulo con dicho nombre de módulo.

   Si no se encuentra un módulo coincidente, no existen funciones candidatas. Si se encuentra un módulo coincidente, el gestor de bases de datos busca en el módulo seleccionado las funciones candidatas, basándose en los criterios siguientes:
   - El nombre de la instancia de función debe coincidir con el nombre de la invocación de función.
   - El número de parámetros de la instancia de función debe coincidir con el número de argumentos de la invocación de función.
   - La función debe ser una función de módulo publicado.

   Si se encuentran una o varias funciones candidatas en los módulos seleccionados, se procesa este conjunto de funciones candidatas para determinar cuál es la mejor opción (véase "Determinación de la mejor opción").

Si el gestor de bases de datos no encuentra ninguna candidata, se devuelve un error (SQLSTATE 42884).

- *Resolución de función no calificada*: cuando se invoca una función sin un calificador, el gestor de bases de datos considera el contexto de la función invocada para determinar los conjuntos de funciones candidatas.
  1. Si se invoca una función con un nombre de función no calificado desde un objeto de módulo, el gestor de bases de datos busca en el módulo de contexto las funciones candidatas, basándose en los criterios siguientes:
     - El nombre de la instancia de función debe coincidir con el nombre de la invocación de función.
     - El número de parámetros de la instancia de función debe coincidir con el número de argumentos de la invocación de función.

     Si se encuentran una o varias funciones candidatas en el módulo de contexto, estas funciones candidatas se unen a las funciones candidatas de los esquemas de la vía de acceso de SQL (véase el punto siguiente).
  2. Si se invoca una función con un nombre de función no calificado, tanto desde dentro de un objeto de módulo como desde fuera de un módulo, el gestor de bases de datos busca la lista de esquemas en la vía de acceso de SQL para resolver la instancia de función que se debe ejecutar. Para cada esquema de la vía de acceso de SQL (consulte "Vía de acceso de SQL"), el gestor de bases de datos busca en el esquema las funciones candidatas según los criterios siguientes:

- El nombre de la instancia de función debe coincidir con el nombre de la invocación de función.
- El número de parámetros de la instancia de función debe coincidir con el número de argumentos de función en la invocación de función.
- El ID de autorización de la sentencia debe tener el privilegio EXECUTE para la instancia de la función.

Si se encuentran una o varias funciones candidatas en los esquemas de la vía de acceso de SQL, estas funciones candidatas se unen a las funciones candidatas del módulo de contexto (véase el punto anterior). Este conjunto de funciones candidatas se procesa para determinar cuál es la mejor opción (véase "Determinación de la mejor opción").

Si el gestor de bases de datos no encuentra ninguna función candidata, se devuelve un error (SQLSTATE 42884).

Determinación de la mejor opción

El conjunto de funciones candidatas puede contener tanto una función como varias funciones con el mismo nombre. En ambos casos, los tipos de datos de los parámetros de cada función perteneciente al conjunto de funciones candidatas se utilizan para determinar si la función cumple los requisitos de mejor opción.

El gestor de bases de datos determina cuál es la función, o el conjunto de funciones, que mejor se adapta a los requisitos de mejor opción para la invocación, comparando los tipos de datos de argumento y parámetro. Tenga en cuenta que al tipo de datos del resultado de la función o al tipo de función (agregada, escalar o de tabla) en cuestión no se le aplica esta determinación.

Cuando se determina si los tipos de datos de los parámetros son iguales que los argumentos:

- Los sinónimos de los tipos de datos coinciden. Por ejemplo, FLOAT y DOUBLE se consideran iguales.
- Los atributos de un tipo de datos como la longitud, la precisión, la escala y la página de códigos se ignoran. Por lo tanto, se considera que CHAR(8) y CHAR(35) son iguales, como lo son DECIMAL(11,2) y DECIMAL(4,3).

Se obtiene un subconjunto de funciones candidatas considerando sólo las funciones para las que el tipo de datos de cada argumento de la invocación de función coincide con el tipo de datos del parámetro correspondiente de la instancia de función o se puede promocionar a éste. La lista de prioridades de promoción de tipos de datos que aparece en "Promoción de tipos de datos" muestra los tipos de datos apropiados (en lo que concierne a la promoción) para cada tipo de datos en un orden de mejor a peor. Si este subconjunto no está vacío, el tipo más apropiado se determina mediante el proceso promocionable en este subconjunto de las funciones candidatas. Si este subconjunto está vacío, el tipo más apropiado se determina mediante el proceso convertible en el conjunto original de funciones candidatas.

Proceso promocionable

Este proceso determina el tipo más apropiado teniendo en cuenta solamente si los argumentos de la invocación de la función coinciden o pueden promocionarse al tipo de datos del parámetro correspondiente de la definición de función. Para el subconjunto de funciones candidatas, compare las listas de parámetros de izquierda a derecha mediante el proceso siguiente:

1. Compare el tipo de datos del argumento de la invocación de la función con el tipo de datos del parámetro correspondiente de cada definición de función candidata. (Como se ha indicado anteriormente, los sinónimos de los tipos de datos coinciden y los atributos de los tipos de datos se ignoran).
2. Para este argumento, si una función candidata tiene un tipo de datos que es más apropiado (en lo que concierne exclusivamente a la promoción) para la invocación de la función que los tipos de datos de otras funciones candidatas, esa función es la más apropiada. La lista de prioridades de promoción de tipos de datos que aparece en "Promoción de tipos de datos" muestra los tipos de datos apropiados (en lo que concierne a la promoción) para cada tipo de datos en un orden de mejor a peor.
3. Si el tipo de datos del parámetro para más de una función candidata (en lo que concierne exclusivamente a la promoción) es igual de apropiado, elimine las funciones candidatas que no sean igual de apropiadas para la invocación de la función. Repita este proceso para el siguiente argumento de la invocación de la función.

Si sólo queda una función candidata después de comparar todos los argumentos, esa función es la más apropiada. Si queda más de una función, todas las funciones candidatas que quedan son igual de apropiadas. En este caso, el gestor de bases de datos selecciona la función del módulo de contexto, si ésta continúa siendo una función candidata. En caso contrario, selecciona la función cuyo esquema se encuentre en primer lugar en la vía de acceso de SQL.

Si selecciona una función, su uso satisfactorio depende de si se va a invocar en un contexto en el que se permita el resultado devuelto. Por ejemplo, si la función devuelve una tabla donde no se permite una tabla, se devuelve un error.

Proceso convertible

Este proceso determina el candidato más apropiado teniendo en cuenta si los argumentos de la invocación de la función coinciden o pueden promocionarse al tipo de datos del parámetro correspondiente de la definición de función y si los argumentos de entrada pueden convertirse de forma implícita para la resolución de funciones en el tipo de datos del parámetro correspondiente. Para el conjunto de funciones candidatas, compare las listas de parámetros de izquierda a derecha mediante el proceso siguiente:

1. Compare el tipo de datos del argumento de la invocación de la función con el tipo de datos del parámetro correspondiente de cada definición de función candidata. (Como se ha indicado anteriormente, los sinónimos de los tipos de datos coinciden y los atributos de los tipos de datos se ignoran).
2. Para este argumento, si una función candidata tiene un tipo de datos que es más apropiado (en lo que concierne exclusivamente a la promoción) para la invocación de la función que los tipos de datos de otras funciones candidatas, esa función es la más apropiada. La lista de prioridades de promoción de tipos de datos que aparece en "Promoción de tipos de datos" muestra los tipos de datos apropiados (en lo que concierne a la promoción) para cada tipo de datos en un orden de mejor a peor. En este caso, cuando sólo quede una función candidata, los argumentos que quedan se evalúan para garantizar que se puede

promocionar o convertir cada argumento al parámetro correspondiente. Si no, se devuelve un error (SQLSTATE 42884).

3. Cuando el tipo de datos del parámetro para más de una función candidata (en lo que concierne exclusivamente a la promoción) es igual de apropiado, elimine las funciones candidatas que no sean igual de apropiadas para la invocación de la función. Si el tipo de datos del parámetro no es apropiado (en lo que concierne exclusivamente a la promoción) para la invocación de cualquier función candidata, no se elimina ninguna función candidata. Repita este proceso para el siguiente argumento de la invocación de la función.

Si queda más de una función candidata después de procesar los argumentos, compare cada parámetro que tenga un tipo de datos que no sea apropiado (teniendo en cuenta sólo la promoción) de izquierda a derecha mediante el proceso siguiente:

1. Compruebe los tipos de datos de los parámetros correspondientes para el resto de funciones candidatas. Si todos los tipos de datos no pertenecen a la misma lista de prioridades de tipos de datos, como se explica en "Promoción de tipos de datos", se devuelve un error (SQLSTATE 428F5).
2. Si el tipo de datos del argumento no puede convertirse de forma implícita en el tipo de datos del parámetro correspondiente, como se indica en Conversión implícita para la resolución de funciones, se devuelve un error (SQLSTATE 42884).
3. Compare el tipo de datos del argumento de la invocación de la función con el tipo de datos del parámetro de cada definición candidata. Si una función candidata tiene un tipo de datos que es más apropiado (en lo que concierne a la conversión implícita) para la invocación de la función que los tipos de datos de otras funciones candidatas, esa función es la más apropiada. La lista de tipos de datos que aparece en "Conversión implícita para la resolución de funciones" muestra el tipo de datos más apropiado (en lo que concierne a la conversión implícita). Elimine las funciones candidatas que no sean igual de apropiadas para la invocación de la función (en lo que concierne a la conversión implícita). Repita este proceso para el siguiente argumento de la invocación de la función que tenga un tipo de datos que no sea apropiado (teniendo en cuenta sólo la promoción).

Si sólo queda una función candidata después de comparar todos los argumentos, esa función es la más apropiada. Si queda más de una función, todas las funciones candidatas que quedan son igual de apropiadas. En este caso, el gestor de bases de datos selecciona la función del módulo de contexto, si ésta continúa siendo una función candidata. En caso contrario, selecciona la función cuyo esquema se encuentre en primer lugar en la vía de acceso de SQL.

Una vez seleccionada una función, todavía puede devolverse un error.

- Si se selecciona una función de módulo y la función se invoca desde fuera de un módulo o desde dentro de un objeto de módulo, y si el calificador no coincide con el nombre del módulo de contexto, el ID de autorización de la sentencia que invocó la función debe disponer del privilegio EXECUTE sobre el módulo que contiene la función seleccionada (SQLSTATE 42501).
- Si selecciona una función, su uso satisfactorio depende de si se va a invocar en un contexto en el que se permita el resultado devuelto. Por

ejemplo, si la función devuelve una tabla donde no se permite una tabla, se devuelve un error (SQLSTATE 42887).

- Si se necesitaría convertir (no promocionar) implícitamente una función de conversión, tanto incorporada como definida por el usuario, y cualquier argumento al tipo de datos del parámetro, se devuelve un error (SQLSTATE 42884).
- Si la invocación de una función implica un argumento con un tipo de fila sin nombre, se devuelve un error (SQLSTATE 42884) si se produce una de las siguientes condiciones:
  - El número de campos del tipo de fila sin nombre del argumento no coincide con el número de campos del tipo de fila sin nombre del parámetro
  - Los tipos de datos de los campos del argumento no pueden asignarse al tipo de datos correspondiente de los campos del parámetro

Conversión implícita para la resolución de funciones

La conversión implícita para la resolución de funciones no recibe soporte para argumentos con un tipo definido por el usuario, un tipo de referencia o un tipo de datos XML. Tampoco recibe soporte para funciones de conversión incorporadas o definidas por el usuario. Recibe soporte en los casos siguientes:

- Un valor de un tipo de datos puede convertirse en cualquier otro tipo de datos que esté en la misma lista de prioridades de tipos de datos, tal como se especifica en “Promoción de tipos de datos”.
- Un tipo de datos numérico o de fecha y hora puede convertirse en un tipo de datos de serie de caracteres o gráfica, excepto los LOB
- Un tipo de serie de caracteres o gráfica, excepto los LOB, puede convertirse en un tipo de datos numérico o de fecha y hora
- Un carácter FOR BIT DATA puede convertirse en un BLOB y un BLOB puede convertirse en un tipo FOR BIT DATA de caracteres
- Un tipo de datos TIMESTAMP puede convertirse en un tipo de datos TIME
- Un argumento sin tipo puede convertirse en un tipo de datos de serie, numérico o de indicación de fecha y hora.

Al igual que la lista de prioridades de tipos de datos en el caso de la promoción, para la conversión implícita existe un orden para los tipos de datos que están en el grupo de tipos de datos relacionados. Este orden se utiliza cuando se realiza una resolución de funciones que tiene en cuenta la conversión implícita. En la Tabla 20 en la página 211 se muestra el orden de los tipos de datos para la conversión implícita para la resolución de funciones. Los tipos de datos se listan en orden de mejor a peor (tenga en cuenta que es diferente del orden de la lista de prioridades de tipos de datos para promoción). Observe que, cuando la resolución de funciones selecciona una función incorporada y es necesaria una conversión implícita para algún argumento, si la función incorporada da soporte tanto a la entrada de caracteres como de gráficos para el parámetro, el argumento se convierte de forma implícita en un carácter.

Tabla 20. Orden de los tipos de datos para la conversión implícita para resolución de funciones

Grupo de tipos de datos	Lista de tipos de datos para la conversión implícita para resolución de funciones (en orden de mejor a peor)
Tipos de datos numéricos	DECFLOAT, doble, real, decimal, BIGINT, INTEGER, SMALLINT
Tipos de datos de serie de caracteres y gráfica	VARCHAR o VARGRAPHIC, CHAR o GRAPHIC, CLOB o DBCLOB
Tipos de datos de fecha y hora	TIMESTAMP, DATE

Notas:

1. Los tipos en minúsculas anteriores son los siguientes:
   - decimal = DECIMAL (*p*,*s*) o NUMERIC(*p*,*s*)
   - real = REAL o FLOAT(*n*) donde *n* no puede ser superior a 24
   - doble = DOUBLE, DOUBLE-PRECISION, FLOAT o FLOAT(*n*), donde *n* es mayor que 24

   Los sinónimos, más cortos o más lagos, de los tipos de datos listados se consideran iguales a la forma listada.
2. Sólo en el caso de una base de datos Unicode, los siguientes se consideran tipos de datos equivalentes:
   - CHAR o GRAPHIC
   - VARCHAR y VARGRAPHIC
   - CLOB y DBCLOB

Tabla 21. Longitud derivada de un argumento al invocar una función escalar incorporada en los casos en que se necesita la conversión implícita

	Tipo y longitud de destino								
Tipo de datos fuente	**Char**	**Graphic**	**Varchar**	**Vargraphic**	**Clob**	**DBclob**	**Blob**	**Ind. fecha/hora**	**Decfloat**
UNTYPED	127	127	254	254	32767	32767	32767	12	34
SMALLINT	6	6	6	6	-	-	-	-	-
INTEGER	11	11	11	11	-	-	-	-	-
BIGINT	20	20	20	20	-	-	-	-	-
DECIMAL(*p*,*s*)	2+*p*	2+*p*	2+*p*	2+*p*	-	-	-	-	-
REAL	24	24	24	24	-	-	-	-	-
DOUBLE	24	24	24	24	-	-	-	-	-
DECFLOAT	42	42	42	42	-	-	-	-	-
CHAR (*n*)	-	-	-	-	-	-	min(*n*,254)	12	34
VARCHAR (*n*)	min(*n*,254)	min(*n*,127)	-	-	-	-	min(*n*,32672)	12	34
CLOB (*n*)	min(*n*,254)	min(*n*,127)	min(*n*,32672)	min(*n*,16336)	-	-	-	-	-
GRAPHIC (*n*)	-	-	-	-	-	-	-	12	34
VARGRAPHIC (*n*)	min(*n*,254)	min(*n*,127)	-	-	-	-	-	12	34
DBCLOB (*n*)	min(*n*,254)	min(*n*,127)	min(*n*,32672)	min(*n*,16336)	-	-	-	-	-
BLOB (*n*)	min(*n*,254)	-	min(*n*,32672)	-	-	-	-	-	-
TIME	8	8	8	8	-	-	-	-	-
DATE	10	10	10	10	-	-	-	-	-
TIMESTAMP(*p*)	if *p*=0 then 19 else *p*+20	if *p*=0 then 19 else *p*+20	if *p*=0 then 19 else *p*+20	if *p*=0 then 19 else *p*+20	-	-	-	-	-

Consideraciones sobre las vías de acceso SQL para funciones incorporadas

Las funciones incorporadas residen en un esquema especial denominado SYSIBM. Hay funciones adicionales disponibles en los esquemas SYSFUN, SYSPROC y SYSIBMADM, así como en los módulos del esquema SYSIBMADM que, sin embargo, no se consideran funciones incorporadas porque se han desarrollado como funciones definidas por el usuario y carecen de consideraciones de proceso especiales. Los usuarios no pueden definir funciones adicionales en los esquemas SYSIBM, SYSFUN, SYSPROC o SYSIBMADM (ni en ningún otro esquema cuyo nombre empiece por las letras 'SYS', excepto SYSTOOLS).

Como ya se ha indicado, las funciones incorporadas participan en el proceso de resolución de las funciones exactamente como lo hacen las funciones definidas por el usuario. Una diferencia entre ambas, desde el punto de vista de la resolución de función, es que las funciones incorporadas siempre deben tenerse en cuenta durante la resolución de función. Por este motivo, si se omite SYSIBM de los resultados de vía de acceso se asume (para la resolución de las funciones y los tipos de datos) que SYSIBM es el primer esquema de la vía de acceso.

Por ejemplo, si la vía de acceso de SQL de un usuario está definida de la siguiente manera:

```
"SHAREFUN","SYSIBM","SYSFUN"
```

y hay una función LENGTH definida en el esquema SHAREFUN con el mismo número y los mismos tipos de argumentos que SYSIBM.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SHAREFUN.LENGTH. No obstante, si la vía de acceso de SQL del usuario está definida de la siguiente forma:

```
"SHAREFUN","SYSFUN"
```

y existe la misma función SHAREFUN.LENGTH, una referencia no calificada a LENGTH en la sentencia de SQL de este usuario hará que se seleccione SYSIBM.LENGTH, ya que SYSIBM aparece implícitamente antes en la vía de acceso.

Para minimizar los posibles problemas en este aspecto:

- No utilice nunca los nombres de funciones incorporadas para funciones definidas por el usuario.
- Si, por algún motivo, es necesario crear una función definida por el usuario con el mismo nombre que una función incorporada, asegúrese de calificar todas las referencias a la misma.

Nota: Algunas invocaciones de las funciones incorporadas no dan soporte a SYSIBM como calificador explícito y se resuelven directamente en la función incorporada sin considerar la vía de acceso de SQL. En la descripción de la función incorporada se tratan casos específicos.

Ejemplos de resolución de funciones

Los siguientes son ejemplo de resolución de funciones. (Observe que no se muestran todas las palabras clave necesarias.)

- Este es un ejemplo que ilustra las consideraciones sobre vías de acceso SQL en la resolución de funciones. Para este ejemplo, existen ocho funciones ACT, en tres esquemas distintos, registrados como:

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE) SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_7 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_8 ...
```

La referencia de función es la siguiente (donde I1 e I2 son columnas INTEGER y D es una columna DECIMAL):

```
SELECT ... ACT(I1, I2, D) ...
```

Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
"JULIUS","AUGUSTUS","CAESAR"
```

De acuerdo con el algoritmo...

- La función que tiene el nombre específico ACT_8 se elimina como candidata, ya que el esquema NERO no se ha incluido en la vía de acceso de SQL.
- La función con el nombre específico ACT_3 se elimina como candidato, porque tiene el número incorrecto de parámetros. ACT_1 y ACT_6 se eliminan porque, en ambos casos, el primer argumento no se puede promocionar al tipo de datos del primer parámetro.
- Como sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.
- Para el primer argumento, las funciones restantes, ACT_2, ACT_4 y ACT_5 y ACT_7 coinciden exactamente con el tipo de argumento. No se puede pasar por alto ninguna de las funciones; así pues, se debe examinar el argumento siguiente.
- Para este segundo argumento, ACT_2, ACT_5 y ACT_7 son coincidencias exactas pero ACT_4 no lo es, por lo cual se descarta. Se examina el siguiente argumento para determinar alguna diferencia entre ACT_2, ACT_5 y ACT_7.
- Para el tercer y último argumento, ni ACT_2 ni ACT_5 ni ACT_7 coinciden exactamente con el tipo de argumento. Aunque ACT_2 y ACT_5 son igualmente buenos, ACT_7 no es tan bueno como los otros dos ya que el tipo DOUBLE está más próximo a DECIMAL que DECFLOAT. ACT_7 se elimina..
- Quedan dos funciones, ACT_2 y ACT_5, con signaturas de parámetros idénticas. La criba final consiste en determinar el esquema de qué función aparece primero en la vía de acceso de SQL y, en base a ello, se elige ACT_5.

- A continuación se muestra un ejemplo de una situación en la que la resolución de funciones generará un error (SQLSTATE 428F5), ya que más de una función candidata es igual de apropiada, pero los parámetros correspondientes para uno de los argumentos no pertenecen a la misma lista de prioridades de tipos.

  En este ejemplo, sólo hay tres funciones en un solo esquema definido de la misma manera:

  ```
  CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5))SPECIFIC ACT_1 ...
  CREATE FUNCTION CAESAR.ACT (INT, INT, DATE)     SPECIFIC ACT_2 ...
  CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)   SPECIFIC ACT_3 ...
  ```

  La referencia de función es la siguiente (donde I1 e I2 son columnas INTEGER y VC es una columna VARCHAR):

  ```
  SELECT ... ACT(I1, I2, VC) ...
  ```

  Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

  ```
  "CAESAR"
  ```

  De acuerdo con el algoritmo...

  - Cada una de las funciones candidatas se evalúa para determinar si el tipo de datos de cada argumento de entrada de la invocación de la función coincide con el tipo de datos del parámetro correspondiente de la instancia de la función o se puede promocionar a éste:
    - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.
    - Para el segundo argumento, ACT_1 se elimina porque INTEGER no se promociona a VARCHAR.
    - Para el tercer argumento, tanto ACT_2 como ACT_3 se eliminan, porque VARCHAR no se puede promocionar a DATE o DOUBLE, por lo que no queda ninguna función candidata.
  - Puesto que el subconjunto de funciones candidatas anterior está vacío, las funciones candidatas se toman en consideración utilizando el proceso convertible:
    - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.
    - Para el segundo argumento, ACT_1 se elimina puesto que INTEGER no se promociona a VARCHAR. ACT_2 y ACT_3 son mejores candidatas.
    - Para el tercer argumento, el tipo de datos de los parámetros correspondientes de ACT_2 y ACT_3 no pertenecen a la misma lista de prioridades de tipos de datos, por lo que se ha devuelto un error (SQLSTATE 428F5).

- En este ejemplo se ilustra una situación en la que la resolución de funciones será satisfactoria si se utiliza el proceso convertible. En este ejemplo, sólo hay tres funciones en un solo esquema definido de la misma manera:

  ```
  CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5))SPECIFIC ACT_1 ...
  CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT)     SPECIFIC ACT_2 ...
  CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)   SPECIFIC ACT_3 ...
  ```

  La referencia de función es la siguiente (donde I1 e I2 son columnas INTEGER y VC es una columna VARCHAR):

  ```
  SELECT ... ACT(I1, I2, VC) ...
  ```

  Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

  ```
  "CAESAR"
  ```

  De acuerdo con el algoritmo...

  - Cada una de las funciones candidatas se evalúa para determinar si el tipo de datos de cada argumento de entrada de la invocación de la función coincide con el tipo de datos del parámetro correspondiente de la instancia de la función o se puede promocionar a éste:
    - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.
    - Para el segundo argumento, ACT_1 se elimina porque INTEGER no se promociona a VARCHAR.
    - Para el tercer argumento, tanto ACT_2 como ACT_3 se eliminan, porque VARCHAR no se puede promocionar a DECFLOAT o DOUBLE, por lo que no queda ninguna función candidata.
  - Puesto que el subconjunto de funciones candidatas anterior está vacío, las funciones candidatas se toman en consideración utilizando el proceso convertible:
    - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.

- Para el segundo argumento, ACT_1 se elimina puesto que INTEGER no se promociona a VARCHAR. ACT_2 y ACT_3 son mejores candidatas.
- Para el tercer argumento, tanto DECFLOAT como DOUBLE están en la misma lista de prioridades de tipos de datos y VARCHAR puede convertirse de forma implícita tanto en DECFLOAT como en DOUBLE. Puesto que DECFLOAT es más apropiada para la conversión implícita, ACT_2 es la candidata más apropiada.

• En este ejemplo se muestra cómo, durante la resolución de funciones mediante el proceso convertible, la promoción de los argumentos anteriores tiene prioridad sobre la conversión implícita. En este ejemplo, sólo hay tres funciones en un solo esquema definido de la misma manera:

```
CREATE FUNCTION CAESAR.ACT (INT, INT, VARCHAR(5))SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE)  SPECIFIC ACT_3 ...
```

La referencia de función es como se indica a continuación (donde I1 es una columna INTEGER y VC1 es una columna VARCHAR y C1 es una columna CHAR):

```
SELECT ... ACT(I1, VC1, C1) ...
```

Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
"CAESAR"
```

De acuerdo con el algoritmo:

– Cada una de las funciones candidatas se evalúa para determinar si el tipo de datos de cada argumento de entrada de la invocación de la función coincide con el tipo de datos del parámetro correspondiente de la instancia de la función o se puede promocionar a éste:
  - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.
  - Para el segundo argumento, todas las funciones candidatas se eliminan, porque VARCHAR no puede promocionarse a INTEGER, por lo que no queda ninguna función candidata.

– Puesto que el subconjunto de funciones candidatas anterior está vacío, las funciones candidatas se toman en consideración utilizando el proceso convertible.
  - Para el primer argumento, todas las funciones candidatas tienen una coincidencia exacta con el tipo de parámetro.
  - Para el segundo argumento, ninguna de las funciones candidatas tiene un parámetro al que el argumento correspondiente pueda promocionarse, por lo que no se elimina ninguna función candidata.
  - Puesto que el tercer argumento puede promocionarse al parámetro de ACT_1, pero no a los parámetros de ACT_2 o ACT_3, ACT_1 es la más apropiada.

Métodos

Un método de base de datos de un tipo estructurado es una relación entre un conjunto de valores de datos de entrada y un conjunto de valores resultantes, donde el primer valor de entrada (o *argumento sujeto*) tiene el mismo tipo, o es un subtipo del tipo sujeto (también llamado *parámetro sujeto*) del método. Por ejemplo, es posible pasar valores de datos de entrada de tipo VARCHAR a un método denominado CITY, de tipo ADDRESS y el resultado será un valor ADDRESS (o un subtipo de ADDRESS).

Los métodos se definen implícita o explícitamente, como parte de la definición de un tipo estructurado definido por el usuario.

Los métodos definidos implícitamente se crean para cada tipo estructurado. Se definen *métodos observadores* para cada atributo del tipo estructurado. Los métodos observadores permiten que una aplicación obtenga el valor de un atributo para una instancia del tipo. También se definen *métodos mutadores* para cada atributo, que permiten que una aplicación cambie la instancia de tipo modificando el valor de un atributo de una instancia de tipo. El método CITY descrito anteriormente es un ejemplo de método mutador para el tipo ADDRESS.

Los métodos definidos explícitamente o *métodos definidos por el usuario* son métodos que se registran en el catálogo SYSCAT.ROUTINES de una base de datos, utilizando una combinación de las sentencias CREATE TYPE (o ALTER TYPE ADD METHOD) y CREATE METHOD. Todos los métodos definidos para un tipo estructurado se definen en el mismo esquema que el tipo.

Los métodos definidos por el usuario para tipos estructurados amplían la función de sistema de bases de datos añadiendo definiciones de método (proporcionadas por usuarios o proveedores) que pueden aplicarse a instancias de tipo estructurado en el núcleo de la base de datos. La definición de los métodos de la base de datos permite que la base de datos explote los mismos métodos en su núcleo que los que utiliza una aplicación, proporcionando más sinergia entre la aplicación y la base de datos.

Métodos definidos por el usuario externos y SQL

Un método definido por el usuario puede ser externo o estar basado en una expresión SQL. Un método externo se define para la base de datos con una referencia a una biblioteca de códigos objeto y una función en dicha biblioteca que se ejecutará cuando se invoque el método. Un método basado en una expresión SQL devuelve el resultado de la expresión SQL cuando se invoca el método. Tales métodos no necesitan ninguna biblioteca de códigos objeto, ya que están escritos completamente en SQL.

Un método definido por el usuario puede devolver un solo valor cada vez que se invoca. Este valor puede ser un tipo estructurado. Un método se puede definir como *preservador del tipo* (utilizando SELF AS RESULT), para permitir que el tipo dinámico del argumento sujeto sea el tipo devuelto del método. Todos los métodos mutadores definidos implícitamente son preservadores del tipo.

Signaturas de método

Un método se identifica por su tipo sujeto, un nombre de método, el número de parámetros y los tipos de datos de sus parámetros. Esto se denomina una *signatura de método* y debe ser exclusiva en la base de datos.

Puede existir más de un método con el mismo nombre para un tipo estructurado, siempre que:

- El número de parámetros o los tipos de datos de los parámetros sean diferentes o
- Los métodos formen parte de la misma jerarquía de métodos (es decir, los métodos estén en una relación de alteración temporal o alteren temporalmente el mismo método original) o
- No exista la misma signatura de función (utilizando el tipo sujeto o cualquiera de sus subtipos o supertipos como primer parámetro).

Un nombre de método que tiene varias instancias de método se denomina *método sobrecargado*. Un nombre de método puede estar sobrecargado dentro de un tipo, lo que significa que existe más de un método de ese nombre para el tipo (todos los cuales tienen diferentes tipos de parámetros). Un nombre de método también puede estar sobrecargado en la jerarquía de tipos sujeto, en cuyo caso existe más de un método con ese nombre en la jerarquía de tipos. Estos métodos deben tener tipos de parámetros distintos.

Un método se puede invocar haciendo referencia (en un contexto permitido) al nombre de método, precedido por una referencia a una instancia de tipo estructurado (el argumento sujeto) y por el operador de doble punto. A continuación debe seguir una lista de argumentos entre paréntesis. El método que se invoca realmente depende del tipo estático del tipo sujeto, utilizando el proceso de resolución de métodos descrito en la sección siguiente. Los métodos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la invocación de funciones, en cuyo caso se aplican las normas normales para la resolución de la función.

Si la resolución de la función da como resultado un método definido con WITH FUNCTION ACCESS, se procesan todos los pasos siguientes de invocación de métodos.

El acceso a los métodos se controla mediante el privilegio EXECUTE. Se utilizan sentencias GRANT y REVOKE para especificar quién puede o no puede ejecutar un método o conjunto de métodos determinado. Se necesita el privilegio EXECUTE (o la autorización DATAACCESS) para invocar un método. La persona que define el método recibe el privilegio EXECUTE de forma automática. Si se trata de un método externo o un método SQL que tienen la opción WITH GRANT en todos los objetos subyacentes, la persona que lo define también recibe la opción WITH GRANT con el privilegio EXECUTE sobre el método. El definidor (o el ID que tenga autorización ACCESSCTRL o SECADM) debe otorgarse, pues, al usuario que desee invocar el método desde cualquier sentencia de SQL o hacer referencia al método en cualquier sentencia DDL (como CREATE VIEW, CREATE TRIGGER o al definir una restricción). Si no se otorga a un usuario el privilegio EXECUTE, el algoritmo de resolución de métodos no tendrá en cuenta el método aunque éste se corresponda mucho mejor.

Resolución de métodos

Después de invocar un método, el gestor de bases de datos debe decidir cuál de los posibles métodos con el mismo nombre es el "más apropiado ". Las funciones (incorporadas o definidas por el usuario) no se tienen en cuenta durante la resolución del método.

Un *argumento* es un valor que se pasa a un método en una invocación. Cuando un método se invoca en SQL, se le pasa el argumento sujeto (de algún tipo estructurado) y opcionalmente una lista de argumentos. Son argumentos posicionales en tanto que la semántica de dichos argumentos viene determinada por su posición en la lista de argumentos. Un *parámetro* es una definición formal de una entrada en un método. Cuando se define un método para la base de datos, ya sea implícitamente (método generado por el sistema para un tipo) o por un usuario (método definido por el usuario), se especifican sus parámetros (con el parámetro sujeto como primer parámetro) y el orden de sus definiciones determina sus posiciones y su semántica. Por tanto, cada parámetro es una entrada posicional determinada de un método. En la invocación, un argumento corresponde a un parámetro determinado en virtud a la posición que éste ocupe en la lista de argumentos.

El gestor de bases de datos utiliza el nombre de método proporcionado en la invocación, el privilegio EXECUTE sobre el método, el número y los tipos de datos de los argumentos, todos los métodos que tienen el mismo nombre para el tipo estático del argumento sujeto y los tipos de datos de sus parámetros correspondientes como base para decidir si selecciona o no un método. A continuación se muestran los resultados posibles del proceso de decisión:

- Un método determinado se considera que es el más apropiado. Por ejemplo, para los métodos denominados RISK del tipo SITE con signaturas definidas como:

  ```
  PROXIMITY(INTEGER) FOR SITE
  PROXIMITY(DOUBLE) FOR SITE
  ```

  la siguiente invocación de método (donde ST es una columna SITE, DB es una columna DOUBLE):

  ```
  SELECT ST..PROXIMITY(DB) ...
  ```

  se elegiría el segundo PROXIMITY.

  La siguiente invocación de método (donde SI es una columna SMALLINT):

  ```
  SELECT ST..PROXIMITY(SI) ...
  ```

  elegirá el primer PROXIMITY, ya que SMALLINT se puede promover a INTEGER y es una coincidencia mejor que DOUBLE, que se encuentra más abajo en la lista de prioridad.

  Cuando se tienen en cuenta argumentos que son tipos estructurados, la lista de prioridad incluye los supertipos del tipo estático del argumento. La función que mejor se ajusta es la definida con el parámetro de supertipo más cercano, en la jerarquía de tipos estructurados, al tipo estático del argumento de función.
- Ningún método se considera una opción aceptable. Tomando como ejemplo las dos mismas funciones del caso anterior y la siguiente referencia de función (donde C es una columna CHAR(5)):

  ```
  SELECT ST..PROXIMITY(C) ...
  ```

  el argumento es incoherente con el parámetro de las dos funciones PROXIMITY.

- Se selecciona un método determinado basándose en los métodos de la jerarquía de tipos y en el número y tipos de datos de los argumentos pasados en la invocación. Por ejemplo, para los métodos denominados RISK de los tipos SITE y DRILLSITE (un subtipo de SITE) con signaturas definidas como:

  ```
  RISK(INTEGER) FOR DRILLSITE
  RISK(DOUBLE) FOR SITE
  ```

  y la siguiente invocación de método (donde DRST es una columna DRILLSITE, DB es una columna DOUBLE):

  ```
  SELECT DRST..RISK(DB) ...
  ```

  se elegirá el segundo RISK, ya que DRILLSITE se puede promocionar a SITE.

  La siguiente referencia a método (donde SI es una columna SMALLINT):

  ```
  SELECT DRST..RISK(SI) ...
  ```

  elegirá el primer RISK, ya que SMALLINT se puede promocionar a INTEGER, que está más cerca en la lista de prioridad que DOUBLE, y DRILLSITE es una opción mejor que SITE, que es un supertipo.

  Los métodos con la misma jerarquía de tipos no pueden tener las mismas signaturas, teniendo en cuenta parámetros distintos al parámetro sujeto.

Determinación de la mejor opción

La comparación de los tipos de datos de los argumentos con los tipos de datos definidos de los parámetros de los métodos en cuestión, constituye la base primordial para decidir qué método de un grupo de métodos con el mismo nombre se considera el "más apropiado". Observe que los tipos de datos de los resultados de los métodos en cuestión no intervienen en esa decisión.

Para la resolución de métodos, si se considera al determinar la mejor opción que el tipo de datos de los argumentos de entrada se puede promocionar al tipo de datos del parámetro correspondiente. A diferencia de la resolución de función, si no se considera al determinar la mejor opción que los argumentos de entrada se pueden convertir de forma implícita al tipo de datos del parámetro correspondiente. Los módulos no se consideran durante la resolución de métodos porque los métodos no se pueden definir en módulos.

La resolución del método se realiza siguiendo los pasos siguientes:

1. En primer lugar, busque todos los métodos del catálogo (SYSCAT.ROUTINES) que cumplan las condiciones siguientes:
   - El nombre del método coincide con el nombre de invocación, y el parámetro sujeto es el mismo tipo o es un supertipo del tipo estático del argumento sujeto.
   - La persona que lo invoca tiene el privilegio EXECUTE sobre el método.
   - El número de parámetros definidos coincide con la invocación.
   - Cada argumento de invocación coincide con el tipo de datos del parámetro definido correspondiente del método o es "promocionable" a ese tipo.
2. A continuación, examine de izquierda a derecha cada argumento de la invocación del método. El argumento situado más a la izquierda (y por tanto el primer argumento) es el parámetro SELF implícito. Por ejemplo, un método definido para el tipo ADDRESS_T tiene un primer parámetro implícito de tipo ADDRESS_T. Para cada argumento, elimine todas las funciones que no sean la mejor coincidencia para ese argumento. La mejor opción para un argumento dado es el primer tipo de datos que aparece en la lista de prioridad

correspondiente al tipo de datos del argumento para el cual existe una función con un parámetro de ese tipo de datos. La longitud, la precisión, la escala y el atributo FOR BIT DATA no se tienen en cuenta en esta comparación. Por ejemplo, un argumento DECIMAL(9,1) se considera una coincidencia exacta para un parámetro DECIMAL(6,5) un argumento DECFLOAT(34) se considera una coincidencia exacta para un parámetro DECFLOAT(16) y un argumento VARCHAR(19) es una coincidencia exacta para un parámetro VARCHAR(6).

La mejor coincidencia para un argumento de tipo estructurado definido por el usuario es el propio argumento; la siguiente mejor coincidencia es el supertipo inmediato, y así sucesivamente para cada supertipo del argumento. Observe que sólo se tiene en cuenta el tipo estático (tipo declarado) del argumento de tipo estructurado, no el tipo dinámico (tipo más específico).

3. Como máximo, después del paso 2 queda un método elegible. Este es el método que se elige.
4. Si después del paso 2 no queda ningún método elegible, se produce un error (SQLSTATE 42884).

Ejemplo de resolución de método

A continuación se muestra un ejemplo de una resolución de método satisfactoria.

Existen siete métodos FOO para tres tipos estructurados definidos en una jerarquía de GOVERNOR como un subtipo de EMPEROR, como un subtipo de HEADOFSTATE, registrados con las signaturas siguientes:

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE)  FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE)   FOR EMPEROR     SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR EMPEROR     SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR     SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2))    FOR GOVERNOR    SPECIFIC FOO_7 ...
```

La referencia al método es la siguiente (donde I1 e I2 son columnas INTEGER, D es una columna DECIMAL y E es una columna EMPEROR):

```
SELECT E..FOO(I1, I2, D) ...
```

De acuerdo con el algoritmo...

- FOO_7 se elimina como candidato, porque el tipo GOVERNOR es un subtipo (no un supertipo) de EMPEROR.
- FOO_3 se elimina como candidato, porque tiene un número erróneo de parámetros.
- FOO_1 y FOO_6 se eliminan porque, en ambos casos, el primer argumento (no el argumento sujeto) no se puede promocionar al tipo de datos del primer parámetro. Como sigue habiendo más de un candidato, los argumentos se tienen en cuenta siguiendo un orden.
- Para el argumento sujeto, FOO_2 es un supertipo, mientras que FOO_4 y FOO_5 coinciden con el argumento sujeto.
- Para el primer argumento, los métodos restantes, FOO_4 y FOO_5, coinciden exactamente con el tipo del argumento. No se puede asignar ningún método y, por tanto, se debe examinar el argumento siguiente.
- Para este segundo argumento, FOO_5 es una coincidencia exacta pero FOO_4 no lo es, por lo cual se descarta. Esto deja FOO_5 como método elegido.

Invocación de métodos

Una vez seleccionado el método, pueden todavía existir algunos motivos por los cuales no se pueda utilizar el método.

Cada método está definido para devolver un resultado con un tipo de datos específico. Si este tipo de datos resultante no es compatible con el contexto donde se invoca el método, se produce un error. Por ejemplo, supongamos que se definen los siguientes métodos llamados STEP, cada uno con un tipo de datos diferentes como resultado:

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

y la siguiente referencia a método (donde S es una columna SMALLINT y TA es una columna TYPEA):

```
SELECT 3 + TA..STEP(S) ...
```

en este caso se elige el primer STEP, pues hay una coincidencia exacta del tipo del argumento. Se produce un error en la sentencia, porque el tipo resultante es CHAR(5) en lugar de un tipo numérico, tal como necesita un argumento del operador de suma.

Empezando por el método que se ha seleccionado, se utiliza el algoritmo descrito en "Asignación dinámica de métodos" para crear el conjunto de métodos asignables durante la compilación. En "Asignación dinámica de métodos" se describe con exactitud el método que se invoca.

Observe que cuando el método seleccionado es un método conservador del tipo:

- el tipo estático resultante tras la resolución de la función es el mismo que el tipo estático del argumento sujeto de la invocación del método
- el tipo dinámico resultante cuando se invoca el método es el mismo que el tipo dinámico del argumento sujeto de la invocación del método.

Esto puede ser un subtipo del tipo resultante especificado en la definición del método conservador del tipo, que a su vez puede ser un supertipo del tipo dinámico devuelto realmente cuando se procesa el método.

En los casos donde los argumentos de la invocación del método no coinciden exactamente con los tipos de datos de los parámetros del método seleccionado, los argumentos se convierten al tipo de datos del parámetro durante la ejecución, utilizando las mismas normas que para la asignación a columnas. Esto incluye el caso en el que la precisión, escala o longitud difiere entre el argumento y el parámetro, pero excluye el caso en el que el tipo dinámico del argumento es un subtipo del tipo estático del parámetro.

Asignación dinámica de métodos

Los métodos proporcionan la funcionalidad y encapsulan los datos de un tipo. Un método se define para un tipo y siempre puede asociarse con este tipo. Uno de los parámetros del método es el parámetro implícito SELF. El parámetro SELF es del tipo para el que se ha declarado el método. El argumento que se pasa al argumento SELF cuando se invoca el método en una sentencia DML se denomina *sujeto*.

Cuando se selecciona un método utilizando la resolución de métodos (vea "Resolución de métodos" en la página 218), o se ha especificado un método en una

sentencia de DDL, este método se conoce como el "método autorizado aplicable más específico". Si el sujeto es de tipo estructurado, es posible que el método tenga uno o varios métodos alternativos. Entonces, DB2 debe determinar a cuál de estos métodos debe invocar, en base al tipo dinámico (tipo más específico) del sujeto en tiempo de ejecución. Esta determinación se denomina "determinación del método asignable más específico". Este proceso se describe aquí.

1. En la jerarquía de métodos, busque el método original del que forme parte el método autorizado más específico. Se denomina el *método raíz*.
2. Cree el conjunto de métodos asignables, que debe incluir los siguientes:
   - El método autorizado aplicable más específico.
   - Cualquier método que altere temporalmente el método autorizado aplicable más específico y que esté definido para un tipo que sea un subtipo del sujeto de esta invocación.
3. Determine el método asignable más específico, de la forma siguiente:
   a. Empiece con un método arbitrario que sea un elemento del conjunto de métodos asignables y que sea un método del tipo dinámico del sujeto o de uno de sus supertipos. Es el método asignable inicial más específico.
   b. Itere por los elementos del conjunto de métodos asignables. Para cada método: Si el método está definido para uno de los subtipos adecuados del tipo para el que está definido el método asignable más específico y si está definido para uno de los supertipos del tipo más específico del sujeto, repita el paso 2 con este método como el método asignable más específico; de lo contrario, siga iterando.
4. Invoque el método asignable más específico.

Ejemplo:

Se proporcionan tres tipos: "Persona", "Empleado" y"Director". Existe un método original "ingresos", definido para "Persona", que calcula los ingresos de una persona. Por omisión, una persona es un desempleado (un niño, un jubilado, etc.). Por lo tanto, "ingresos" para el tipo "Persona" siempre devuelve cero. Para el tipo "Empleado" y para el tipo "Director", deben aplicarse algoritmos distintos para calcular los ingresos. Por lo tanto, el método "ingresos" para el tipo "Persona" se altera temporalmente en "Empleado" y en "Director".

Cree y rellene una tabla de la manera siguiente:

```
   CREATE
TABLE aTable (id integer, personColumn Person);
     INSERT INTO aTable VALUES (0, Persona()), (1, Empleado()), (2, Director());
```

Liste todas las personas que tengan unos ingresos mínimos de 40000€:

```
   SELECT id, persona, name
     FROM aTable
     WHERE persona..ingresos() >= 40000;
```

Utilizando la resolución de métodos, se selecciona el método "ingresos" para el tipo "Persona" como el método autorizado aplicable más específico.

1. El método raíz es "ingresos" para "Persona".
2. El segundo paso del algoritmo anterior se lleva a cabo para construir el conjunto de métodos asignables:
   - Se incluye el método "ingresos" para el tipo "Persona", porque es el método autorizado aplicable más específico.

- Se incluye el método ″ingresos″ para el tipo ″Empleado″ e ″ingresos″ para ″Director″, porque ambos métodos alteran temporalmente el método raíz y tanto ″Empleado″ como ″Director″ son subtipos de ″Persona″.

Por lo tanto, el conjunto de métodos asignables es: {″ingresos″ para ″Persona″, ″ingresos″ para ″Empleado″, ″ingresos″ para ″Director″}.

3. Determine el método asignable más específico:
   - Para un sujeto cuyo tipo más específico sea ″Persona″:
     a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.
     b. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto, ″ingresos″ para el tipo ″Persona″ es el método asignable más específico.
   - Para un sujeto cuyo tipo más específico sea ″Empleado″:
     a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.
     b. Itere por el conjunto de métodos asignables. Como el método ″ingresos″ para el tipo ″Empleado″ está definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto (Nota: Un tipo es su propio supertipo y subtipo)el método ″ingresos″ para el tipo ″Empleado″ es una opción mejor para el método asignable más específico. Repita este paso con el método ″ingresos″ para el tipo ″Empleado″ como el método asignable más específico.
     c. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Empleado″ y para un supertipo del tipo más específico del sujeto, el método ″ingresos″ para el tipo ″Empleado″ es el método asignable más específico.
   - Para un sujeto cuyo tipo más específico sea ″Director″:
     a. Deje que el método asignable inicial más específico sea ″ingresos″ para el tipo ″Persona″.
     b. Itere por el conjunto de métodos asignables. Como el método ″ingresos″ para el tipo ″Director″ está definido para un subtipo adecuado de ″Persona″ y para un supertipo del tipo más específico del sujeto (Nota: Un tipo es su propio supertipo y subtipo), el método ″ingresos″ para el tipo ″Director″ es una opción mejor para el método asignable más específico. Repita este paso con el método ″ingresos″ para el tipo ″Director″ como el método asignable más específico.
     c. Como no hay ningún otro método en el conjunto de métodos asignables que esté definido para un subtipo adecuado de ″Director″ y para un supertipo del tipo más específico del sujeto, el método ″ingresos″ para el tipo ″Director″ es el método asignable más específico.
4. Invoque el método asignable más específico.

Semántica de vinculación conservadora

La resolución de los objetos se lleva a cabo al definir un objeto SQL o al procesar una operación de vinculación de paquetes.

El gestor de bases de datos elige qué objeto SQL definido concreto se utilizará para un objeto SQL al que se hace referencia en una sentencia DDL o que está codificado en una aplicación.

Posteriormente el gestor de bases de datos podría dar como resolución un objeto SQL distinto, incluso aunque el objeto SQL original no haya cambiado en modo alguno. Esta resolución como otro objeto SQL es el resultado de definir otro objeto SQL (o de añadir un privilegio a una función existente) que el algoritmo de resolución de objetos define como resuelto antes del objeto SQL elegido inicialmente. A continuación se citan algunos ejemplos de objetos SQL y casos en los que se da esta resolución como otro objeto SQL:

- Rutinas: se puede definir una rutina nueva que sea más adecuada o que sea igual de adecuada pero que aparezca antes en la vía de acceso de SQL, o bien se podría otorgar un privilegio a una rutina existente que sea más adecuado o que sea igual de adecuado pero que aparezca antes en la vía de acceso de SQL.
- Tipos de datos definidos por el usuario: se puede definir un nuevo tipo de datos definido por el usuario con el mismo nombre y en el mismo esquema que aparezca antes en la vía de acceso de SQL.
- Variables globales: se puede definir una variable global nueva con el mismo nombre y en el mismo esquema que aparezca antes en la vía de acceso de SQL.
- Tablas o vistas que se resuelven con un alias público: se puede definir un alias privado, de tabla o de vista real con el mismo nombre en el esquema actual.
- Secuencias que se resuelven con un alias de secuencia público: se puede definir un alias de secuencia real o de secuencia privada con el mismo nombre en el esquema actual.
- Módulos que se resuelven como un alias de módulo público: se puede definir un alias de módulo real o de módulo privado con el mismo nombre en un esquema que esté en la vía de acceso de SQL.

Hay casos en los que el gestor de bases de datos debe poder repetir la resolución de objetos SQL tal como se resolvieron originalmente cuando se procesó la sentencia. Esto se da cuando se utilizan los objetos estáticos siguientes:

- Sentencias en paquetes de DML estático
- Vistas
- Desencadenantes
- Restricciones de comprobación
- Rutinas de SQL
- Las variables globales con un tipo definido por el usuario o una expresión por omisión
- Las rutinas con un tipo de parámetro definido por el usuario o una expresión por omisión

Para las sentencias en paquetes de DML estático, las referencias a objetos SQL se resuelven durante una operación de vinculación. Las referencias del objeto SQL en vistas, activadores, rutinas de SQL y restricciones de comprobación se resuelven cuando el objeto SQL se define o se vuelve a validar. Si se utiliza un objeto estático

que ya existe, se aplica la *semántica de vinculación conservadora* a menos que el objeto se haya marcado como no válido o no operativo por un cambio en el esquema de base de datos.

La semántica de vinculación conservadora garantiza que las referencias del objeto SQL se resuelven utilizando la misma vía de acceso de SQL, el esquema por omisión y el mismo conjunto de rutinas con los que se resolvió anteriormente. También garantiza que la indicación de fecha y hora de la definición de los objetos SQL que se tienen en cuenta durante la resolución de vinculación conservadora no sea posterior a la indicación de fecha y hora correspondiente al momento en que la sentencia se vinculó o se validó por última vez mediante la semántica de *vinculación no conservadora*. La semántica de vinculación no conservadora utiliza la misma vía de acceso de SQL y el esquema por omisión que la generación original del paquete o de la sentencia, pero no tiene en cuenta la indicación de fecha y hora de la definición de los objetos SQL ni los conjuntos de rutinas resueltos con anterioridad.

Algunos cambios en el esquema de base de datos, como descartar objetos, alterar objetos o revocar privilegios, pueden afectar a un objeto SQL de modo que el gestor de bases de datos ya no pueda resolver todos los objetos SQL dependientes de un objeto SQL existente mediante la semántica de vinculación conservadora.

- Cuando esto sucede para una sentencia estática en un paquete de SQL, el paquete se marca como inoperativo. La siguiente utilización de una sentencia de este paquete provocará una revinculación implícita del paquete utilizando la semántica de vinculación no conservadora para que sea posible resolver los objetos SQL teniendo en cuenta los cambios más recientes en el esquema de base de datos que ha provocado que el paquete sea inoperativo.
- Cuando esto sucede con una vista, un activador, una restricción de comprobación o una rutina de SQL, el objeto SQL se marca como no válido. La siguiente utilización del objeto provoca una revalidación implícita del objeto SQL con la semántica de vinculación no conservadora.

Piense en una base de datos con dos funciones que tienen las signaturas SCHEMA1.BAR(INTEGER) y SCHEMA2.BAR(DOUBLE). Supongamos que una vía de acceso de SQL contiene los dos esquemas, SCHEMA1 y SCHEMA2 (aunque su orden en la vía de acceso de SQL carece de importancia). A USER1 se le ha otorgado el privilegio EXECUTE sobre la función SCHEMA2.BAR(DOUBLE). Supongamos que USER1 crea una vista que invoca a BAR(INT_VAL), siendo INT_VAL una columna o una variable global con el tipo de datos INTEGER. Esta referencia de función en la vista se resuelve como la función SCHEMA2.BAR(DOUBLE) porque USER1 no tiene el privilegio EXECUTE sobre SCHEMA1.BAR(INTEGER). Si a USER1 se le otorga el privilegio EXECUTE sobre SCHEMA1.BAR(INTEGER) después de que se haya creado la vista, ésta continuará utilizando SCHEMA2.BAR(DOUBLE)a menos que un cambio en el esquema de base de datos haga que la vista se marque como no válida. La vista se marca como no válida si se revoca un privilegio necesario o si un objeto del que depende se descarta o se altera.

En el caso de los paquetes de DML estático, los paquetes se pueden volver a vincular implícitamente o emitiendo explícitamente el mandato REBIND (o la API correspondiente) o el mandato BIND (o la API correspondiente). La revinculación implícita se realiza con la semántica de vinculación conservadora si el paquete se marca como no válido, pero utiliza la semántica de vinculación no conservadora si el paquete se marca como inoperativo. Un paquete se marca como no válido solamente si se descarta o se altera un índice del que depende. El mandato

Semántica de vinculación conservadora

REBIND proporciona la posibilidad de resolver con la semántica de vinculación conservadora (RESOLVE CONSERVATIVE) o resolver teniendo en cuenta las rutinas, los tipos de datos y las variables globales nuevos (RESOLVE ANY, la opción por omisión). La opción RESOLVE CONSERVATIVE se puede utilizar solamente si el gestor de bases de datos no ha marcado el paquete como inoperativo (SQLSTATE 51028).

La descripción de la semántica de vinculación conservadora de este tema presupone que el parámetro de configuración de base de datos **auto_reval** tiene un valor distinto de DISABLED. El valor por omisión para las bases de datos nuevas es DEFERRED; el valor por omisión para las bases de datos actualizadas a la versión 9.7 es DISABLED. Si el valor de **auto_reval** es DISABLED, el comportamiento de la semántica de vinculación conservadora, la invalidación y la revalidación es el mismo que el de los releases anteriores a la versión 9.7. Con este valor, la semántica de vinculación conservadora solamente tiene en cuenta la indicación de fecha y hora de la definición de los objetos SQL para las funciones, los métodos, los tipos definidos por el usuario y las variables globales. En lo que respecta al comportamiento de la invalidación y la revalidación, esto significa, en el caso de las sentencias DROP, REVOKE y ALTER, que la semántica es más restrictiva o que el impacto sobre los objetos dependientes es disponer en cascada y descartar el objeto. En el caso de los paquetes, la mayoría de los cambios del esquema de base de datos dan como resultado que el paquete se marque como no válido y se utilice la semántica de vinculación conservadora durante la revinculación implícita. Sin embargo, cuando el esquema se cambia como consecuencia de descartar una función dependiente y **auto_reval** se establece en DISABLED, el paquete dependiente de la función se marca como inoperativo y no hay revinculación implícita de dicho paquete.

Expresiones

Una expresión especifica un valor. Puede ser un valor simple, formado sólo por una constante o un nombre de columna, o puede ser más complejo. Si se utilizan repetidamente expresiones complejas similares, puede plantearse la utilización de una función SQL para encapsular una expresión común.

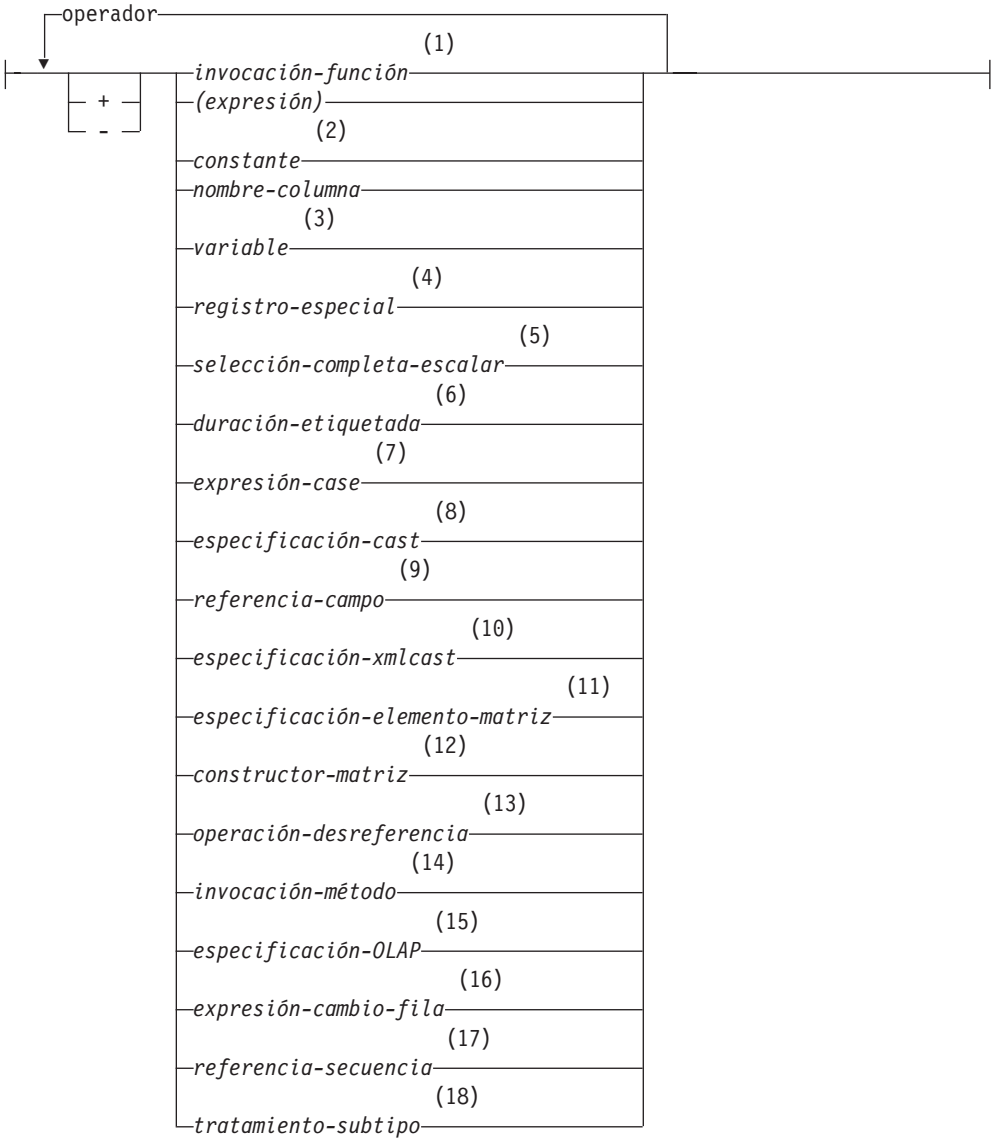
Autorización

Para utilizar algunas expresiones, como subselección-escalar, referencia-secuencia o invocación-función, puede ser necesario disponer de la autorización adecuada. Para estas expresiones, los privilegios del ID de autorización de la sentencia deben incluir las autorizaciones siguientes:

- subselección-escalar. Para obtener información acerca de las consideraciones sobre la autorización, consulte la sección sobre consultas.
- referencia-secuencia. Autorización para hacer referencia a la secuencia. Para obtener información acerca de las consideraciones sobre la autorización, consulte la sección sobre autorizaciones de secuencia.
- invocación-función. Autorización para ejecutar una función definida por el usuario. Para obtener información acerca de las consideraciones sobre la autorización, consulte la sección sobre invocación de funciones.
- variable. Si la variable es una variable global, se exige una autorización para hacer referencia a la variable global. Para obtener información, consulte Variables globales.

En una base de datos Unicode, una expresión que acepte una serie de caracteres o gráfica aceptará todo tipo de serie para el que se soporte la conversión.

expresión:



operador:

Notas:

1 Vea “Invocación de funciones” en la página 203 para obtener más información.

2 Vea “Constantes” en la página 155 para obtener más información.

3 Vea “Referencias a variables” en la página 76 para obtener más información.

4 Vea “Registros especiales” en la página 160 para obtener más información.

5 Vea “Selección completa escalar” en la página 238 para obtener más información.

6 Vea “Duraciones” en la página 239 para obtener más información.

7 Vea “Expresión CASE” en la página 245 para obtener más información.

8 Vea “Especificación CAST” en la página 248 para obtener más información.

9 Vea “Referencia a campo” en la página 254 para obtener más información.

10 Vea “Especificación XMLCAST” en la página 255 para obtener más información.

11 Vea “Especificación del elemento ARRAY” en la página 257 para obtener más información.

12 Vea “Constructor de matrices” en la página 258 para obtener más información.

13 Vea “Operación de desreferencia” en la página 260 para obtener más información.

14 Vea “Invocación de métodos” en la página 262 para obtener más información.

15 Vea “Especificaciones OLAP” en la página 264 para obtener más información.

16 Vea “Expresión ROW CHANGE” en la página 273 para obtener más información.

17 Vea “Referencia de secuencia” en la página 275 para obtener más información.

18 Vea “Tratamiento de los subtipos” en la página 279 para obtener más información.

19 || puede utilizarse como sinónimo de CONCAT.

Expresiones sin operadores

Si no se utilizan operadores, el resultado de la expresión es el valor especificado.

Ejemplos:

```
SALARY:SALARY'SALARY'MAX(SALARY)
```

Expresiones con el operador de concatenación

El operador de concatenación (CONCAT) combina dos operandos para formar una *expresión de serie*.

El primer operando es una expresión que devuelve un valor de cualquier tipo de datos de serie, cualquier tipo de datos numérico o cualquier tipo de datos de indicación de fecha y hora. El segundo operando también es una expresión que devuelve un valor de cualquier tipo de datos de serie, cualquier tipo de datos numérico o cualquier tipo de datos de indicación de fecha y hora. Algunos tipos de datos, sin embargo, no están soportados cuando se combinan con el tipo de datos del primer operando, como se describe a continuación.

Los operandos pueden ser cualquier combinación de valores de serie (excepto series binarias), numéricos y de fecha y hora. Cuando cualquier operando no es un valor de serie, se convierte de forma implícita en VARCHAR. Una serie binaria sólo se puede concatenar con otra serie binaria. No obstante, a través del proceso

convertible de resolución de función, una serie binaria se puede concatenar con una serie de caracteres definida como FOR BIT DATA cuando el primer operando es la serie binaria.

Las concatenaciones que implican operandos de serie de caracteres y operandos de serie gráfica sólo reciben soporte en una base de datos Unicode. Los operandos de caracteres primero se convierten en el tipo de datos gráficos antes de la concatenación. Las series de caracteres definidas como FOR BIT DATA no se pueden convertir en el tipo de datos gráfico.

Si ambos operandos pueden ser nulos, el resultado también podrá ser nulo; si alguno de ellos es nulo, el resultado es el valor nulo. De lo contrario, el resultado constará de la serie del primer operando seguida por la del segundo. La comprobación se efectúa para detectar los datos mixtos formados defectuosamente al realizar la concatenación.

La longitud del resultado es la suma de las longitudes de los operandos.

El tipo de datos y el atributo de longitud del resultado vienen determinados por los de los operandos, tal como se muestra en la tabla siguiente:

Tabla 22. Tipo de datos y longitud de los operandos concatenados

Operandos	**Atributos de longitud combinados**	**Resultado**
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC

Tabla 22. Tipo de datos y longitud de los operandos concatenados (continuación)

Operandos	Atributos de longitud combinados	Resultado
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

Observe que, por compatibilidad con versiones anteriores, no hay escalamiento automático de los resultados que implican tipos de datos LONG VARCHAR o LONG VARGRAPHIC a tipos de datos LOB. Por ejemplo, la concatenación de un valor CHAR(200) y un valor LONG VARCHAR totalmente completo da como resultado un error en lugar de una promoción de un tipo de datos CLOB.

La página de códigos del resultado se considera una página de códigos derivada que viene determinada por la página de códigos de sus operandos.

Un operando puede ser un marcador de parámetros. Si se utiliza un marcador de parámetro, el tipo de datos y los atributos de longitud del operando se consideran los mismos que los del operando que no es el marcador de parámetro. El orden de las operaciones tiene su importancia, puesto que determina estos atributos en casos en los que se produce una concatenación anidada.

Ejemplo 1: Si FIRSTNME es Pierre y LASTNAME es Fermat, entonces lo siguiente:

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

devuelve el valor `Pierre Fermat`

Ejemplo 2: Dado:

- `COLA` definido como VARCHAR(5) con valor `'AA'`
- `:host_var` definida como una variable del lenguaje principal con una longitud 5 y el valor `'BB  '`
- `COLC` definido como CHAR(5) con valor `'CC'`
- `COLD` definido como CHAR(5) con valor `'DDDDD'`

El valor de `COLA CONCAT :host_var CONCAT COLC CONCAT COLD` es
`'AABB  CC  DDDDD'`

El tipo de datos es VARCHAR, el atributo de longitud es 17 y la página de códigos del resultado es la página de códigos de la sección. Para obtener más información sobre las páginas de código de las secciones, consulte ″Derivación de valores de página de códigos″.

Ejemplo 3: Dado:

- `COLA` definido como CHAR(10)
- `COLB` definido como VARCHAR(5)

El marcador de parámetros de la expresión:

```
COLA CONCAT COLB CONCAT ?
```

se considera VARCHAR(15), porque COLA CONCAT COLB se evalúa primero, dando como resultado el primer operando de la segunda operación CONCAT.

Tipos definidos por el usuario

No se puede utilizar un tipo definido por el usuario con el operador de concatenación, aunque sea un tipo diferenciado con un tipo de datos fuente de tipo serie. Para poder concatenar, es preciso crear una función con el operador CONCAT como fuente. Por ejemplo, si existieran los tipos diferenciados TITLE y TITLE_DESCRIPTION y ambos tuvieran los tipos de datos VARCHAR(25), la siguiente función definida por el usuario, ATTACH, se podría utilizar para concatenarlos.

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

También existe la posibilidad de sobrecargar el operador de concatenación empleando una función definida por el usuario para añadir los tipos de datos nuevos.

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

Expresiones con operadores aritméticos

Si se utilizan operadores aritméticos, el resultado de la expresión es un valor derivado de la aplicación de los operadores a los valores de los operandos.

Si cualquier operando puede ser nulo o la base de datos se ha configurado con **dft_sqlmathwarn** establecido en yes, el resultado puede ser nulo.

Si algún operando tiene el valor nulo, el resultado de la expresión es el valor nulo.

Los operadores numéricos se pueden aplicar a tipos numéricos con signo y a tipos de fecha y hora (vea “Aritmética de fecha y hora en SQL” en la página 240). Por ejemplo, USER+2 no es válido. Las funciones con fuente se pueden definir para operaciones aritméticas sobre tipos diferenciados con un tipo fuente que sea un tipo numérico con signo.

El operador de prefijo + (más unario) no modifica su operando. El operador de prefijo - (menos unario) invierte el signo de un operando distinto de cero de coma flotante no decimal. El operador de prefijo - (menos unario) invierte el signo de todos los operandos de coma flotante decimal, incluyendo el cero y los valores especiales, es decir los Signalling y Non-signalling NAN y más y menos infinito. Si el tipo de datos de A es un entero pequeño, el tipo de datos de -A será un entero grande. El primer carácter del símbolo que sigue a un operador de prefijo no debe ser un signo más ni un signo menos.

Los *operadores infijos* +, -, * y / especifican, respectivamente, una suma, una resta, una multiplicación y una división. El valor del segundo operando de una división no debe ser cero, a no ser que el cálculo se realice utilizando la aritmética de coma flotante decimal. Estos operadores también se pueden tratar como funciones. Por consiguiente, la expresión *"+"(a,b)* es equivalente a la función de “operador” cuya expresión es *a+b*.

Los operandos con un tipo de datos de serie gráfica o de caracteres, salvo los LOB, se convierten a DECFLOAT(34) con las normas para la especificación CAST, antes de que se realice la operación aritmética. Para obtener más información, consulte la

sección sobre conversión entre tipos de datos. Tenga en cuenta que la aritmética relacionada con operandos de serie gráfica sólo se soporta en las bases de datos Unicode.

Los operandos con un tipo de datos de serie se convierten a DECFLOAT(34) con las normas para la especificación CAST, antes de que se realice la operación aritmética. Para obtener más información, consulte la sección sobre conversión entre tipos de datos. La serie debe contener una representación válida de un número.

Errores aritméticos

Si se produce un error aritmético como, por ejemplo, una división por cero o se produce un desbordamiento numérico durante el proceso de una expresión de coma flotante no decimal, se devuelve un error (SQLSTATE 22003 ó 22012). Para expresiones de coma flotante decimal, se devuelve un aviso (SQLSTATEs 0168C, 0168D, 0168E o 0168F) que depende de la naturaleza de la condición aritmética.

Una base de datos puede configurarse (con **dft_sqlmathwarn** establecido en yes) de modo que los errores aritméticos devuelvan un valor nulo para la expresión de coma flotante no decimal, la consulta devuelva un aviso (SQLSTATE 01519 ó 01564) y continúe con el proceso de la sentencia de SQL.

Para las expresiones de coma flotante decimal, **dft_sqlmathwarn** no tiene ningún efecto; las condiciones aritméticas devuelven un valor adecuado (posiblemente un valor especial de coma flotante decimal), la consulta devuelve un aviso (SQLSTATE 0168C, 0168D, 0168E o 0168F), y continúa con el proceso de la sentencia de SQL. Los valores especiales devueltos incluyen infinito más y menos y no un número. Las expresiones aritméticas que implican uno o más números de coma flotante decimal nunca se evalúan en valor nulo a menos que uno o más de los argumentos de la expresión sean nulos.

Cuando los errores aritméticos se tratan como nulos, hay implicaciones en los resultados de las sentencias de SQL. A continuación encontrará algunos ejemplos de dichas implicaciones.

- Un error aritmético que se produce en la expresión que es el argumento de una función agregada provoca que se ignore la fila en la determinación del resultado de la función agregada. Si el error aritmético ha sido un desbordamiento, puede afectar de manera significativa a los valores del resultado.
- Un error aritmético que se produce en la expresión de un predicado en una cláusula WHERE puede hacer que no se incluyan filas en el resultado.
- Un error aritmético que se produce en la expresión de un predicado en una restricción de comprobación da como resultado el proceso de actualización o inserción ya que la restricción no es falsa.

Si estos tipos de efectos no son aceptables, deben seguirse pasos adicionales para manejar el error aritmético y producir resultados aceptables. Algunos ejemplos son:

- añadir una expresión case para comprobar la división por cero y establecer el valor deseado para dicha situación
- añadir predicados adicionales para manejar los nulos (por ejemplo, una restricción de comprobación en columnas sin posibilidad de nulos daría:

  ```
  check (c1*c2 is not null and c1*c2>5000)
  ```

  para hacer que la restricción se violase en un desbordamiento).

Dos operandos enteros

Si ambos operandos de un operador aritmético son enteros, la operación se realiza en binario y el resultado es un *entero grande* a no ser que uno de los operandos (o ambos) sea un entero superior, en cuyo caso el resultado es un entero superior. Se pierde cualquier resto de una división. El resultado de una operación aritmética de enteros (incluyendo el menos unario) debe estar dentro del rango del tipo del resultado.

Operandos enteros y decimales

Si un operando es un entero y el otro es un decimal, la operación se realiza en decimal utilizando una copia temporal del entero que se habrá convertido a número decimal con la precisión *p* y la escala 0; *p* es 19 para un entero superior, 11 para un entero grande y 5 para un entero pequeño.

Dos operandos decimales

Si los dos operandos son decimales, la operación se efectúa en decimal. El resultado de cualquier operación aritmética decimal es un número decimal con una precisión y una escala que dependen de la operación y de la precisión y la escala de los operandos. Si la operación es una suma o una resta y los operandos no tienen la misma escala, la operación se efectúa con una copia temporal de uno de los operandos. La copia del operando más corto se extiende con ceros de cola de manera que la parte de la fracción tenga el mismo número de dígitos que el otro operando.

El resultado de una operación decimal no debe tener una precisión mayor que 31. El resultado de una suma, resta y multiplicación decimal se obtiene de un resultado temporal que puede tener una precisión mayor que 31. Si la precisión del resultado temporal no es mayor que 31, el resultado final es el mismo que el resultado temporal.

Aritmética decimal en SQL

Las fórmulas siguientes definen la precisión y la escala del resultado de las operaciones decimales en SQL. Los símbolos *p* y *s* indican la precisión y la escala del primer operando y los símbolos *p'* y *s'* indican y la precisión y la escala del segundo operando.

Sumas y restas

La precisión es min(31,max(*p*-*s*,*p'*-*s'*) +max(*s*,*s'*)+1). La escala del resultado de una suma o una resta es max (*s*,*s'*).

Multiplicaciones

La precisión del resultado de una multiplicación es min (31,*p*+*p'*) y la escala es min(31,*s*+*s'*).

Divisiones

La precisión del resultado de la división es 31. La escala es *31*-*p*+*s*-*s'*. La escala no debe ser negativa.

Nota: El parámetro de configuración de base de datos **min_dec_div_3** modifica la escala para las operaciones aritméticas decimales que incluyen la división. Si el valor del parámetro se establece en NO, la escala se calcula como *31-p+s-s'*. Si el parámetro se establece en YES, la escala se calcula como MAX(3, *31-p+ s-s'*). Esto asegura que el resultado de una división decimal tenga siempre una escala de 3 como mínimo (la precisión es siempre 31).

Operandos de coma flotante

Si alguno de los operandos de un operador aritmético es de coma flotante, pero no de coma flotante decimal, la operación se realiza en coma flotante. Los operandos se convierten primero a números de coma flotante de doble precisión, si es necesario. Por lo tanto, si cualquier elemento de una expresión es un número de coma flotante, el resultado de la expresión es un número de coma flotante de precisión doble.

Una operación en la que intervenga un número de coma flotante y un entero se realiza con una copia temporal del entero que se ha convertido a coma flotante de precisión doble. Una operación en la que intervenga un número de coma flotante y un número decimal se efectúa con una copia temporal del número decimal que se ha convertido a coma flotante de precisión doble. El resultado de una operación de coma flotante debe estar dentro del rango de los números de coma flotante.

El orden en el que se procesan los operandos de coma flotante (o argumentos en funciones) pueden afectar ligeramente los resultados porque los operandos de coma flotante son representaciones aproximadas de números reales. Dado que es posible que el optimizador modifique implícitamente el orden en que se procesan los operandos (por ejemplo, es posible que el optimizador decida qué grado de paralelismo se debe utilizar y qué plan de acceso se debe utilizar), una aplicación que utilice operandos de coma flotante no debe depender de que los resultados sean exactamente iguales cada vez que se ejecuta una sentencia de SQL.

Operandos de coma flotante decimal

Si cualquiera de los operandos de un operador aritmético es de coma flotante decimal, la operación se realiza en coma flotante decimal.

Operandos de coma flotante enteros y decimales
Si un operando es un entero pequeño o un entero grande y el otro es un número DECFLOAT(*n*), la operación se realiza en DECFLOAT(*n*) utilizando una copia temporal del entero que se ha convertido a número DECFLOAT(*n*). Si un operando es un entero grande y el otro es un número de coma flotante decimal, una copia temporal del entero grande se convierte a número DECFLOAT(34). A continuación, se aplican las normas para los operandos de coma flotante de dos decimales.

Operandos de coma flotante decimal y decimales
Si un operando es un decimal y el otro es un número de coma flotante decimal, la operación se realiza en coma flotante decimal utilizando una copia temporal del número decimal que se ha convertido a número de coma flotante decimal basándose en la precisión del número decimal. Si el número decimal tiene una precisión inferior a 17, el número decimal se convierte a número DECFLOAT(16); de lo contrario, el número decimal se convierte a número DECFLOAT(34). A continuación, se aplican las normas para los operandos de coma flotante de dos decimales.

Operandos de coma flotante y de coma flotante decimal
Si un operando es un número (REAL o DOUBLE) de coma flotante y el otro es un número DECFLOAT(*n*), la operación se realiza en coma flotante decimal utilizando una copia temporal del número de coma flotante que se ha convertido a número DECFLOAT(*n*).

Dos operandos de coma flotante decimales
Si ambos operandos son DECFLOAT(*n*), la operación se realiza en DECFLOAT(*n*). Si un operando es DECFLOAT(16) y el otro es DECFLOAT(34), la operación se realiza en DECFLOAT(34).

Normas generales de operaciones aritméticas para coma flotante decimal

Se aplican las siguientes normas generales a todas las operaciones aritméticas en el tipo de datos de coma flotante decimal:

- Cada operación en los números finitos se lleva a cabo como si se calculara un resultado matemático exacto, utilizando la aritmética del entero en el coeficiente, donde sea posible.

  Si el coeficiente del resultado exacto teórico no tiene más dígitos que el que refleja su precisión (16 ó 34), éste se utilizará para el resultado sin cambios (a menos que haya una condición de desbordamiento o subdesbordamiento). Si el coeficiente tiene más dígitos que el que refleja su precisión, éste se redondeará a exactamente el número de dígitos que refleje su precisión (16 ó 34) y el exponente aumentará en el número de dígitos eliminados.

  El registro especial CURRENT DECFLOAT ROUNDING MODE determina la modalidad de redondeo.

  Si el valor del exponente ajustado del resultado es inferior a E_{min}, el coeficiente calculado y el exponente forman el resultado, a menos que el valor del exponente sea inferior a E_{tiny}, en cuyo caso el exponente se define como E_{tiny}, el coeficiente se redondea (posiblemente a cero) para que se corresponda con el ajuste del exponente y el signo permanece sin modificaciones. Si este redondeo proporciona un resultado inexacto, se devolverá una condición de excepción de subdesbordamiento.

  Si el valor del exponente ajustado del resultado es superior a E_{max}, se devolverá una condición de excepción de desbordamiento. En este caso, el resultado se define como una condición de excepción de desbordamiento y podría ser infinito. Tiene el mismo signo que el resultado teórico.
- La aritmética que utiliza el valor especial de infinito sigue las normas habituales, en las que infinito negativo es inferior a todos los números finitos e infinito positivo es superior a todos los números finitos. Basándose en dichas normas, un resultado infinito es siempre exacto. Ciertos usos de infinito devuelven una condición de operación no válida. La lista siguiente muestra las operaciones que pueden ocasionar una condición de operación no válida. El resultado de dicha operación es NaN cuando uno de los operandos es infinito pero el otro operando no es ni NaN ni sNaN.
  - Sumar +infinity a -infinity durante una operación de sumar o restar
  - Multiplicar 0 por +infinito o -infinito
  - Dividir +infinito o -infinito por +infinito o -infinito
  - Cualquiera de los dos argumentos de la función QUANTIZE es +infinito o -infinito
  - El segundo argumento de la función POWER es +infinito o -infinito

– Los Signalling NaN utilizados como operandos para las operaciones aritméticas

Las normas siguientes se aplican a las operaciones aritméticas y al valor de NaN:

– El resultado de cualquier operación aritmética que tenga un operando NaN (Quiet o Signalling) es NaN. El signo del resultado se copia del primer operando que sea un Signalling NaN; si ninguno de los operandos es de señalización (Signalling), el signo se copiará del primer operando que sea NaN. Cada vez que un resultado sea un NaN, el signo del resultado dependerá sólo del operando copiado.
– El signo del resultado de una operación de multiplicación o división sólo es negativo si los operandos tienen signos diferentes y ninguno de ellos es un NaN.
– El signo del resultado de una operación de suma o resta sólo es negativo si el resultado es inferior a cero y ninguno de los operandos es NaN, excepto en los casos siguientes en los que el resultado es un 0 negativo.
  - Un resultado se redondea a cero y el valor, antes del redondeo, tenía un signo negativo
  - -0 se suma a 0
  - 0 se resta de -0
  - Los operandos con signos opuestos se suman o los operandos con el mismo signo se restan; el resultado tiene un coeficiente de 0 y la modalidad de redondeo es ROUND_FLOOR
  - Los operandos se multiplican o dividen, el resultado tiene un coeficiente de 0 y los signos de los operandos son diferentes
  - El primer argumento de la función POWER es -0 y el segundo argumento es un número impar positivo
  - El argumento de la función CEIL, FLOOR o SQRT es -0
  - El primer argumento de la función ROUND o TRUNCATE es -0

Los ejemplos siguientes muestran valores de coma flotante decimal especiales como operandos:

```
INFINITY + 1          = INFINITY
INFINITY + INFINITY   = INFINITY
INFINITY + -INFINITY  = NAN         -- warning
NAN + 1               = NAN
NAN + INFINITY        = NAN
1 - INFINITY          = -INFINITY
INFINITY - INFINITY   = NAN         -- warning
-INFINITY - -INFINITY = NAN         -- warning
-0.0 - 0.0E1          = -0.0
-1.0 * 0.0E1          = -0.0
1.0E1 / 0             = INFINITY    -- warning
-1.0E5 / 0.0          = -INFINITY   -- warning
1.0E5 / -0            = -INFINITY   -- warning
INFINITY / -INFINITY  = NAN         -- warning
INFINITY / 0          = INFINITY
-INFINITY / 0         = -INFINITY
-INFINITY / -0        = INFINITY
```

Tipos definidos por el usuario como operandos

Un tipo definido por el usuario no puede utilizarse con operadores aritméticos ni siquiera aunque el tipo de datos fuente sea numérico. Para llevar a cabo una operación aritmética, cree una función con el operador aritmético como fuente. Por

ejemplo, si existen los tipos diferenciados INCOME y EXPENSES, y ambos tienen tipos de datos DECIMAL(8,2), se podría utilizar la función REVENUE definida por el usuario para restar uno de otro, de la forma siguiente:

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

El operador - (menos) se puede sobrecargar de forma alternativa utilizando la función definida por el usuario para restar los tipos de datos nuevos.

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

Prioridad de las operaciones

Las expresiones entre paréntesis y las expresiones de desreferencia se evalúan primero de izquierda a derecha. (Los paréntesis también se utilizan en sentencias de subselección, condiciones de búsqueda y funciones. Sin embargo, no deben utilizarse para agrupar arbitrariamente secciones dentro de sentencias de SQL.) Cuando del orden de evaluación no se especifica mediante paréntesis, los operadores de prefijo se aplican antes que la multiplicación y división, y la multiplicación y división se aplican antes que la suma y la resta. Los operadores de un mismo nivel de prioridad se aplican de izquierda a derecha.

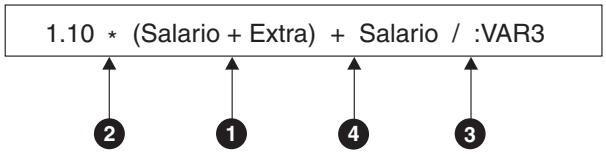


Figura 11. Prioridad de las operaciones

Selección completa escalar

Selección completa escalar:

Una *selección completa escalar* es una selección completa, especificada entre paréntesis, que devuelve una única fila que consta de un único valor de columna. Si la selección completa no devuelve una fila, el resultado de la expresión es el valor nulo. Si el elemento de la lista de selección es una expresión que simplemente es un nombre de columna o una operación de desreferencia, el nombre de columna del resultado está basado en el nombre de la columna. La autorización necesaria para una selección completa escalar es la misma que se necesita para una consulta SQL.

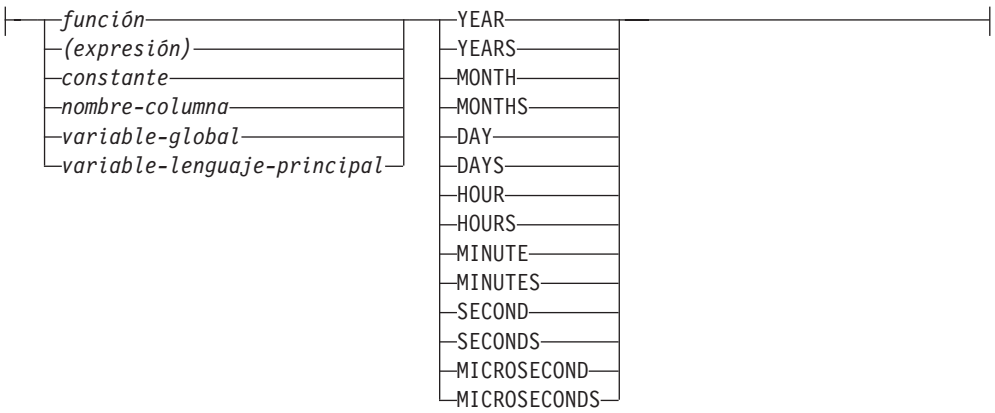
Operaciones de fecha y hora y duraciones

Los valores de fecha y hora se pueden aumentar, disminuir y restar. Estas operaciones pueden incluir números decimales llamados duraciones. Las siguientes secciones describen los tipos de duraciones y las normas para la aritmética de hora y fecha.

Duraciones

Una *duración* es un número que representa un intervalo de tiempo. Existen cuatro tipos de duraciones:

duración-etiquetada:



Una *duración etiquetada* representa una unidad de tiempo específica expresada por un número (que puede ser el resultado de una expresión) seguido de una de las siete palabras clave de duración: YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS o MICROSECONDS. (También se acepta la forma singular de estas palabras clave: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND y MICROSECOND.) El número especificado se convierte como si se asignara a un número DECIMAL(15,0), excepto en el caso de SECONDS, que utiliza DECIMAL(27,12) para permitir que se incluyan de 0 a 12 dígitos de segundos fraccionados. Sólo puede utilizarse una duración etiquetada como operando de un operador aritmético en el que el otro operando sea un valor de tipo de datos DATE, TIME o TIMESTAMP. Así pues, la expresión HIREDATE + 2 MONTHS + 14 DAYS es válida, mientras que la expresión HIREDATE + (2 MONTHS + 14 DAYS) no lo es. En ambas expresiones, las duraciones etiquetadas son 2 MONTHS (meses) y 14 DAYS (días).

Una *duración de fecha* representa un número de años, meses y días, expresados como un número DECIMAL(8,0). Para que se interprete correctamente, el número debe tener el formato *aaaammdd.*, donde *aaaa* representa el número de años, *mm* el número de meses y *dd* el número de días. (El punto en el formato indica un tipo de datos DECIMAL.) El resultado de restar un valor de fecha de otro, como sucede en la expresión HIREDATE - BRTHDATE, es una duración de fecha.

Una *duración de hora* representa un número de horas, minutos y segundos, expresado como un número DECIMAL(6,0). Para interpretarse correctamente, el número debe tener el formato *hhmmss.*, donde *hh* representa el número de horas,

mm el número de minutos y *ss* el número de segundos. (El punto en el formato indica un tipo de datos DECIMAL.) El resultado de restar un valor de hora de otro es una duración de hora.

Una *duración de indicación de fecha y hora* representa un número de años, meses, días, horas, minutos, segundos y segundos fraccionados, expresados como un número DECIMAL(14+*s*,*s*), donde *s* es el número de dígitos de segundos fraccionados que van de 0 a 12. Para que se interprete correctamente, el número debe tener el formato *aaaammddhhmmss.nnnnnnnnnnnn*, donde*aaaa*, *mm*, *dd*, *hh*, *mm*, *ss* y *nnnnnnnnnnnn* representan, respectivamente, el número del año, mes, día, horas, minutos, segundos y segundos fraccionados. El resultado de restar un valor de indicación de fecha y hora de otra es la duración de indicación de fecha y hora, con una escala que coincide con la máxima precisión de la indicación de fecha y hora de sus operandos.

Aritmética de fecha y hora en SQL

Las únicas operaciones aritméticas que pueden efectuarse con valores de fecha y hora son la suma y la resta. Si un valor de fecha y hora es un operando de suma, el otro operando debe ser una duración. A continuación, encontrará las normas específicas que rigen la utilización del operador de suma con valores de fecha y hora.

- Si un operando es una fecha, el otro operando debe ser una duración de fecha o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si un operando es una hora, el otro operando debe ser una duración de hora o una duración etiquetada de HOURS, MINUTES o SECONDS.
- Si un operando es una fecha y hora, el otro operando debe ser una duración. Cualquier tipo de duración es válido.
- Ningún operando del operador de suma puede ser un marcador de parámetro.

Las normas para la utilización del operador de resta con valores de fecha y hora no son las mismas que para la suma, porque un valor de fecha y hora no puede restarse de una duración, y porque la operación de restar dos valores de fecha y hora no es la misma que la operación de restar una duración de un valor de fecha y hora. A continuación se muestran las normas específicas que rigen la utilización del operador de resta con valores de fecha y hora.

- Si el primer operando es una indicación de fecha y hora, el segundo operando debe ser una fecha, una indicación de fecha y hora, una representación de serie de una fecha, una representación de serie de una indicación de fecha y hora o una duración.
- Si el segundo operando es una indicación de fecha y hora, el primer operando debe ser una fecha, una indicación de fecha y hora, una representación de serie de una fecha o una representación de serie de una indicación de fecha y hora.
- El primer operando es una fecha, el segundo operando debe ser una fecha, una duración de fecha, una representación de una fecha en forma de serie o una duración etiquetada de YEARS, MONTHS o DAYS.
- Si el segundo operando es una fecha, el primer operando debe ser una fecha o una representación de una fecha en forma de serie.
- Si el primer operando es una hora, el segundo operando debe ser una hora, una duración de hora, una representación de una hora en forma de serie o una duración etiquetada de HOURS, MINUTES o SECONDS.
- Si el segundo operando es una hora, el primer operando debe ser una hora o una representación de una hora en forma de serie.

- Ningún operando del operador de resta puede ser un marcador de parámetro.

Aritmética de fecha

Las fechas se pueden restar, aumentar o disminuir.

- Al restar una fecha (DATE2) de otra (DATE1) se obtiene como resultado una duración de fecha que especifica el número de años, meses y días entre las dos fechas. El tipo de datos del resultado es DECIMAL(8,0). Si DATE1 es mayor o igual que DATE2, DATE2 se resta de DATE1. Si DATE1 es menor que DATE2, DATE1 se resta de DATE2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = DATE1 - DATE2.

  ```
  Si DAY(DATE2) <= DAY(DATE1)
  entonces DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

  Si DAY(DATE2) > DAY(DATE1)
  entonces DAY(RESULT) = N + DAY(DATE1)
- DAY(DATE2)
  donde N = el último día de MONTH(DATE2).
  MONTH(DATE2) se aumenta en 1.

  Si MONTH(DATE2) <= MONTH(DATE1)
  entonces MONTH(RESULT) = MONTH(DATE1)
- MONTH(DATE2).

  Si MONTH(DATE2) > MONTH(DATE1)
  entonces MONTH(RESULT) = 12 + MONTH(DATE1)
- MONTH(DATE2).
  YEAR(DATE2) se aumenta en 1.

  YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
  ```

  Por ejemplo, el resultado de DATE('15/3/2000') - '31/12/1999' es 00000215. (o una duración de 0 años, 2 meses y 15 días).

- Al añadir o restar una duración a una fecha se obtiene como resultado también una fecha. (En esta operación, un mes equivale a una página de un calendario. La adición de meses a una fecha es como ir pasando páginas a un calendario, empezando por la página en la que aparece la fecha.) El resultado debe estar comprendido entre las fechas 1 de enero de 0001 y 31 de diciembre de 9999, ambos inclusive.

  Si se suma o resta una duración de años, solamente la parte de la fecha correspondiente a los años se verá afectada. Tanto el mes como el día permanecen inalterados, a no ser que el resultado fuera el 29 de febrero en un año no bisiesto. En este caso, el día se cambia a 28 y se define un indicador de aviso en la SQLCA para indicar el ajuste.

  Del mismo modo, si se suma o resta una duración de meses, solamente los meses, y los años si fuera necesario, se verán afectados. La parte de una fecha correspondiente a los años no se cambia a no ser que el resultado no fuera válido (31 de setiembre, por ejemplo). En este caso, el día se establece en el último día del mes y se define un indicador de aviso en la SQLCA para indicar el ajuste.

  Al añadir o restar una duración de días afectará, obviamente, a la parte de la fecha correspondiente a los días y potencialmente al mes y al año.

  Las duraciones de fecha, ya sean positivas o negativas, también se pueden añadir y restar a las fechas. Tal como ocurre con las duraciones etiquetadas, se obtiene como resultado una fecha válida y se define un indicador de aviso en la SQLCA siempre que se deba efectuar un ajuste de fin de mes.

  Cuando se suma una duración de fecha positiva a una fecha, o una duración de fecha negativa se resta de una fecha, la fecha aumenta el número especificado de

años, meses y días, en ese orden. Así pues, DATE1 + X, donde X es un número DECIMAL(8,0) positivo, equivale a la expresión:

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

Cuando una duración de fecha positiva se resta de una fecha, o bien se añade una duración de fecha negativa a una fecha, la fecha disminuye en el número días, meses y años especificados, en este orden. Así pues, DATE1 - X, donde X es un número DECIMAL(8,0) positivo, equivale a la expresión:

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

Al añadir duraciones a fechas, la adición de un mes a una fecha determinada da la misma fecha un mes posterior a menos que la fecha no exista en el siguiente mes. En este caso, se establece la fecha correspondiente al último día del siguiente mes. Por ejemplo, 28 de enero más un mes da como resultado 28 de febrero y si se añade un mes al 29, 30 ó 31 de enero también se obtendrá como resultado el 28 de febrero o bien 29 de febrero si se trata de un año bisiesto.

Nota: Si se añade uno o más meses a una fecha determinada y del resultado se resta la misma cantidad de meses, la fecha final no tiene por qué ser necesariamente la misma que la original.

Aritmética de las horas

Las horas se pueden restar, aumentar o disminuir.

- El resultado de restar una hora (HOUR2) de otra (HOUR1) es una duración que especifica el número de horas, minutos y segundos entre las dos horas. El tipo de datos del resultado es DECIMAL(6,0).

  Si HOUR1 es mayor o igual que HOUR2, HOUR2 se resta de HOUR1.

  Si HOUR1 es menor que HOUR2, HOUR1 se resta de HOUR2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = HOUR1 - HOUR2.

  ```
  Si SECOND(TIME2) <= SECOND(TIME1)
  entonces SECOND(RESULT) = SECOND(HOUR1) - SECOND(HOUR2).

  Si SECOND(TIME2) > SECOND(TIME1)
  entonces SECOND(RESULT) =
  60 + SECOND(HOUR1) - SECOND(HOUR2).
  MINUTE(HOUR2) se aumenta entonces en 1.

  Si MINUTE(TIME2) <= MINUTE(TIME1)
  entonces MINUTE(RESULT) = MINUTE(HOUR1)
- MINUTE(HOUR2).

  Si MINUTE(TIME1) > MINUTE(TIME1)
  entonces MINUTE(RESULT) =
  60 + MINUTE(HOUR1) - MINUTE(HOUR2).
  HOUR(HOUR2) se aumenta entonces en 1.

  HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).
  ```

  Por ejemplo, el resultado de TIME('11:02:26') - '00:32:56' es 102930. (una duración de 10 horas, 29 minutos y 30 segundos).

- El resultado de sumar una duración a una hora, o de restar una duración de una hora, es una hora. Se rechaza cualquier desbordamiento o subdesbordamiento de horas, garantizando de este modo que el resultado sea siempre una hora. Si se suma o resta una duración de horas, sólo se ve afectada la parte correspondiente a las horas. Los minutos y los segundos no cambian.

  De manera parecida, si se suma o resta una duración de minutos, sólo se afecta a los minutos y, si fuera necesario, a las horas. La parte correspondiente a los segundos no cambia.

Al añadir o restar una duración de segundos afectará, obviamente, a la parte de la fecha correspondiente a los segundos y potencialmente a los minutos y a las horas.

Las duraciones de hora, tanto positivas como negativas, pueden también sumarse y restarse a las horas. El resultado es una hora que se ha incrementado o disminuido en el número especificado de horas, minutos y segundos, por ese orden. `TIME1 + X`, donde "X" es un número DECIMAL(6,0), equivale a la expresión:

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

Al restar una duración etiquetada de SECOND o SECONDS que tiene un valor que incluye fracciones de un segundo, la resta se realiza como si el valor de tiempo tuviera 12 dígitos de segundos fraccionarios como máximo, pero el resultado se devuelve con los segundos fraccionarios truncados.

Nota: Aunque la hora '24:00:00' se acepta como una hora válida, no se devuelve nunca como resultado de una suma o resta de horas, ni siquiera aunque el operando de duración sea cero (por ejemplo, hora('24:00:00')±0 segundos = '00:00:00').

Aritmética de la indicación de fecha y hora

Las indicaciones de fecha y hora se pueden restar, incrementar o disminuir.

- El resultado de restar una indicación de fecha y hora (TS2) de otra (TS1) es una duración que especifica el número de año, mes, día, horas, minutos, segundos y segundos fraccionados entre las dos indicaciones de fecha y hora. El tipo de datos del resultado es DECIMAL(14+*s*,*s*), donde *s* es la máxima precisión de la indicación de fecha y hora de TS1 y TS2.

  Si TS1 es mayor o igual que TS2, TS2 se resta de TS1. Si TS1 es menor que TS2, TS1 se resta de TS2 y el signo del resultado se convierte en negativo. La descripción siguiente clarifica los pasos que intervienen en el resultado de la operación = TS1 - TS2:

  ```
  Si SECOND(TS2,s) <= SECOND(TS1,s)
  entonces SECOND(RESULT,s) = SECOND(TS1,s) -
  SECOND(TS2,s).

  Si SECOND(TS2,s) > SECOND(TS1,s)
  entonces SECOND(RESULT,s) = 60 +
  SECOND(TS1,s) – SECOND(TS2,s).
  MINUTE(TS2) se aumenta entonces en 1.
  ```

  La parte correspondiente a los minutos de las indicaciones de fecha y hora se resta tal como se especifica en las normas para la resta de horas.

  ```
  Sí HOUR(TS2) <= HOUR(TS1)
  entonces HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).

  Si HOUR(TS2) > HOUR(TS1)
  entonces HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
  y DAY(TS2) se aumenta en 1.
  ```

  La parte correspondiente a la fecha de las indicaciones de fecha y hora se resta tal como se especifica en las normas para la resta de fechas.

- El resultado de restar una fecha (D1) de una indicación de fecha y hora (TS1) es el mismo que restar TIMESTAMP(D1) de TS1. De un modo similar, el resultado de restar una indicación de fecha y hora (TS1) de una fecha (D2) es el mismo que restar TS1 de TIMESTAMP(D2).
- El resultado de sumar una duración a una indicación de fecha y hora, o de restar una duración de una indicación de fecha y hora, es en sí mismo una indicación de fecha y hora.La precisión del resultado coincide con la precisión

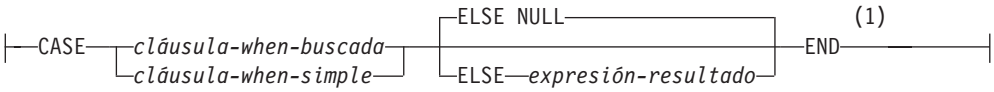
del operando de la indicación de fecha y hora. La parte correspondiente a la aritmética de fecha se realiza tal como se ha definido anteriormente, excepto que se acarrea un desbordamiento o subdesbordamiento a la parte de fecha del resultado, que debe estar dentro del rango de fechas válidas. La parte correspondiente a la aritmética de fecha es parecida a la aritmética de hora, salvo por el hecho de que toman en cuenta los segundos fraccionarios incluidos en la duración. Por lo tanto, la resta de una duración, *X*, de una indicación de fecha y hora, TIMESTAMP1, donde *X* es un número DECIMAL(14+s,s), es equivalente a la expresión:

```
TIMESTAMP1 - YEAR(X)  YEARS - MONTH(X)  MONTHS - DAY(X)  DAYS
           - HOUR(X)  HOURS - MINUTE(X)  MINUTES - SECOND(X, s)  SECONDS
```

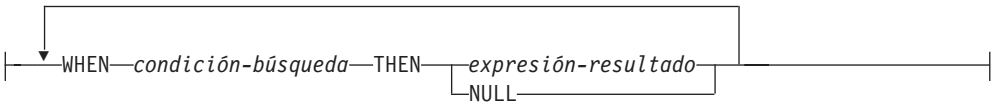
Al restar una duración con escala distinta de cero o una duración etiquetada de SECOND o SECONDS que tiene un valor que incluye fracciones de un segundo, la resta se realiza como si el valor de indicación de fecha y hora tuviera 12 dígitos de segundos fraccionarios como máximo. El valor resultante se asigna a un valor de indicación de fecha y hora con la precisión de fecha y hora del operando de la indicación de fecha y hora, lo que podría provocar la truncación de los dígitos de segundos fraccionarios.

Expresión CASE

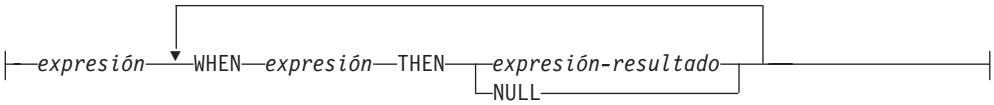
expresión-case:



cláusula-searched-when:



cláusula-when-simple:



Notas:

1 Si el tipo de resultado de *expresión-resultado* es un tipo de fila, la sintaxis representa una *expresión-case-fila* y sólo se puede utilizar donde se permita una *expresión-fila*.

Las expresiones CASE permiten seleccionar una expresión en función de la evaluación de una o varias condiciones. En general, el valor de la *expresión-case* es el valor de la *expresión-resultado* que sigue a la primera expresión case (la que está más a la izquierda) que se evalúa como cierta. Si ninguna se evalúa como cierta y está presente la palabra clave ELSE, el resultado es el valor de la *expresión-resultado* o NULL. Si ninguna se evalúa como cierta y no se utiliza la palabra clave ELSE, el resultado es NULL. Tenga presente que cuando una expresión CASE se evalúa como desconocida (debido a valores NULL), la expresión CASE no es cierta y por eso se trata igual que una expresión CASE que se evalúa como falsa.

Si la expresión CASE está en una cláusula VALUES, un predicado IN, una cláusula GROUP BY o en una cláusula ORDER BY, la *condición-búsqueda* de una *cláusula-searched-when* no puede ser un predicado cuantificado, un predicado IN que hace uso de una selección completa ni un predicado EXISTS (SQLSTATE 42625).

Cuando se utiliza la *cláusula-simple-when*, se comprueba si el valor de la *expresión* anterior a la primera palabra clave *WHEN* es igual al valor de la *expresión* posterior a la palabra clave WHEN. Por lo tanto, el tipo de datos de la *expresión* anterior a la primera palabra clave *WHEN* debe ser comparable a los tipos de datos de cada *expresión* posterior a la palabra o palabras clave WHEN. La *expresión* antes de la primera palabra clave *WHEN* en una *cláusula-when-simple* no se puede incluir una función que sea no determinista o que tenga una acción externa (SQLSTATE 42845).

Una *expresión-resultado* es una *expresión* que sigue a las palabras clave THEN o ELSE. Debe haber, como mínimo, una *expresión-resultado* en la expresión CASE

(NULL no puede especificarse para cada case) (SQLSTATE 42625). Todas las expresiones-resultado deben tener tipos de datos compatibles (SQLSTATE 42804).

Ejemplos

- Si el primer carácter de un número de departamento corresponde a una división dentro de la organización, se puede utilizar una expresión CASE para listar el nombre completo de la división a la que pertenece cada empleado:

```
SELECT EMPNO, LASTNAME,
  CASE SUBSTR(WORKDEPT,1,1)
    WHEN 'A' THEN 'Administración'
    WHEN 'B' THEN 'Recursos humanos'
    WHEN 'C' THEN 'Contabilidad'
    WHEN 'D' THEN 'Diseño'
    WHEN 'E' THEN 'Operaciones'
  END
  FROM EMPLOYEE;
```

- El número de años de formación académica se usa en la tabla EMPLOYEE para obtener el nivel de formación. Una expresión CASE se puede utilizar para agrupar estos datos y para mostrar el nivel de formación.

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
  CASE
    WHEN EDLEVEL < 15 THEN 'SECONDARY'
    WHEN EDLEVEL < 19 THEN 'COLLEGE'
    ELSE 'POST GRADUATE'
  END
  FROM EMPLOYEE
```

- Otro ejemplo interesante del uso de una expresión CASE consiste en la protección de los errores que surjan de una división por 0. Por ejemplo, el siguiente código detecta los empleados que perciben más de un 25% de sus ingresos en comisiones, pero que su sueldo no se basa enteramente en comisiones.

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
  ELSE COMM/SALARY
  END) > 0.25;
```

- Las siguientes expresiones CASE son iguales:

```
SELECT LASTNAME,
  CASE
  WHEN LASTNAME = 'Haas' THEN 'Presidente'
  ...

SELECT LASTNAME,
  CASE LASTNAME
  WHEN 'Haas' THEN 'Presidente'
  ...
```

Existen dos funciones escalares, NULLIF y COALESCE, que sirven exclusivamente para manejar un subconjunto de la funcionalidad que una expresión CASE puede ofrecer. La Tabla 23 muestra la expresión equivalente al utilizar CASE o estas funciones.

Tabla 23. Expresiones CASE equivalentes

Expresión	Expresión equivalente

CASE
  WHEN e1=e2 THEN NULL
  ELSE e1
END
	NULLIF(e1,e2)

Tabla 23. Expresiones CASE equivalentes (continuación)

Expresión	Expresión equivalente

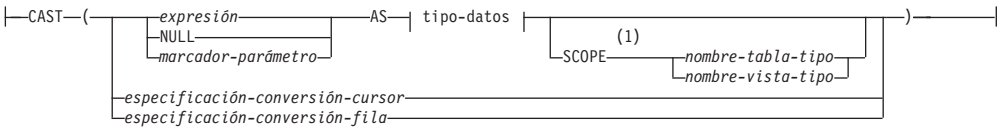
CASE
  WHEN e1 IS NOT NULL THEN e1
  ELSE e2
END
	COALESCE(e1,e2)

CASE
  WHEN e1 IS NOT NULL THEN e1
  ELSE COALESCE(e2,...,eN)
END
	COALESCE(e1,e2,...,eN)

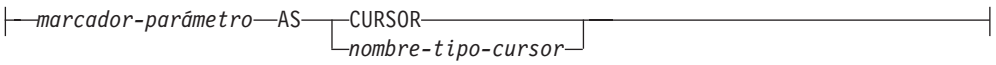
CASE
  WHEN c1=var1 OR (c1 IS NULL AND var1 IS NULL)
  THEN 'a'
  WHEN c1=var2 OR (c1 IS NULL AND var2 IS NULL)
  THEN 'b'
  ELSE NULL
END
	DECODE(c1,var1, 'a', var2, 'b')

Especificación CAST

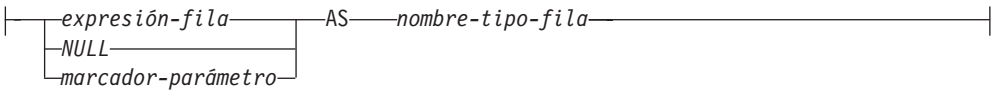
especificación-cast:



especificación-conversión-cursor:

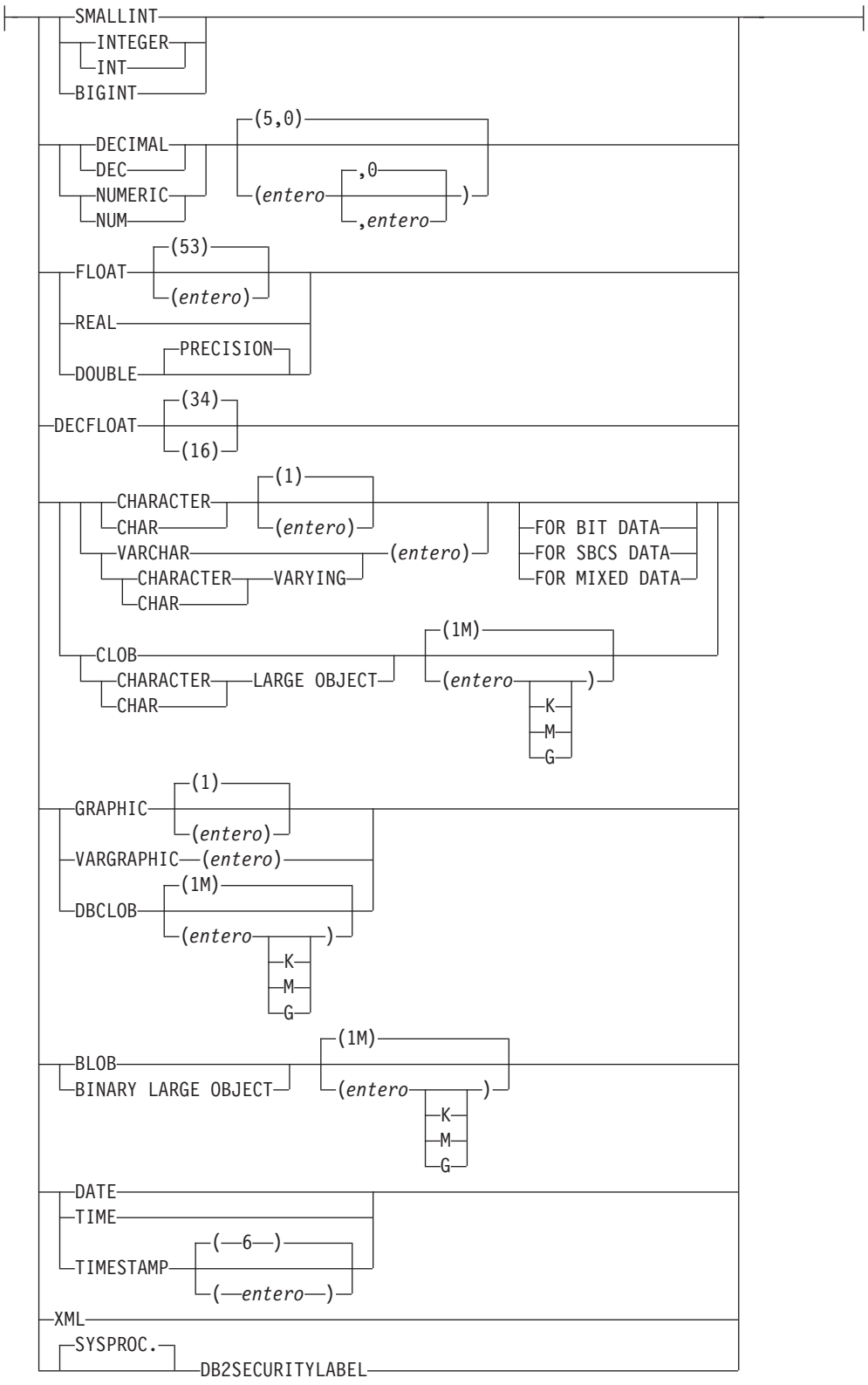


especificación-conversión-fila:



tipo-datos:

tipo-incorporado:



Notas:

1 La cláusula SCOPE sólo se aplica al tipo de datos REF.

La especificación CAST devuelve el operando cast (el primer operando) convertido al tipo especificado por el *tipo-datos*. Si no se soporta cast, se devuelve un error (SQLSTATE 42846).

expresión
Si el operando cast es una expresión (distinta del marcador de parámetro o NULL), el resultado es el valor del argumento convertido al *tipo-datos* de destino especificado.

Al convertir series de caracteres (que no sean CLOB) en una serie de caracteres de longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Al convertir series de caracteres gráficas (que no sean DBCLOB) en una serie de caracteres gráfica con una longitud diferente, se devuelve un aviso (SQLSTATE 01004) si se truncan otros caracteres que no sean los blancos de cola. Para los operandos BLOB, CLOB y DBCLOB de cast, el mensaje de aviso aparece si se trunca cualquier carácter.

Al convertir una matriz, el tipo de datos de destino debe ser un tipo de datos de matriz definido por el usuario (SQLSTATE 42821). El tipo de datos de los elementos de la matriz debe ser el mismo que el tipo de datos de los elementos del tipo de datos de la matriz de destino (SQLSTATE 42846). La cardinalidad de la matriz debe ser inferior o igual a la cardinalidad máxima del tipo de datos de la matriz de destino (SQLSTATE 2202F).

NULL
Si el operando cast es la palabra clave NULL, el resultado es un valor nulo que tiene el *tipo -datos* especificado.

marcador-parámetro
Un marcador de parámetro suele considerarse como una expresión, pero en este caso se documenta por separado porque tiene un significado especial. Si el operando cast es un *marcador-parámetros*, el *tipo-datos* especificado se considera una promesa de que la sustitución se podrá asignar al tipo de datos especificado (utilizando la asignación de almacenamiento para series). Un marcador de parámetro como este se considera un *marcador de parámetro con tipo*. Los marcadores de parámetro con tipo se considerarán como cualquier otro valor con tipo para la resolución de funciones, para DESCRIBE de una lista de selección o para la asignación de columnas.

especificación-conversión-cursor
Especificación de conversión que se utiliza para indicar que se espera que un marcador de parámetro sea un tipo de cursor. Puede utilizarse siempre que se dé soporte a una expresión en contextos que permitan tipos de cursor.

marcador-parámetro
El operando de conversión es un marcador de parámetro y se considera un compromiso de que la sustitución se podrá asignar al tipo de cursor especificado.

CURSOR
Especifica el tipo de datos incorporado CURSOR.

nombre-tipo-cursor
Especifica el nombre de un tipo de cursor definido por el usuario.

especificación-conversión-fila
Especificación de conversión donde la entrada es un valor de fila y el resultado es un tipo de fila definido por el usuario. Una *especificación-conversión-fila* sólo es válida allí donde se permite una *expresión-fila*.

expresión-fila
El tipo de datos de una *expresión-fila* debe ser una variable de tipo de fila que esté anclada en la definición de una tabla o vista. El tipo de datos de *expresión-fila* no puede ser un tipo de fila definido por el usuario (SQLSTATE 42846).

NULL
Especifica que el operando de conversión es el valor nulo. El resultado es una fila con el valor nulo de cada campo del tipo de datos especificado.

marcador-parámetro
El operando de conversión es un marcador de parámetro y se considera un compromiso de que la sustitución se podrá asignar al *nombre-tipo-fila*.

nombre-tipo-fila
Especifica el nombre de un tipo de fila definido por el usuario. La *expresión-fila* debe poderse convertir en el *nombre-tipo-fila* (SQLSTATE 42846).

tipo-datos
Nombre de un tipo de datos existente. Si el nombre del tipo no está calificado, la vía de acceso de SQL se utiliza para realizar la resolución del tipo de datos. Un tipo de datos que tiene atributos asociados como, por ejemplo, la longitud o precisión y la escala, debería incluir estos atributos al especificar *tipo-datos*. (CHAR toma por omisión una longitud de 1,DECIMAL toma por omisión una precisión de 5 y una escala de 0 y DECFLOAT toma por omisión una precisión de 34 si no se especifica.) Para convertir una serie FOR BIT DATA a la página de códigos de la base de datos se puede utilizar la cláusula FOR SBCS DATA o la cláusula FOR MIXED DATA (sólo se da soporte a una, según si la base de datos da soporte o no al tipo de datos gráfico). Las restricciones sobre los tipos de datos soportados se basan en el operando cast especificado.

- Para un operando cast que sea una *expresión*, los tipos de datos de destino a los que se da soporte dependen del tipo de datos del operando cast (tipo de datos fuente).
- Para un operando cast que sea la palabra clave NULL se puede utilizar cualquier tipo de datos existente.
- Para un operando cast que sea un marcador de parámetro, el tipo de datos de destino puede ser cualquier tipo de datos existente. Si el tipo de datos es un tipo diferenciado definido por el usuario, la aplicación que hace uso del marcador de parámetro utilizará el tipo de datos fuente del tipo diferenciado definido por el usuario. Si el tipo de datos es un tipo estructurado definido por el usuario, la aplicación que hace uso del marcador de parámetro utilizará el tipo de parámetro de entrada de la función de transformación TO de SQL para el tipo estructurado definido por el usuario.

SCOPE
Cuando el tipo de datos es un tipo de referencia, puede definirse un ámbito que identifique la tabla de destino o la vista de destino de la referencia.

nombre-tabla-tipo
El nombre de una tabla con tipo. Ya debe existir la tabla (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM).

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir o tener el mismo nombre que la vista a crear que incluye la conversión del tipo de datos

como parte de la definición de la vista (SQLSTATE 42704). La conversión debe hacerse hacia el *tipo-datos* REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM).

Cuando se convierten datos numéricos en datos de caracteres, el tipo de datos resultante es una serie de caracteres de longitud fija. Cuando se convierten datos de caracteres en datos numéricos, el tipo de datos resultante depende del tipo de número especificado. Por ejemplo, si se convierte en un entero, pasará a ser un entero grande.

Ejemplos

- A una aplicación sólo le interesa la parte entera de SALARY (definido como decimal (9,2)) de la tabla EMPLOYEE. Se podría preparar la siguiente consulta, con el número de empleado y el valor del entero de SALARY.

  ```
  SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
  ```

- Supongamos que hay un tipo diferenciado denominado T_AGE que se define como SMALLINT y se utiliza para crear la columna AGE en la tabla PERSONNEL. Supongamos también que existe también un tipo diferenciado denominado R_YEAR que está definido en INTEGER y que se utiliza para crear la columna RETIRE_YEAR en la tabla PERSONNEL. Se podría preparar la siguiente sentencia de actualización.

  ```
  UPDATE PERSONNEL SET RETIRE_YEAR =?
    WHERE AGE = CAST( ? AS T_AGE)
  ```

  El primer parámetro es un marcador de parámetro sin tipo que tendría un tipo de datos de R_YEAR, si bien la aplicación utilizará un entero para este marcador de parámetro. Esto no necesita la especificación explícita de CAST porque se trata de una asignación.

  El segundo marcador de parámetro es un marcador de parámetro con tipo que se convierte como un tipo diferenciado T_AGE. Esto cumple el requisito de que la comparación debe realizarse con tipos de datos compatibles. La aplicación utilizará el tipo de datos fuente (que es SMALLINT) para procesar este marcador de parámetro.

  El proceso satisfactorio de esta sentencia supone que la vía de acceso SQL incluye el nombre de esquema del esquema (o esquemas) donde están definidos los dos tipos diferenciados.

- Una aplicación proporciona un valor que es una serie de bits, por ejemplo una corriente de audio, y no se debe realizar la conversión de página de códigos antes de que se utilice en una sentencia de SQL. La aplicación podría utilizar la siguiente función CAST:

  ```
  CAST( ? AS VARCHAR(10000) FOR BIT DATA)
  ```

- Suponga que se han creado un tipo de matriz y una tabla de la siguiente manera:

  ```
  CREATE TYPE PHONELIST AS DECIMAL(10, 0) ARRAY[5]

  CREATE TABLE EMP_PHONES
    (ID          INTEGER,
     PHONENUMBER DECIMAL(10,0) )
  ```

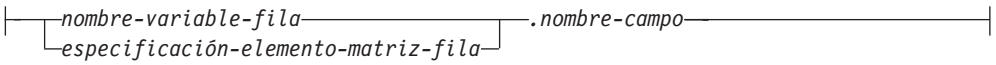
  El procedimiento siguiente devuelve una matriz con los números de teléfono del empleado con el ID 1775. Si hay más de cinco números de teléfono de este empleado, se devuelve un error (SQLSTATE 2202F).

  ```
  CREATE PROCEDURE GET_PHONES(OUT EPHONES PHONELIST)
  BEGIN
    SELECT CAST(ARRAY_AGG(PHONENUMBER) AS PHONELIST)
  ```

```
  INTO EPHONES
  FROM EMP_PHONES
  WHERE ID = 1775;
END
```

Referencia a campo

referencia-campo:



Campo de tipo de fila al que se hace referencia mediante el nombre de campo calificado por:

- Una variable que devuelve un tipo de fila que incluye un campo con ese nombre de campo
- Una especificación de elemento de matriz que devuelve un tipo de fila que incluye un campo con ese nombre de campo

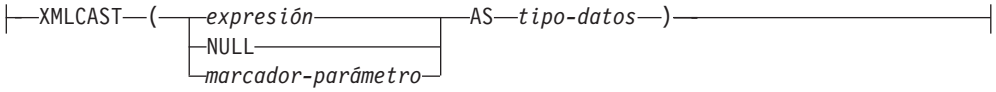
nombre-variable-fila
El nombre de una variable con un tipo de datos que es un tipo de fila.

especificación-elemento-matriz-fila
Una *especificación-elemento-matriz* donde el tipo de datos del elemento de matriz es un tipo de fila.

nombre-campo
El nombre de un campo dentro de un tipo de fila.

Especificación XMLCAST

especificación-xmlcast:



La especificación XMLCAST devuelve el operando cast (el primer operando) convertido al tipo especificado por el tipo de datos. XMLCAST da soporte a conversiones que requieren valores XML, incluidas las conversiones entre los tipos de datos que no son XML y el tipo de datos XML. Si la conversión no está soportada se devuelve un error (SQLSTATE 22003).

expresión
Si el operando cast es una expresión (distinta del marcador de parámetro o NULL), el resultado es el valor del argumento convertido al tipo de datos de destino especificado. El tipo de datos de destino o la expresión deben ser del tipo de datos XML (SQLSTATE 42846).

NULL
Si el operando cast es la palabra clave NULL, el tipo de datos de destino debe ser el tipo de datos XML (SQLSTATE 42846). El resultado es un valor XML nulo.

marcador-parámetro
Si el operando cast es un marcador de parámetro, el tipo de datos de destino debe ser XML (SQLSTATE 42846). Un marcador de parámetro suele considerarse que es una expresión, pero en este caso se documenta por separado porque tiene un significado especial. Si el operando cast es un marcador de parámetro, el tipo de datos especificado se considera una promesa de que la sustitución se podrá asignar al tipo de datos especificado (utilizando la asignación de almacenamiento). Un marcador de parámetro de este tipo se considera un marcador de parámetro con tipo, y se trata como cualquier otro valor con tipo para fines de resolución de función, una operación de descripción en una lista de selección o una asignación de columna.

tipo-datos
El nombre de un tipo de datos SQL existente. Si el nombre no está calificado, la vía de acceso de SQL se utiliza para realizar la resolución del tipo de datos. Si un tipo de datos tiene atributos asociados como, por ejemplo, la longitud o precisión y la escala, estos atributos deben incluirse cuando se especifica un valor para *tipo-datos*. CHAR toma por omisión una longitud de 1 y DECIMAL toma por omisión una precisión de 5 y una escala de 0 si no se especifican. Las restricciones sobre los tipos de datos soportados se basan en el operando cast especificado.

- Para un operando cast que sea una expresión, los tipos de datos de destino a los que se da soporte dependen del tipo de datos del operando cast (tipo de datos fuente).
- Para un operando cast que sea la palabra clave NULL, el tipo de datos de destino debe ser XML.
- Para un operando cast que sea un marcador de parámetro, el tipo de datos de destino debe ser XML.

Nota: Soporte en bases de datos no Unicode: Cuando se utiliza XMLCAST para convertir un valor XML en un tipo de datos SQL, se realiza la conversión de página de códigos. La codificación de la expresión de conversión se convierte desde UTF-8 a la página de códigos de la base de datos. Como consecuencia de esta conversión, los caracteres de la expresión original que no estén presentes en la página de códigos de la base de datos se sustituirán por caracteres de sustitución.

Ejemplos

- Crear un valor XML nulo.

```
XMLCAST(NULL AS XML)
```

- Convertir un valor extraído de una expresión XMLQUERY en un INTEGER:

```
XMLCAST(XMLQUERY('$m/PRODUCT/QUANTITY'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS INTEGER)
```

- Convertir un valor extraído de una expresión XMLQUERY en una serie de caracteres de longitud variable:

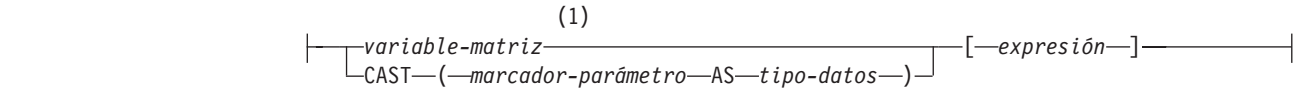
```
XMLCAST(XMLQUERY('$m/PRODUCT/ADD-TIMESTAMP'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS VARCHAR(30))
```

- Convertir un valor extraído de una subconsulta escalar SQL en un valor XML.

```
XMLCAST((SELECT quantity FROM product AS p
  WHERE p.id = 1077) AS XML)
```

Especificación del elemento ARRAY

especificación-elemento-matriz:



Notas:

1 Si el tipo de datos de los elementos de la matriz es un tipo de fila, la sintaxis representa una especificación-elemento-matriz con un tipo de datos de fila y sólo se puede utilizar donde se permita una *expresión-fila*.

La especificación del elemento ARRAY devuelve el elemento de una matriz especificada por medio de *expresión*. Si algún argumento de *expresión* es nulo, se devuelve el valor nulo.

variable-matriz
Una variable de SQL, un parámetro de SQL o una variable global con tipo de matriz.

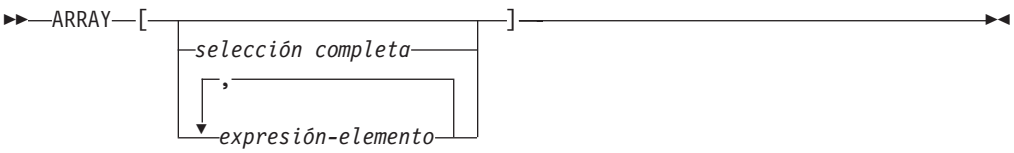
CAST (*marcador-parámetro* **AS** *tipo-datos***)**
Normalmente, se considera que un marcador de parámetro es una expresión, aunque en este caso debe convertirse explícitamente a un tipo de datos de matriz definido por el usuario.

[*expresión***]**
Especifica el índice de matriz del elemento que ha de extraerse de la matriz. El índice de matriz de una matriz común debe ser asignable a INTEGER (SQLSTATE 428H1); su valor debe estar entre 1 y la cardinalidad de la matriz (SQLSTATE 2202E). El índice de matriz de una matriz asociativa se debe poder asignar al tipo de datos de índice (SQLSTATE 428H1).

Constructor de matrices

Un constructor de matrices es un elemento de lenguaje que se puede utilizar para definir y construir un valor de tipo de datos de matriz dentro de un contexto válido.

Sintaxis



Autorización

No se necesitan autorizaciones específicas para hacer referencia a un constructor de matrices dentro de una sentencia de SQL. No obstante, para que la ejecución de la sentencia resulte satisfactoria, se deben cumplir los demás requisitos de autorización de la sentencia.

Descripción

ARRAY[]
Especifica una matriz vacía.

ARRAY[*selección-completa***]**
Especifica una matriz cuyos elementos son las filas de resultado de una *selección-completa* que devuelve una sola columna.

Si la *selección-completa* incluye una *cláusula-order-by*, el orden determina el orden en que los valores de fila se asignan a los elementos de la matriz. Si no se especifica ninguna *cláusula-order-by*, el orden en que los valores de fila se asignan a los elementos de la matriz no está determinado.

ARRAY[*expresión-elemento***,...]**
Especifica una matriz que utiliza el valor de una expresión para cada elemento. La primera *expresión-elemento* se asigna al elemento de la matriz con índice de matriz 1. La segunda *expresión-elemento* se asigna al elemento de la matriz con índice de matriz 2, y así sucesivamente. Cada *expresión-elemento* debe tener un tipo de datos compatible con las demás *expresiones-elemento*, donde el tipo base de la matriz se determina de acuerdo con las "Normas para tipos de datos de resultados".

Normas

- El tipo base del *constructor-matriz*, tal como se deriva de las *expresiones-elemento* o la *selección-completa*, debe poderse asignar al tipo base de la matriz de destino (SQLSTATE 42821).
- El número de elementos del *constructor-matriz* no debe superar la cardinalidad máxima de la variable de matriz de destino (SQLSTATE 2202F).

Notas

- Un constructor de matrices se puede utilizar para definir únicamente una matriz común con elementos que no son de tipo fila. Un constructor de matrices no se puede utilizar para definir una matriz asociativa o una matriz común con elementos de tipo fila. Este tipo de matrices solamente se puede construir mediante la asignación de elementos individuales.

Ejemplos

Ejemplo 1: Establecer la variable de matriz RECENT_CALLS del tipo de matrizPHONENUMBERS como una matriz de números fijos.

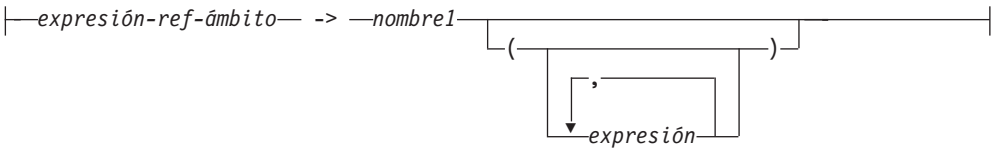
```
SET RECENT_CALLS = ARRAY[9055553907, 4165554213, 4085553678]
```

Ejemplo 2: Establecer la variable de matriz DEPT_PHONES del tipo de matriz PHONENUMBERS como una matriz de números de teléfono recuperados de la tabla DEPARTMENT_INFO.

```
SET DEPT_PHONES = ARRAY[SELECT DECIMAL(AREA_CODE CONCAT '555' CONCAT EXTENSION,16)
                                  FROM DEPARTMENT_INFO
                                  WHERE DEPTID = 624]
```

Operación de desreferencia

operación-desreferencia:



El ámbito de la expresión de referencia con ámbito es una tabla o vista llamada tabla o vista *destino*. La expresión de referencia con ámbito identifica una *fila destino*. La *fila destino* es la fila de la tabla o vista destino (o de una sus subtablas o subvistas) cuyo valor de la columna de identificador de objeto (OID) coincide con la expresión de referencia. Se puede utilizar la operación de desreferencia para acceder a una columna de la fila destino, o para invocar un método, utilizando la fila destino como sujeto del método. El resultado de una operación de desreferencia puede siempre ser nulo. La operación de desreferencia tiene prioridad por encima de todos los otros operadores.

expresión-ref-ámbito
Una expresión que es un tipo de referencia que tiene un ámbito (SQLSTATE 428DT). Si la expresión es una variable del lenguaje principal, un marcador de parámetro u otro valor de tipo de referencia sin ámbito, se necesita una especificación CAST con una cláusula SCOPE para proporcionar un ámbito a la referencia.

name1
Especifica un identificador no calificado.

Si *nombre1* no va seguido por ningún paréntesis y *nombre1* coincide con el nombre de un atributo del tipo destino, el valor de la operación de desreferencia es el valor de la columna mencionada de la fila destino. En este caso, el tipo de datos de la columna (que puede contener nulos) determina el tipo del resultado de la operación de desreferencia. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el resultado de la operación de desreferencia es nulo. Si la operación de desreferencia se utiliza en una lista de selección y no se incluye como parte de una expresión, *nombre1* pasa a ser el nombre de la columna resultante.

Si *nombre1* va seguido por un paréntesis o *nombre1* no coincide con el nombre de un atributo del tipo destino, la operación de desreferencia se trata como una invocación de método. El nombre del método invocado es *nombre1*. El sujeto del método es la fila destino, que se considera como una instancia de su tipo estructurado. Si no existe ninguna fila destino cuyo identificador de objeto coincida con la expresión de referencia, el sujeto del método es un valor nulo del tipo destino. Las expresiones entre paréntesis, si las hay, proporcionan los restantes parámetros de la invocación del método. El proceso normal se utiliza para la resolución de la invocación del método. El tipo resultante del método seleccionado (que puede contener nulos) determina el tipo resultante de la operación de desreferencia.

El ID de autorización de la sentencia que utiliza una operación de desreferencia debe tener el privilegio SELECT sobre la tabla de destino de la *expresión-ref-ámbito* (SQLSTATE 42501).

Una operación de desreferencia no puede nunca modificar valores de la base de datos. Si se utiliza una operación de desreferencia para invocar un método mutador, éste modifica una copia de la fila destino y devuelve la copia, dejando inalterada la base de datos.

Ejemplos

- Suponga que existe una tabla EMPLOYEE que contiene una columna denominada DEPTREF, que es un tipo de referencia con ámbito para una tabla con tipo basada en un tipo que incluye el atributo DEPTNAME. Los valores de DEPTREF de la tabla EMPLOYEE deben corresponderse con los valores de la columna de OID de la tabla de destino de la columna DEPTREF.

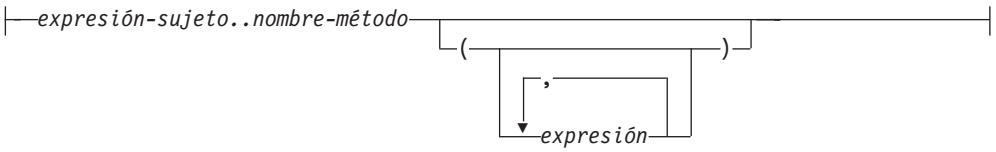
```
SELECT EMPNO, DEPTREF->DEPTNAME
  FROM EMPLOYEE
```

- Utilizando las mismas tablas que en el ejemplo anterior, utilice una operación de desreferencia para invocar un método llamado BUDGET, con la fila destino como parámetro sujeto y '1997' como parámetro adicional.

```
   SELECT EMPNO,
DEPTREF->BUDGET('1997') AS DEPTBUDGET97
     FROM EMPLOYEE
```

Invocación de métodos

invocación-método:



El método observador y el método mutador, ambos generados por el sistema, así como los métodos definidos por el usuario se invocan utilizando el operador formado por dos puntos.

expresión-sujeto
Es una expresión con un tipo resultante estático que es un tipo estructurado definido por el usuario.

nombre-método
Es el nombre no calificado de un método. El tipo estático de *expresión-sujeto* o uno de sus supertipos debe incluir un método que tenga el nombre especificado.

(*expresión***,...)**
Los argumentos de *nombre-método* se especifican entre paréntesis. Se pueden utilizar paréntesis vacíos para indicar que no existen argumentos. El *nombre-método* y los tipos de datos de las expresiones argumento especificadas se utilizan para obtener el método específico, basándose en el tipo estático de *expresión-sujeto*.

El operador .. utilizado para invocar el método es un operador infijo que define una prioridad de operaciones de izquierda a derecha. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
a..b..c + x..y..z
```

y

```
((a..b)..c) + ((x..y)..z)
```

Si un método no tiene ningún otro parámetro que no sea su sujeto, éste se puede invocar con o sin paréntesis. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
point1..x
point1..x()
```

Los sujetos nulos de una invocación de método se gestionan de este modo:

- Si un método mutador generado por el sistema se invoca con un sujeto nulo, se produce un error (SQLSTATE 2202D)
- Si cualquier método distinto de un método mutador generado por el sistema se invoca con un sujeto nulo, el método no se ejecuta y su resultado es nulo. Esta norma incluye los métodos definidos por el usuario con SELF AS RESULT.

Cuando se crea un objeto de base de datos (por ejemplo, un paquete, vista o activador), se determina el método de ajuste óptimo que existe para cada invocación de método.

Nota: Los métodos de los tipos definidos con WITH FUNCTION ACCESS también se pueden invocar utilizando la notación normal de funciones. La resolución de la función considera como aceptables todas las funciones, así como los métodos con acceso a función. Sin embargo, las funciones no se pueden invocar utilizando la invocación de método. La resolución del método considera aceptables todos los métodos, pero no las funciones. Si la resolución no proporciona una función o método apropiado, se produce un error (SQLSTATE 42884).

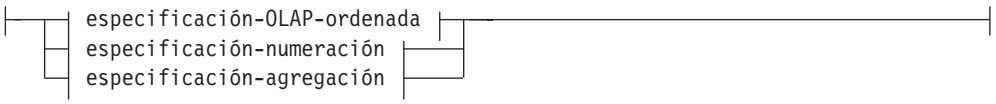
Ejemplo

- Este ejemplo utiliza el operador .. para invocar un método llamado AREA. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del tipo estructurado CIRCLE. Se supone también que el método AREA se ha definido previamente para el tipo CIRCLE como `AREA() RETURNS DOUBLE`.

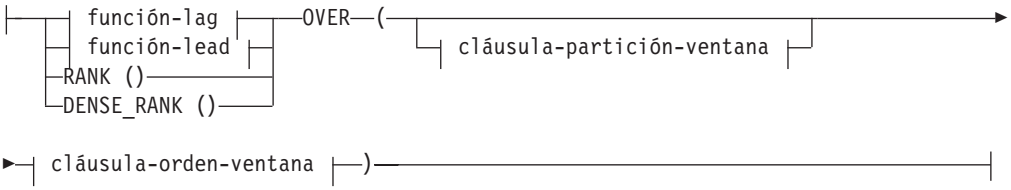
```
SELECT CIRCLE_COL..AREA() FROM RINGS
```

Especificaciones OLAP

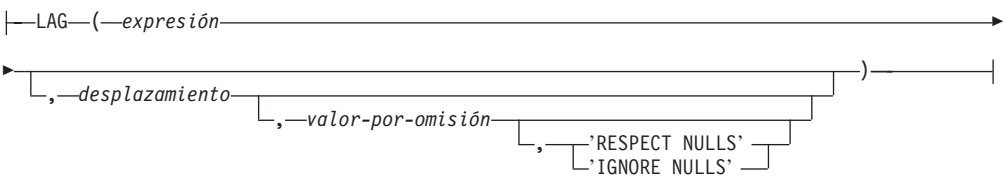
especificación-OLAP:



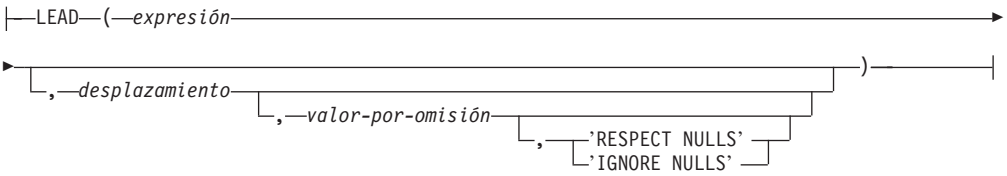
especificación-OLAP-ordenada:



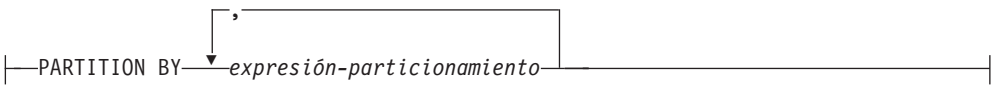
función-lag:



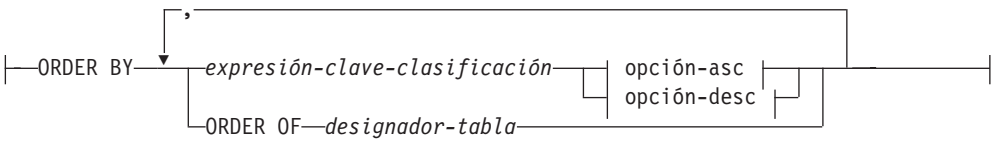
función-lead:



cláusula-partición-ventana:



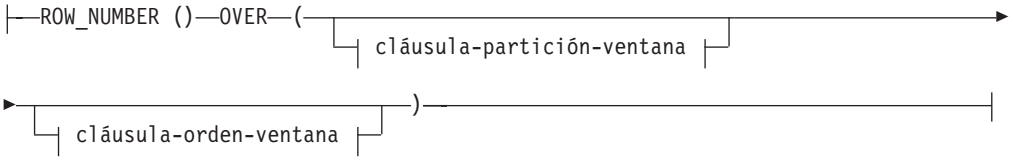
cláusula-orden-ventana:



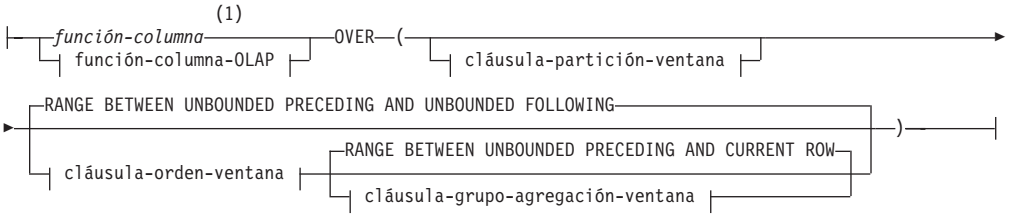
opción-asc:

opción-desc:

especificación-numeración:



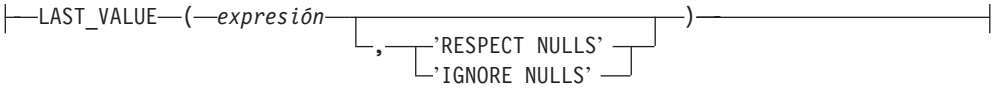
especificación-agregación:



función-columna-OLAP:

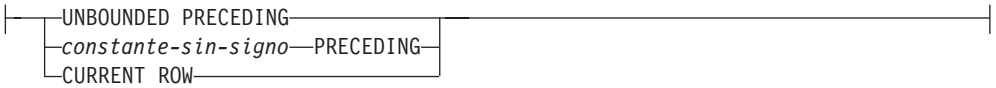
función-first-value:

función-last-value:

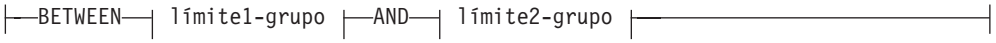


cláusula-grupo-agregación-ventana:

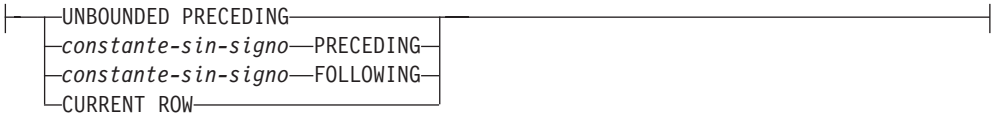
inicio-grupo:



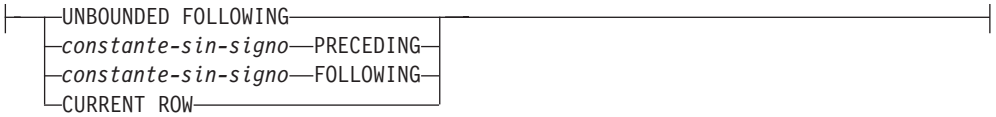
entre-grupo:



límite-grupo1:



límite-grupo2:



final-grupo:

Notas:

1 ARRAY_AGG no se soporta como función agregada en *especificación-agregación* (SQLSTATE 42887).

Las funciones OLAP (On-Line Analytical Processing) proporcionan la capacidad de devolver la ordenación, numeración de filas e información sobre funciones agregadas existentes, así como un valor escalar en el resultado de una consulta. Se puede incluir una función OLAP en expresiones, en una lista de selección o en la cláusula ORDER BY de una sentencia SELECT (SQLSTATE 42903). Una función OLAP no se puede utilizar en un argumento con una expresión XMLQUERY ni XMLEXISTS (SQLSTATE 42903). Una función OLAP no se puede utilizar como un argumento de una función agregada (SQLSTATE 42607). La función OLAP se aplica a la tabla resultante de la subselección más interna donde reside la función OLAP.

Cuando se utiliza una función OLAP, se especifica una ventana que define las filas a las que se aplica la función, y en qué orden. Si se utiliza con una función

agregada, las filas aplicables se pueden refinar más, con relación a la fila actual, como un rango o como un número de filas que preceden y siguen a la fila actual. Por ejemplo, dentro de una división por meses, se puede calcular un valor promedio respecto a los tres meses anteriores.

La función de ordenación calcula la posición ordinal de una fila dentro de la ventana. Las filas que no son distintas con respecto a la ordenación dentro de sus ventanas tienen asignada la misma posición. Los resultados de la ordenación se pueden definir con o sin huecos en los números que resultan de valores duplicados.

Si se especifica RANK, la posición de una fila se define como 1 más el número de filas que preceden estrictamente a la fila. Por lo tanto, si dos o más filas no difieren con respecto a la ordenación, habrá uno o más huecos en la numeración jerárquica secuencial.

Si se especifica DENSE_RANK (o DENSERANK), el rango de una fila se define como 1 más el número de filas que la preceden que son distintas respecto a la ordenación. Por tanto, no habrá huecos en la numeración jerárquica secuencial.

La función ROW_NUMBER (o ROWNUMBER) calcula el número secuencial de la fila dentro de la ventana definida por la ordenación, empezando por 1 para la primera fila. Si la cláusula ORDER BY no está especificada en la ventana, los números de fila se asignan a las filas en un orden arbitrario, tal como son devueltas por la subselección (no de acuerdo con ninguna cláusula ORDER BY de la sentencia-select).

Si se utiliza la cláusula FETCH FIRST *n* ROWS ONLY junto con la función ROW_NUMBER, es posible que el número de filas no se visualice en orden. La cláusula FETCH FIRST se aplica después de que se haya generado el conjunto de resultados (incluidas las asignaciones ROW_NUMBER); por lo tanto, si el orden del número de filas no es el mismo que el orden del conjunto de resultados, es posible que falten algunos números asignados en la secuencia.

El tipo de datos del resultado de RANK, DENSE_RANK o ROW_NUMBER es BIGINT. El resultado no puede ser nulo.

La función LAG devuelve el valor de expresión para la fila situada en el *desplazamiento* filas antes de la fila actual. El *desplazamiento* debe ser un entero positivo (SQLSTATE 42815). Un valor *desplazamiento* de 0 significa la fila actual. Si se especifica una cláusula-partición-ventana, *desplazamiento* significa filas *desplazamiento* antes de la fila actual y dentro de la partición actual. Si no se especifica *desplazamiento*, se utiliza el valor 1. Si se especifica *valor-por-omisión* (que puede ser una expresión), éste se devolverá si el desplazamiento va más allá del ámbito de la partición actual. De lo contrario, se devolverá el valor nulo. Si se especifica 'IGNORE NULLS', no se considera en el cálculo ninguna de las filas donde el valor de expresión para la fila es el valor nulo. Si se especifica 'IGNORE NULLS' y todas las filas son nulas, se devuelve *valor-por-omisión* (o el valor nulo si no se ha especificado *valor-por-omisión*).

La función LEAD devuelve el valor de expresión para la fila situada en el *desplazamiento* filas después de la fila actual. El *desplazamiento* debe ser un entero positivo (SQLSTATE 42815). Un valor *desplazamiento* de 0 significa la fila actual. Si se especifica una cláusula-partición-ventana, *desplazamiento* significa filas *desplazamiento* después de la fila actual y dentro de la partición actual. Si no se especifica *desplazamiento*, se utiliza el valor 1. Si se especifica *valor-por-omisión* (que

puede ser una expresión), éste se devolverá si el desplazamiento va más allá del ámbito de la partición actual. De lo contrario, se devolverá el valor nulo. Si se especifica 'IGNORE NULLS', no se considera en el cálculo ninguna de las filas donde el valor de expresión para la fila es el valor nulo. Si se especifica 'IGNORE NULLS' y todas las filas son nulas, se devuelve *valor-por-omisión* (o el valor nulo si no se ha especificado *valor-por-omisión*).

La función FIRST_VALUE devuelve el valor de expresión para la primera fila en una ventana OLAP. Si se especifica 'IGNORE NULLS', no se considera en el cálculo ninguna de las filas donde el valor de expresión para la fila es el valor nulo. Si se especifica 'IGNORE NULLS' y todos los valores de la ventana OLAP son nulos, FIRST_VALUE devuelve el valor nulo.

La función LAST_VALUE devuelve el valor de expresión para la última fila de una ventana OLAP. Si se especifica 'IGNORE NULLS', no se considera en el cálculo ninguna de las filas donde el valor de expresión para la fila es el valor nulo. Si se especifica 'IGNORE NULLS' y todos los valores de la ventana OLAP son nulos, LAST_VALUE devuelve el valor nulo.

El tipo de datos del resultado de FIRST_VALUE, LAG, LAST_VALUE y LEAD es el tipo de datos de la expresión. El resultado puede ser nulo.

PARTITION BY (*expresión-particionamiento***,...)**
Define la partición que se utiliza para aplicar la función. Una *expresión-particionamiento* es una expresión utilizada para definir el particionamiento del conjunto resultante. Cada *nombre-columna* referenciado en una *expresión-particionamiento* debe hacer referencia sin ambigüedades a una columna de la tabla de resultados de la subselección que contiene la especificación OLAP (SQLSTATE 42702 ó 42703). Una *expresión-particionamiento* no puede incluir una selección completa escalar o una expresión XMLQUERY o XMLEXISTS (SQLSTATE 42822) ni tampoco ninguna función o consulta que no sea determinista o que tenga una acción externa (SQLSTATE 42845).

cláusula-orden-ventana

ORDER BY (*expresión-clave-clasificación***,...)**
Define la ordenación de las filas dentro de una partición que determina el valor de la función OLAP o el significado de los valores de fila en la cláusula-grupo-agregación-ventana (no define la ordenación del conjunto resultante de la consulta).

expresión-clave-clasificación
Una expresión utilizada para definir la ordenación de las filas dentro de una partición de ventana. Cada nombre de columna referenciado en una *expresión-clave-clasificación* debe identificar, sin ambigüedades, una columna del conjunto resultante de la subselección, incluida la función OLAP (SQLSTATE 42702 ó 42703). Una *expresión-clave-clasificación* no puede incluir una selección completa escalar ni una expresión XMLQUERY o XMLEXISTS (SQLSTATE 42822) ni cualquier función o consulta que no sea determinante o que tenga una acción externa (SQLSTATE 42845). Esta cláusula es necesaria para las funciones RANK y DENSE_RANK (SQLSTATE 42601).

ASC
Utiliza los valores de la expresión-clave-clasificación en orden ascendente.

DESC
Utiliza los valores de la expresión-clave-clasificación en orden descendente.

NULLS FIRST
La ordenación de la ventana tiene en cuenta los valores nulos *antes* de todos los valores no nulos en el orden de clasificación.

NULLS LAST
La ordenación de la ventana tiene en cuenta los valores nulos *después* de todos los valores no nulos en el orden de clasificación.

ORDER OF *designador-tabla*
Especifica que debe aplicarse el mismo orden utilizado en *diseñador-tabla* a la tabla resultante de la subselección. Debe haber una referencia de tabla que se corresponda con *diseñador-tabla* en la cláusula FROM de la subselección que especifica esta cláusula (SQLSTATE 42703). La subselección (o selección completa) correspondiente al *diseñador-tabla* especificado debe incluir una cláusula ORDER BY que dependa de los datos (SQLSTATE 428FI). El orden que se aplica es el mismo que si las columnas de la cláusula ORDER BY de la subselección anidada (o selección completa) se incluyeran en la subselección exterior (o selección completa) y estas columnas se especificaran en lugar de la cláusula ORDER OF.

cláusula-grupo-agregación-ventana
El grupo de agregación de una fila R es un conjunto de filas definidas en relación a R (en la ordenación de las filas de la partición de R). Esta cláusula especifica el grupo de agregación. Si no se especifica esta cláusula y tampoco se especifica una cláusula-orden-ventana, el grupo de agregación consta de todas las filas de la partición de ventana. Este valor por omisión se puede especificar explícitamente utilizando RANGE (como se muestra) o ROWS.

Si se especifica una cláusula-orden-ventana, el comportamiento por omisión es diferente cuando no se especifica cláusula-grupo-agregación-ventana. El grupo de agregación de ventana consta de todas las filas de la partición R que preceden a R o que son iguales a R en la clasificación de ventanas de la partición de ventanas definida por la cláusula-orden-ventana.

ROWS
Indica que el grupo de agregación se define mediante el recuento de filas.

RANGE
Indica que el grupo de agregación se define mediante un valor de desplazamiento con respecto a una clave de clasificación.

inicio-grupo
Especifica el punto de inicio del grupo de agregación. El final del grupo de agrupación es la fila actual. La cláusula inicio-grupo es equivalente a una cláusula entre-grupo en la forma ″BETWEEN inicio-grupo AND CURRENT ROW″.

entre-grupo
Especifica el inicio y final del grupo de agregación basándose en ROWS o RANGE.

final-grupo
Especifica el punto final del grupo de agregación. El inicio del grupo de agregación es la fila actual. La especificación de la cláusula final-grupo es equivalente a la de una cláusula entre-grupo del formato ″BETWEEN CURRENT ROW AND final-grupo″.

UNBOUNDED PRECEDING
Incluye la partición completa que precede a la fila actual. Esto se puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

UNBOUNDED FOLLOWING
Incluye la partición completa que sigue a la fila actual. Esto se puede especificar con ROWS o RANGE. También se puede especificar con varias expresiones-clave-clasificación en la cláusula-orden-ventana.

CURRENT ROW
Especifica el inicio o el final del grupo de agregación basándose en la fila actual. Si se especifica ROWS, la fila actual es el límite del grupo de agregación. Si se especifica RANGE, el límite del grupo de agregación incluye el conjunto de filas con los mismos valores para las *expresiones-clave-clasificación* que la fila actual. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* especifica el *valor* FOLLOWING.

valor **PRECEDING**
Especifica el rango o número de filas que preceden a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de resta. Esta cláusula no se puede especificar en *límite-grupo2* si *límite-grupo1* es CURRENT ROW o *valor* FOLLOWING.

valor **FOLLOWING**
Especifica el rango o número de filas que siguen a la fila actual. Si se especifica ROWS, *valor* es un entero positivo que indica un número de filas. Si se especifica RANGE, el tipo de datos de *valor* debe ser comparable con el tipo de la expresión-clave-clasificación de la cláusula-orden-ventana. Sólo puede haber una sola expresión-clave-clasificación y el tipo de datos de esa expresión debe permitir la operación de suma.

Ejemplos

- Este ejemplo muestra la ordenación de los empleados, dispuestos por apellidos, de acuerdo con un salario total (salario más prima) que sea mayor que $30.000.

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
  FROM EMPLOYEE WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

  Observe que si el resultado debe estar ordenado de acuerdo con la escala de salarios, debe sustituir ORDER BY LASTNAME por:

```
ORDER BY RANK_SALARY
```

  o bien

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- Este ejemplo ordena los departamentos de acuerdo con su salario total medio.

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
    RANK() OVER (ORDER BY AVG(SALARY+BONUS) DESC) AS RANK_AVG_SAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY RANK_AVG_SAL
```

- Este ejemplo ordena los empleados de un departamento de acuerdo con su nivel de formación. Si varios empleados de un departamento tienen el mismo nivel, ello no debe suponer un aumento del nivel siguiente.

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL,
    DENSE_RANK() OVER
      (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- Este ejemplo proporciona números a las filas del resultado de una consulta.

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
    LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- Este ejemplo lista los cinco empleados con mayores ingresos.

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
  FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
    FROM EMPLOYEE) AS RANKED_EMPLOYEE
  WHERE RANK_SALARY < 6
  ORDER BY RANK_SALARY
```

  Observe que primero se ha utilizado una expresión de tabla anidada para calcular el resultado, incluidos los niveles de ordenación, para poder utilizar el nivel en la cláusula WHERE. También se podría haber utilizado una expresión de tabla común.

- Para cada departamento, haga una lista de los salarios de los empleados y muestre cuánto gana de menos cada persona en comparación con el empleado del mismo departamento con el siguiente salario más alto.

```
SELECT EMPNO, WORKDEPT, LASTNAME, FIRSTNME, JOB, SALARY,
    LEAD(SALARY, 1) OVER (PARTITION BY WORKDEPT
      ORDER BY SALARY) - SALARY AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, SALARY
```

- Calcule el salario de un empleado relativo al salario del empleado que en primer lugar se contrató para el mismo tipo de trabajo.

```
SELECT JOB, HIREDATE, EMPNO, LASTNAME, FIRSTNME, SALARY,
    FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS FIRST_SALARY,
    SALARY - FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY JOB, HIREDATE
```

- Calcule el precio de cierre medio de las acciones XYZ durante el mes de junio de 2006. Si las acciones no se negocian en un día determinado, su precio de cierre en la tabla DAILYSTOCKDATA es el valor de nulo. En vez de devolver el valor de nulo para los días en que no se negocian las acciones, utilice la función COALESCE y la función LAG para devolver el precio de cierre del día más reciente en que se negociaron las acciones. Limite la búsqueda de un valor de cierre anterior que no sea de nulo a un mes antes del día uno de enero de 2006.

```
WITH V1(SYMBOL, TRADINGDATE, CLOSEPRICE) AS
(
SELECT SYMBOL, TRADINGDATE,
  COALESCE(CLOSEPRICE,
    LAG(CLOSEPRICE,
      1,
      CAST(NULL AS DECIMAL(8,2)),
      'IGNORE NULLS')
    OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE)
)
FROM DAILYSTOCKDATA
WHERE SYMBOL = 'XYZ' AND
  TRADINGDATE BETWEEN '2005-12-01' AND '2006-01-31'
```

```
)
SELECT SYMBOL, AVG(CLOSEPRICE) AS AVG
  FROM V1
  WHERE TRADINGDATE BETWEEN '2006-01-01' AND '2006-01-31'
  GROUP BY SYMBOL
```

- Calcule el promedio variable de 30 días de las acciones ABC y XYZ durante el año 2005.

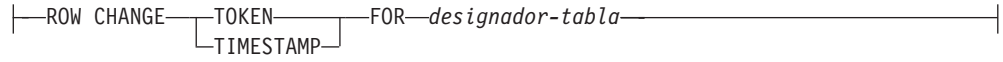
```
WITH V1(SYMBOL, TRADINGDATE, MOVINGAVG30DAY) AS
(
SELECT SYMBOL, TRADINGDATE,
  AVG(CLOSEPRICE) OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE
  ROWS BETWEEN 29 PRECEDING AND CURRENT ROW)
FROM DAILYSTOCKDATA
WHERE SYMBOL IN ('ABC', 'XYZ')
  AND TRADINGDATE BETWEEN DATE('2005-01-01') - 2 MONTHS
    AND '2005-12-31'
)
SELECT SYMBOL, TRADINGDATE, MOVINGAVG30DAY
  FROM V1
  WHERE TRADINGDATE BETWEEN '2005-01-01' AND '2005-12-31'
  ORDER BY SYMBOL, TRADINGDATE
```

- Utilice una expresión para definir la posición del cursor y consultar una ventana deslizante de 50 filas antes de dicha posición.

```
SELECT DATE, FIRST_VALUE(CLOSEPRICE + 100) OVER
    (PARTITION BY SYMBOL
    ORDER BY DATE
    ROWS BETWEEN 50 PRECEDING AND 1 PRECEDING) AS FV
  FROM DAILYSTOCKDATA
  ORDER BY DATE
```

Expresión ROW CHANGE

expresión-cambio-fila:



Una expresión ROW CHANGE devuelve un símbolo o una indicación de fecha y hora que representa el último cambio de una fila.

TOKEN
Especifica que se devuelve un valor BIGINT que representa un punto relativo en el orden de modificación de una fila. Si no se ha cambiado la fila, el resultado es un símbolo que representa cuándo se insertó el valor inicial. El resultado puede ser nulo. ROW CHANGE TOKEN no es determinante.

TIMESTAMP
Especifica que se devuelve un valor TIMESTAMP que representa la última vez que se cambió una fila. Si no se ha cambiado la fila, el resultado es la hora en la que se insertó el valor inicial. El resultado puede ser nulo. ROW CHANGE TIMESTAMP no es determinante.

FOR *designador-tabla*
Identifica la tabla en la que se hace referencia a la expresión. El *designador-tabla* debe identificar de manera exclusiva una tabla base, una vista o una expresión de tabla anidada (SQLSTATE 42867). Si *designador-tabla* identifica una vista o una expresión de tabla anidada, la expresión ROW CHANGE devuelve el TOKEN o la TIMESTAMP de la tabla base de la vista o expresión de tabla anidada. La vista o expresión de tabla anidada debe contener sólo una tabla base en su subselección exterior (SQLSTATE 42867). Si el *designador-tabla* es una expresión de tabla anidada o vista, debe ser suprimible (SQLSTATE 42703). Para obtener más información sobre vistas suprimibles, consulte el apartado "Notas" o "CREATE VIEW". El designador de tabla de una expresión ROW CHANGE TIMESTAMP debe resolverse en una tabla base que contenga una columna de indicación de fecha y hora de cambio de fila (SQLSTATE 55068).

Notas

- Las aplicaciones que utilizan un bloqueo optimista pueden utilizar con la función escalar RID_BIT los valores devueltos por la expresión ROW CHANGE TOKEN.

Ejemplos

- Devuelva un valor de indicación de fecha y hora que se corresponda con el cambio más reciente de cada fila de la tabla EMPLOYEE para los empleados del departamento 20. Suponga que se ha modificado la tabla EMPLOYEE para que contenga una columna definida con la cláusula ROW CHANGE TIMESTAMP.

```
SELECT ROW CHANGE TIMESTAMP FOR EMPLOYEE
  FROM EMPLOYEE WHERE DEPTNO = 20
```

- Devuelva un valor BIGINT que represente un punto relativo en el orden de modificación de la fila correspondiente al empleado número 3.500. Devuelva también el valor de la función escalar RID_BIT que se va a utilizar en un escenario DELETE de bloqueo optimista. Especifique la opción WITH UR para obtener el último valor ROW CHANGE TOKEN.

```
SELECT ROW CHANGE TOKEN FOR EMPLOYEE, RID_BIT (EMPLOYEE)
  FROM EMPLOYEE WHERE EMPNO = '3500' WITH UR
```

La sentencia anterior es satisfactoria tanto si hay una columna de indicación de fecha y hora de cambio de fila en la tabla EMPLOYEE como si no la hay. La siguiente sentencia buscada DELETE suprime la fila especificada por los valores ROW CHANGE TOKEN y RID_BIT de la anterior sentencia SELECT, suponiendo que los dos valores de marcador de parámetro estén definidos como los valores obtenidos por la sentencia anterior.

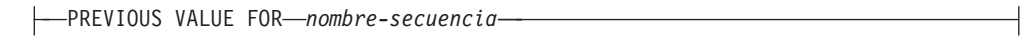
```
DELETE FROM EMPLOYEE E
  WHERE RID_BIT (E) = ? AND ROW CHANGE TOKEN FOR E = ?
```

Referencia de secuencia

referencia-secuencia:

expresión-nextval:

expresión-prevval:



NEXT VALUE FOR *nombre-secuencia*
Una expresión NEXT VALUE genera y devuelve el siguiente valor de la secuencia especificada por *nombre-secuencia*.

PREVIOUS VALUE FOR *nombre-secuencia*
Una expresión PREVIOUS VALUE devuelve el valor generado más recientemente de la secuencia especificada para una sentencia anterior del proceso de aplicación actual. Se puede hacer referencia a este valor repetidamente utilizando expresiones PREVIOUS VALUE que especifican el nombre de la secuencia. Pueden existir múltiples instancias de las expresiones PREVIOUS VALUE especificando el mismo nombre de secuencia en una sola sentencia; todas ellas devuelven el mismo valor. En un entorno de bases de datos particionadas, es posible que una expresión PREVIOUS VALUE no devuelva el valor generado más recientemente.

Una expresión PREVIOUS VALUE sólo se puede utilizar si ya se ha hecho referencia a una expresión NEXT VALUE que especifica el mismo nombre de secuencia en el proceso de aplicación actual, ya sea en la transacción actual ya sea en una transacción anterior (SQLSTATE 51035).

Notas

- Se genera un valor nuevo para una secuencia cuando la expresión NEXT VALUE especifica el nombre de dicha secuencia. Sin embargo, si existen múltiples instancias de una expresión NEXT VALUE que especifican el mismo nombre de secuencia en una consulta, el contador para la secuencia se incrementa sólo una vez para cada fila del resultado y todas las instancias de NEXT VALUE devuelven el mismo valor para una fila del resultado.
- Se puede utilizar el mismo número de secuencia como valor de clave de unicidad en dos tablas independientes haciendo referencia al número de secuencia con una expresión NEXT VALUE para la primera fila (esto genera el valor de secuencia) y una expresión PREVIOUS VALUE para las demás filas (la instancia de PREVIOUS VALUE hace referencia al valor de secuencia generado más recientemente en la sesión actual), tal como se muestra a continuación:

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item  (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- Las expresiones NEXT VALUE y PREVIOUS VALUE pueden especificarse en los lugares siguientes:
  - sentencia-select o sentencia SELECT INTO (en la cláusula-select, a condición de que la sentencia no contenga una palabra clave DISTINCT, una cláusula GROUP BY, una cláusula ORDER BY, una palabra clave UNION, una palabra clave INTERSECT o una palabra clave EXCEPT)
  - sentencia INSERT (en una cláusula VALUES)
  - sentencia INSERT (en la cláusula-select de la selección completa (fullselect))
  - sentencia UPDATE (en la cláusula SET (una sentencia UPDATE buscada o colocada), excepto que no se puede especificar NEXT VALUE en la cláusula-select de la selección completa de una expresión de la cláusula SET)
  - sentencia SET variable (excepto en la cláusula-select de la selección completa de una expresión; una expresión NEXT VALUE puede especificarse en un activador, pero una expresión PREVIOUS VALUE no puede especificarse)
  - sentencia VALUES INTO (en la cláusula-select de la selección completa (fullselect) de una expresión)
  - sentencia CREATE PROCEDURE (en el cuerpo-rutina de un procedimiento SQL)
  - sentencia CREATE TRIGGER en la acción-activada (se puede especificar una expresión NEXT VALUE, pero no se puede especificar una expresión PREVIOUS VALUE)
- Las expresiones NEXT VALUE y PREVIOUS VALUE no se pueden especificar (SQLSTATE 428F9) en los lugares siguientes:
  - Las condiciones de unión de una unión externa completa
  - El valor DEFAULT de una columna en una sentencia CREATE o ALTER TABLE
  - La definición de columna generada en una sentencia CREATE o ALTER TABLE
  - La definición de tabla de resumen de una sentencia CREATE TABLE o ALTER TABLE
  - La condición de una restricción CHECK
  - Sentencia CREATE TRIGGER (se puede especificar una expresión NEXT VALUE, pero no se puede especificar una expresión PREVIOUS VALUE)
  - CREATE VIEW, sentencia
  - CREATE METHOD, sentencia
  - Sentencia CREATE FUNCTION
  - Una lista de argumentos de una expresión XMLQUERY, XMLEXISTS o XMLTABLE
- Además, no se puede especificar una expresión NEXT VALUE (SQLSTATE 428F9) en los lugares siguientes:
  - Expresión CASE
  - La lista de parámetros de una función agregada
  - La subconsulta en un contexto distinto de los explícitamente permitidos mencionados anteriormente
  - La sentencia SELECT para la que la SELECT externa contiene un operador DISTINCT
  - La condición de unión de una unión
  - La sentencia SELECT para la que la SELECT externa contiene una cláusula GROUP BY

  - La sentencia SELECT para la que la SELECT externa está combinada con otra sentencia SELECT utilizando el operador establecido UNION, INTERSECT o EXCEPT
  - Una expresión de tabla anidada
  - La lista de parámetros de una función de tabla
  - La cláusula WHERE de la sentencia SELECT más externa o una sentencia DELETE o UPDATE
  - La cláusula ORDER BY de la sentencia SELECT más externa
  - La cláusula-select de la selección completa (fullselect) de una expresión, en la cláusula SET de una sentencia UPDATE
  - La sentencia IF, WHILE, DO ... UNTIL o CASE de una rutina SQL
- Cuando se genera un valor para una secuencia, se consume dicho valor y, la siguiente vez que se solicita un valor, se genera un valor nuevo. Esto es válido incluso cuando la sentencia que contiene la expresión NEXT VALUE falla o se retrotrae.

  Si una sentencia INSERT incluye una expresión NEXT VALUE en la lista VALUES para la columna y si se produce un error en algún punto durante la ejecución de INSERT (puede ser un problema al generar el siguiente valor de secuencia o un problema con el valor de otra columna), se produce una anomalía de inserción (SQLSTATE 23505) y se considera que el valor generado para la secuencia se ha consumido. En algunos casos, al volver a emitir la misma sentencia INSERT se puede obtener un resultado satisfactorio.

  Por ejemplo, considere un error que es el resultado de la existencia de un índice de unicidad para la columna para la que se ha utilizado NEXT VALUE y el valor de secuencia generado ya existe en el índice. Es posible que el siguiente valor generado para la secuencia sea un valor que no existe en el índice y, por consiguiente, el INSERT subsiguiente dará un resultado satisfactorio.
- ***Ámbito de PREVIOUS VALUE:*** El valor de PREVIOUS VALUE se mantiene hasta que el valor siguiente se genera para la secuencia en la sesión actual, la secuencia se descarta o se modifica, o hasta que finaliza la sesión de la aplicación. El valor no se ve afectado por las sentencias COMMIT o ROLLBACK. El valor de PREVIOUS VALUE no puede establecerse directamente y es el resultado de ejecutar la expresión NEXT VALUE para la secuencia.

  Una técnica utilizada habitualmente, sobre todo para cuestiones de rendimiento, es que una aplicación o producto gestione un conjunto de conexiones y direccione transacciones a una conexión arbitraria. En estas situaciones, la disponibilidad de PREVIOUS VALUE para una secuencia solamente debería ser dependiente hasta que finalice la transacción. Algunos ejemplos de dónde puede producirse este tipo de situación incluyen aplicaciones que utilicen protocolos XA, usen la agrupación de conexiones, empleen el concentrador de conexiones y utilicen HADR para lograr la migración tras error.
- Si al generar un valor para una secuencia se excede el valor máximo para la secuencia (o el valor mínimo para una secuencia descendente) y no se permiten ciclos, se producirá un error (SQLSTATE 23522). En este caso, el usuario puede modificar (ALTER) la secuencia para ampliar el rango de valores aceptables, habilitar ciclos para la secuencia o eliminar (DROP) la secuencia y crear (CREATE) una nueva con un tipo de datos diferente que tenga un mayor rango de valores.

  Por ejemplo, una secuencia puede haberse definido con un tipo de datos de SMALLINT y, finalmente, la secuencia se queda sin valores asignables. Elimine (DROP) y vuelva a crear la secuencia con la nueva definición para volver a definir la secuencia como INTEGER.

- Una referencia a una expresión NEXT VALUE en la sentencia de selección (select) de un cursor hace referencia a un valor que se genera para una fila de la tabla resultante. Se genera un valor de secuencia para una expresión NEXT VALUE para cada fila que se busca desde la base de datos. Si se realiza el bloqueo en el cliente, puede que los valores se hayan generado en el servidor antes del proceso de la sentencia FETCH. Esto puede producirse cuando existe bloqueo de las filas de la tabla resultante. Si la aplicación cliente no capta (FETCH) explícitamente todas las filas que la base de datos ha materializado, la aplicación no verá los resultados de todos los valores de secuencia generados (para las filas materializadas que no se ha devuelto).
- Una referencia a una expresión PREVIOUS VALUE de la sentencia de selección (select) de un cursor hace referencia a un valor que se ha generado para la secuencia especificada antes de la apertura del cursor. Sin embargo, el cierre del cursor puede afectar a los valores devueltos por PREVIOUS VALUE para la secuencia especificada en las sentencias futuras o incluso para la misma sentencia en el caso de que se vuelva a abrir el cursor. Esto sucederá cuando la sentencia de selección del cursor incluya una referencia a NEXT VALUE para el mismo nombre de secuencia.
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - Se pueden especificar NEXTVAL y PREVVAL en lugar de NEXT VALUE y PREVIOUS VALUE
  - Para mantener la compatibilidad con IBM IDS:
    - Se puede especificar *nombre-secuencia*.NEXTVAL en lugar de NEXT VALUE FOR *nombre-secuencia*
    - Se puede especificar *nombre-secuencia*.CURRVAL en lugar de PREVIOUS VALUE FOR *nombre-secuencia*

Ejemplos

Supongamos que existe una tabla llamada ″order″ y que se crea una secuencia llamada ″order_seq″ del modo siguiente:

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

A continuación se muestran algunos ejemplos de cómo generar un número de secuencia ″order_seq″ con una expresión NEXT VALUE:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

o bien

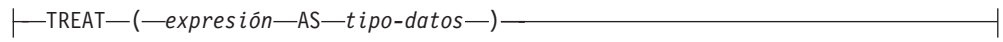
```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

o bien

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

Tratamiento de los subtipos

tratamiento-subtipo:



El *tratamiento-subtipo* se utiliza para convertir una expresión de tipo estructurado en uno de sus subtipos. El tipo estático de *expresión* debe ser un tipo estructurado definido por el usuario, y ese tipo debe ser el mismo que *tipo-datos* o que un subtipo de él. Si el nombre de tipo especificado en *tipo-datos* no está calificado, se utiliza la vía de acceso de SQL para resolver la referencia al tipo. El tipo estático del resultado de tratamiento-subtipo es *tipo-datos*, y el valor del tratamiento-subtipo es el valor de la expresión. Durante la ejecución, si el tipo dinámico de la expresión no es *tipo-datos* o un subtipo de *tipo-datos*, se produce un error (SQLSTATE 0D000).

Ejemplo

- En este ejemplo, todas las instancias de objetos de la columna CIRCLE_COL están definidas con el tipo dinámico COLOREDCIRCLE para una aplicación. Se utiliza la consulta siguiente para invocar el método RGB para tales objetos. Se supone que existe una tabla llamada RINGS, con una columna CIRCLE_COL del tipo estructurado CIRCLE. Se supone también que COLOREDCIRCLE es un subtipo de CIRCLE y que el método RGB se ha definido previamente para COLOREDCIRCLE como `RGB() RETURNS DOUBLE`.

  ```
  SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
    FROM RINGS
  ```

  Durante la ejecución, si hay instancias del tipo dinámico CIRCLE, se produce un error (SQLSTATE 0D000). Este error se puede evitar utilizando el predicado TYPE en una expresión CASE, del modo siguiente:

  ```
  SELECT (CASE
    WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
      THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
      ELSE NULL
    END)
    FROM RINGS
  ```

Determinación de los tipos de datos de las expresiones sin tipo

Una expresión sin tipo hace referencia al uso de un marcador de parámetro o valor nulo que se especifique sin un tipo de datos de destino asociado.

Las expresiones sin tipo pueden utilizarse en sentencias de SQL siempre que se dé una de las condiciones siguientes:

- Se está ejecutando una sentencia PREPARE para compilar la sentencia de SQL; la interfaz de cliente está utilizando la preparación diferida; y la variable de registro, DB2_DEFERRED_PREPARE_SEMANTICS, se establece en YES. En este caso, los marcadores de parámetro sin tipo derivan su tipo de datos según el descriptor de entrada asociado con la sentencia posterior OPEN o EXECUTE. El atributo de longitud se establece en la longitud máxima según la fila UNTYPED, tal y como se describe en la Tabla 21 en la página 211 de "Funciones" y según la longitud que se determina en las tablas siguientes. En cuanto a los tipos de datos que no están listados como tipo de destino en la Tabla 21 en la página 211 de "Funciones", se utilizará la longitud del descriptor de entrada asociado con la sentencia OPEN o EXECUTE posterior. Los tipos de archivo y las longitudes pueden modificarse de acuerdo con el uso del marcador de parámetro sin tipo de la sentencia de SQL.
- El tipo de datos puede determinarse según el contexto en la sentencia de SQL. Estas ubicaciones y los tipos de datos resultantes se muestran en la tabla siguiente. Las ubicaciones se agrupan en expresiones, predicados, funciones incorporadas y rutinas definidas por el usuario para ayudar en la determinación de la aplicabilidad de una expresión sin tipo. El tipo de datos no puede determinarse según el contexto; se emite un error.

En algunos casos que no se enumeran, las expresiones sin tipo de una lista de selección se resolverán en un tipo de datos determinado según el uso en la sentencia de SQL.

La página de códigos de la expresión sin tipo queda determinada por el contexto. Donde no haya ningún contexto, la página de códigos es la misma que si la expresión sin tipo se hubiera convertido en un tipo de datos VARCHAR.

Tabla 24. Uso de expresiones sin tipo en expresiones (incluida la lista de selección, CASE y VALUES)

Ubicación de la expresión sin tipo	Tipo de datos
Solo en una lista de selección	Error si la expresión sin tipo no tiene nombre o tiene nombre, pero no se ha referenciado posteriormente en la sentencia de SQL. Si la expresión sin tipo tiene nombre y se referencia posteriormente en la sentencia de SQL, el tipo de datos puede determinarse a partir del uso posterior. Para obtener más información, consulte la nota ″Determinación del tipo de datos a partir del uso″ a continuación de esta tabla.
Ambos operandos de un solo operador aritmético, después de considerar la prioridad de los operadores y el orden de las normas de la operación Incluye casos como: `(? + ?) + 10`	DECFLOAT(34)

Tabla 24. Uso de expresiones sin tipo en expresiones (incluida la lista de selección, CASE y VALUES) (continuación)

Ubicación de la expresión sin tipo	Tipo de datos
Un operando de un solo operador de una expresión aritmética (no una expresión de indicación de fecha y hora) Incluye casos como: `? + (? * 10)`	El tipo de datos del otro operando.
Duración etiquetada dentro de una expresión de indicación de fecha y hora. (Observe que la parte de una duración etiquetada que indica el tipo de unidades no puede ser un marcador de parámetro).	DECIMAL(15,0)
Cualquier otro operando de una expresión de fecha y hora (por ejemplo, 'timecol + ?' o '? - datecol')	Error
Ambos operandos de un operador CONCAT	VARCHAR (254)
Un operando de un operador CONCAT cuando el otro operando es un tipo de datos de caracteres que no es CLOB	Si un operando es CHAR(*n*) o VARCHAR(*n*), donde *n* es menor que 128, el otro es VARCHAR(254 - *n*); en el resto de casos, el tipo de datos es VARCHAR(254)
Un operando de un operador CONCAT cuando el otro operando es un tipo de datos gráfico que no es DBCLOB	Si un operando es GRAPHIC(*n*) o bien VARGRAPHIC(*n*), donde *n* es menor que 64, el otro valor es VARGRAPHIC(127 - *n*); en todos los demás casos, el tipo de datos es VARGRAPHIC(127).
Un operando de un operador CONCAT cuando el otro operando es una serie de gran objeto	Igual que el del otro operando
La expresión que está a continuación de la palabra clave CASE en una expresión CASE simple	El resultado de la aplicación de las "normas para los tipos de datos de resultados" a las expresiones que siguen a la palabra clave WHEN que no son expresiones sin tipo.
Como mínimo, una de las expresiones de resultados de una expresión CASE (tanto simple como buscada), con el resto de las expresiones de resultados que sean expresiones sin tipo.	Error
Alguna o todas las expresiones que siguen a la palabra clave WHEN en una expresión CASE	El resultado de la aplicación de las "normas para los tipos de datos de resultados" a la expresión que sigue a CASE y a las expresiones que siguen a la palabra clave WHEN que no sean una expresión sin tipo
Una expresión de resultados de una expresión CASE (tanto simple como buscada), cuando, como mínimo, una expresión de resultados no es una expresión sin tipo	El resultado de la aplicación de las "normas para los tipos de datos de resultados" a todas las expresiones de resultados que no sean una expresión sin tipo

Tabla 24. Uso de expresiones sin tipo en expresiones (incluida la lista de selección, CASE y VALUES) (continuación)

Ubicación de la expresión sin tipo	Tipo de datos
Sola como una expresión de columna en una cláusula VALUES de una fila que no esté dentro de la sentencia INSERT ni en la cláusula VALUES de una operación de inserción de una sentencia MERGE	Error si la expresión sin tipo no tiene nombre o tiene nombre, pero no se ha referenciado posteriormente en la sentencia de SQL. Si la expresión sin tipo tiene nombre y se referencia posteriormente en la sentencia de SQL, el tipo de datos puede determinarse a partir del uso posterior. Para obtener más información, consulte la nota ″Determinación del tipo de datos a partir del uso″ a continuación de esta tabla.
Sola como expresión de columna en una cláusula VALUES de varias filas que no esté dentro de una sentencia INSERT y para la que las expresiones de columna de la misma posición en todas las demás expresiones de fila sean expresiones sin tipo	Error si la expresión sin tipo no tiene nombre o tiene nombre, pero no se ha referenciado posteriormente en la sentencia de SQL. Si la expresión sin tipo tiene nombre y se referencia posteriormente en la sentencia de SQL, el tipo de datos puede determinarse a partir del uso posterior. Para obtener más información, consulte la nota ″Determinación del tipo de datos a partir del uso″ a continuación de esta tabla.
Sola como expresión de columna en una cláusula VALUES de varias filas que no esté dentro de una sentencia INSERT y para la que las expresiones de la misma posición de, como mínimo, otra expresión de fila no sea una expresión sin tipo	El resultado de la aplicación de las “normas para los tipos de datos de resultados” en todos los operandos que no sean expresiones sin tipo
Solo como una expresión-columna en una cláusula VALUES de una sola fila dentro de una sentencia INSERT	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Solo como una expresión-columna en una cláusula VALUES de múltiples filas dentro de una sentencia INSERT	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Sola como una expresión de columna en una cláusula de valores de la tabla fuente para una sentencia MERGE	Error si la expresión sin tipo no tiene nombre o tiene nombre, pero no se ha referenciado posteriormente en la sentencia de SQL. Si la expresión sin tipo tiene nombre y se referencia posteriormente en la sentencia de SQL, el tipo de datos puede determinarse a partir del uso posterior. Para obtener más información, consulte la nota ″Determinación del tipo de datos a partir del uso″ a continuación de esta tabla.

Tabla 24. Uso de expresiones sin tipo en expresiones (incluida la lista de selección, CASE y VALUES) (continuación)

Ubicación de la expresión sin tipo	Tipo de datos
Sola como una expresión de columna en la cláusula VALUES de una operación de inserción de una sentencia MERGE	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Sola como expresión de columna a la derecha de la cláusula de asignación para una operación de actualización de la sentencia MERGE	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Sola como expresión de columna a la derecha de una cláusula SET en una sentencia UPDATE	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Como un valor a la derecha de una sentencia SET de registro especial	El tipo de datos del registro especial
Argumento de la cláusula TABLESAMPLE de la cláusula tablesample de una referencia de tabla	DOUBLE
Argumento de la subclase REPEATABLE de la cláusula tablesample de una referencia de tabla	INTEGER
Como valor de la sentencia FREE LOCATOR	Localizador
Como valor para la contraseña en una sentencia SET ENCRYPTION PASSWORD	VARCHAR(128)

Nota:

Determinación del tipo de datos a partir del uso

A continuación se muestra un ejemplo sobre cómo los tipos de datos de una expresión sin tipo pueden determinarse a partir del uso posterior:

Si se hace referencia a la expresión sin tipo y con nombre en un operador de comparación, tendrá, pues, el tipo de datos del otro operando. Si existen varias referencias de la expresión sin tipo y con nombre en la sentencia de SQL, el tipo de datos, la longitud, la precisión, la escala y la página de códigos que se determine de forma independiente para cada una de estas referencias deben ser idénticos o se devolverá un error.

Determinación de los tipos de datos de las expresiones sin tipo

Tabla 25. Uso de expresiones sin tipo en los predicados

Ubicación de la expresión sin tipo	Tipo de datos
Ambos operandos de un operador de comparación	VARCHAR (254)
Un operando de un operador de comparación, cuando el otro operando no es una expresión sin tipo	El tipo de datos del otro operando.
Todos los operandos del predicado BETWEEN	VARCHAR (254)
Dos operandos de un predicado BETWEEEN	El mismo que el de la única expresión con tipo
Sólo un operando de un predicado BETWEEN	El resultado de la aplicación de las “normas para los tipos de datos de resultados” en todos los operandos que no sean expresiones sin tipo
Todos los operandos de un predicado IN, por ejemplo, ? IN (?,?,?)	VARCHAR (254)
El primer operando de un predicado IN, cuando el lado derecho es una selección completa, por ejemplo, IN (fullselect)	El tipo de datos de la columna seleccionada
El primer operando de un predicado IN, cuando el lado derecho no es una subselección; por ejemplo, ? IN (?,A,B) o ? IN (A,?,B,?)	El resultado de la aplicación de las ″normas para los tipos de resultados″ en todos los operandos de la lista IN (operandos situados a la derecha de la palabra clave IN) que no sean expresiones sin tipo
Cualquier o todos los operandos de la lista IN del predicado IN, por ejemplo, A IN (?,B, ?)	El resultado de la aplicación de las ″normas para los tipos de resultados″ en todos los operandos del predicado IN (operandos situados a izquierda y derecha de la palabra clave IN) que no sean expresiones sin tipo
Tanto el operando de una expresión de valor de fila de un predicado IN como la columna de resultados correspondiente de la selección completa, por ejemplo (c1, ?) IN (SELECT c1, ? FROM ...)	VARCHAR(254)
Todos los operandos de una expresión-valor-fila en un predicado IN, por ejemplo, la selección completa (c1,?) IN	El tipo de datos de la columna de resultados correspondiente de la selección completa
Cualquier elemento de la lista de selección de una subconsulta si se especifica una expresión-valor-fila en un predicado IN, por ejemplo, (c1,c2) IN (SELECT?, c1, FROM ...)	El tipo de datos del operando correspondiente de la expresión-valor-fila
Los tres operandos del predicado LIKE	La expresión coincidente (operando 1) y la expresión patrón (operando 2) son VARCHAR(32672); la expresión de escape (operando 3) es VARCHAR(2)
La expresión coincidente del predicado LIKE cuando la expresión de patrón o la expresión de escape no es una expresión sin tipo	VARCHAR(32672) o VARGRAPHIC(16336), según el tipo de datos del primer operando que no sea una expresión sin tipo

Tabla 25. Uso de expresiones sin tipo en los predicados (continuación)

Ubicación de la expresión sin tipo	Tipo de datos
La expresión de patrón del predicado LIKE cuando la expresión coincidente o la expresión de escape no es una expresión sin tipo	VARCHAR(32672) o VARGRAPHIC(16336), según el tipo de datos del primer operando que no sea una expresión sin tipo; si el tipo de datos de la expresión coincidente es BLOB, el tipo de datos de la expresión de patrón se asume que es BLOB(32672)
La expresión de escape del predicado LIKE cuando la expresión coincidente o la expresión de patrón no es una expresión sin tipo	VARCHAR(2) o VARGRAPHIC(1), según el tipo de datos del primer operando que no sea una expresión sin tipo; si el tipo de datos de la expresión coincidente o la expresión de patrón es BLOB, el tipo de datos de la expresión de patrón se asume que es BLOB(1)
Operando del predicado NULL	VARCHAR (254)

Tabla 26. Uso de expresiones sin tipo en funciones incorporadas

Ubicación del marcador de parámetro sin tipo	Tipo de datos
Todos los argumentos de COALESCE, MIN, MAX, NULLIF o VALUE	Error
Cualquier argumento de COALESCE, MIN, MAX, NULLIF, o VALUE, cuando, como mínimo, un argumento no es un marcador de parámetro sin tipo	El resultado de la aplicación de las “normas para los tipos de datos de resultados” en todos los argumentos que no sean marcadores de parámetro sin tipo
Primer argumento de DAYNAME	TIMESTAMP(12)
Argumento de DIGITS	DECIMAL(31,6)
Primer argumento de MONTHNAME	TIMESTAMP(12)
POSSTR (ambos argumentos)	Los dos argumentos son VARCHAR(32672)
POSSTR (un argumento, cuando el otro argumento es un tipo de datos de caracteres)	VARCHAR(32672)
POSSTR (un argumento, cuando el otro argumento es un tipo de datos gráficos)	VARGRAPHIC(16336)
POSSTR (el argumento de serie de búsqueda, cuando el otro argumento es un BLOB)	BLOB(32672)
Primer argumento de SUBSTR	VARCHAR(32672)
Segundo y tercer argumento de SUBSTR	INTEGER
Segundo y tercer argumento de TRANSLATE	VARCHAR(32672) si el primer argumento es un tipo de carácter; VARGRAPHIC(16336) si el primer argumento es un tipo de gráfico
Cuarto argumento de TRANSLATE	VARCHAR(1) si el primer argumento es un tipo de carácter; VARGRAPHIC(1) si el primer argumento es un tipo de gráfico
Segundo argumento de TIMESTAMP	TIME
Primer argumento de VARCHAR_FORMAT	TIMESTAMP(12)
Primer argumento de TIMESTAMP_FORMAT	VARCHAR(254)
Primer argumento de XMLVALIDATE	XML
Primer argumento de XMLCOMMENT	VARCHAR(32672)

Tabla 26. Uso de expresiones sin tipo en funciones incorporadas (continuación)

Ubicación del marcador de parámetro sin tipo	Tipo de datos
Primer argumento de XMLTEXT	VARCHAR(32672)
Segundo argumento de XMLPI	VARCHAR(32672)
Primer argumento de XMLSERIALIZE	XML
Primer argumento de XMLDOCUMENT	XML
Primer argumento de XMLXSROBJECTID	XML
Todos los argumentos de XMLCONCAT	XML
Segundo argumento de TRIM_ARRAY	BIGINT
Índice de matriz de una ARRAY	BIGINT
Menos unario	DECFLOAT(34)
Más unario	DECFLOAT(34)
El resto de argumentos del resto de funciones escalares	El tipo de datos del parámetro de la definición de funciones según determine la resolución de funciones. La longitud del argumento se deriva según se muestra en la Tabla 21 en la página 211 en el apartado Resolución de funciones.
Argumentos de una función agregada	Error

Tabla 27. Uso de expresiones sin tipo en rutinas definidas por el usuario

Ubicación del marcador de parámetro sin tipo	Tipo de datos
El argumento de una función	El tipo de datos y la longitud del parámetro, tal como se ha definido al crearse la función.
El argumento de un método	Error
El argumento de un procedimiento	El tipo de datos del parámetro, tal como se ha definido al crearse el procedimiento

Expresión de fila

Una expresión de fila especifica una fila de datos que puede tener un tipo de fila específico definido por el usuario o el tipo de datos incorporado ROW.

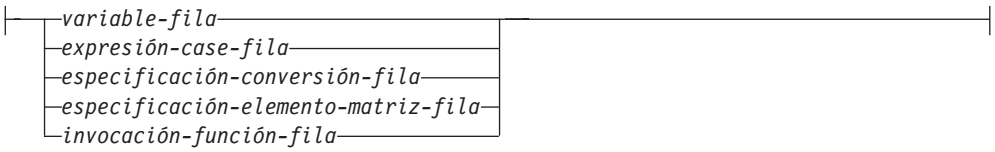
Autorización

El uso de algunas de las expresiones de fila puede requerir la autorización adecuada. Para estas expresiones de fila, el ID de autorización de la sentencia debe tener los privilegios siguientes:

- *variable-fila*. Para mas información sobre las consideraciones de autorización si *variable-fila* es una variable global, vea "Variables globales".
- *invocación-función-fila*. Autorización para ejecutar la función. Para obtener información acerca de las consideraciones sobre la autorización, consulte la sección sobre invocación de funciones.
- *expresión*. Puede que se requieran autorizaciones para usar ciertas expresiones a las que se hace referencia en una *expresión-fila*. Para más detalles sobre las consideraciones de autorización, vea la sección sobre expresiones.

Sintaxis

expresión-fila:



Descripción

variable-fila
Variable que está definida con un tipo de fila.

expresión-case-fila
Expresión-case que devuelve un tipo de fila.

especificación-conversión-fila
CAST que devuelve un tipo de fila.

especificación-elemento-matriz-fila
Especificación-elemento-matriz de una matriz con elementos de tipo de fila.

invocación-función-fila
Invocación-función de una función definida por el usuario que devuelve un tipo que es una fila. La función podría devolver un tipo de fila definido por el usuario o el tipo de datos ROW con tipos de campo y nombres de campo definidos.

Notas

- Las expresiones de fila se pueden utilizar para generar una fila en contextos de SQL PL.

Ejemplo

Predicados

Un *predicado* especifica una condición que es verdadera, falsa o desconocida sobre un valor, fila o grupo determinados.

Las siguientes normas se aplican a todos los tipos de predicados:

- Todos los valores especificados en un predicado debe ser compatibles.
- Una expresión utilizada en un predicado básico, cuantificado, IN o BETWEEN no debe producir una serie de caracteres con un atributo de longitud superior a 4000, una serie gráfica con un atributo de longitud superior a 2000 ni una serie LOB de cualquier tamaño.
- El valor de una variable del lenguaje principal puede ser nulo (es decir, la variable puede tener una variable indicadora negativa).
- La conversión de la página de códigos de los operandos de los predicados que implican dos o más operandos, a excepción de LIKE, se realiza según las normas de conversión de series.
- La utilización de un valor de tipo estructurado está limitado al predicado NULL y al predicado TYPE.
- En una base de datos Unicode, todos los predicados que acepten una serie de caracteres o gráfica aceptarán todo tipo de serie para el que se soporte la conversión.

Una selección completa es una forma de sentencia SELECT que, cuando se utiliza en un predicado, también se denomina una *subconsulta*.

Proceso de predicados para consultas

Un predicado es un elemento de una condición de búsqueda que expresa o implica una operación de comparación. Los predicados pueden agruparse en cuatro categorías que se determinan por la forma y el momento en que se utiliza el predicado en el proceso de evaluación. A continuación se listan las categorías, ordenadas de acuerdo con el rendimiento, empezando por el más favorable:

- Los predicados de delimitación de rango son los que se utilizan para delimitar una exploración de índice; proporcionan valores clave de inicio o de detención para la búsqueda de índice. El gestor de índices evalúa estos predicados.
- Los predicados comparables mediante SARG de índice no se utilizan para delimitar una búsqueda, pero se evalúan desde el índice si se selecciona uno, porque las columnas implicadas en el predicado forman parte de la clave de índice. El gestor de índices evalúa estos predicados.
- Los predicados comparables mediante SARG de datos son los predicados que el gestor de índices no puede evaluar pero que Data Management Services sí puede evaluar. Normalmente, estos predicados requieren el acceso de filas individuales de una tabla base. Si es necesario, DMS recuperará las columnas necesarias para evaluar el predicado, y cualquier otra columna, para satisfacer las columnas de la lista SELECT que no se hayan podido obtener del índice.
- Los predicados residuales son los que requieren E/S más allá del simple acceso de una tabla base. Los ejemplos de predicados residuales incluyen los que utilizan subconsultas cuantificadas (subconsultas con ANY, ALL, SOME o IN), o datos de lectura de objeto grande (LOB) que se almacenan por separado de la tabla. Estos predicados los evalúa Relational Data Services (RDS) y son los más costosos de las cuatro categorías de predicados.

La tabla siguiente proporciona ejemplos de distintos predicados e identifica su tipo de acuerdo con el contexto en que se utilizan.

Nota: En estos ejemplos, se presupone que existe un índice ascendiente de varias columnas (c1, c2, c3) y que se utiliza en la evaluación de los predicados si procede. Si cualquier columna del índice está en orden descendente, las claves de inicio y detención se pueden intercambiar para los predicados de delimitación de rango.

Tabla 28. Proceso de predicados para distintas consultas

Predicados	Columna c1	Columna c2	Columna c3	Comentarios
c1 = 1 and c2 = 2 and c3 = 3	Delimitación de rango (inicio-detención)	Delimitación de rango (inicio-detención)	Delimitación de rango (inicio-detención)	Los predicados de igualdad de todas las columnas del índice se pueden aplicar como claves de inicio-detención.
c1 = 1 and c2 = 2 and c3 = 3	Delimitación de rango (inicio-detención)	Delimitación de rango (inicio-detención)	Delimitación de rango (inicio)	Las columnas c1 y c2 están vinculadas mediante predicados de igualdad y el predicado de c3 sólo se aplica como clave de inicio.

Tabla 28. Proceso de predicados para distintas consultas (continuación)

Predicados	Columna c1	Columna c2	Columna c3	Comentarios
c1 =1 and c2 = 2	Delimitación de rango (inicio)	Delimitación de rango (inicio-detención)	No aplicable	La columna inicial, c1, tiene un predicado = y se puede utilizar como clave de inicio. La siguiente columna, c2, tiene un predicado de igualdad y, por lo tanto, también se puede aplicar como clave de inicio-detención.
c1 = 1 and c3 = 3	Delimitación de rango (inicio-detención)	No aplicable	Comparable mediante SARG de índice	El predicado de c3 no se puede utilizar como clave de inicio-detención, puesto que no hay ningún predicado de c2. No obstante, se puede aplicar como predicado comparable mediante SARG de índice.
c1 = 1 and c2 > 2 and c3 = 3	Delimitación de rango (inicio-detención)	Delimitación de rango (inicio)	Comparable mediante SARG de índice	El predicado de c3 no se puede aplicar como predicado de inicio-detención porque la columna anterior tiene un predicado >. Si, en su lugar, hubiese habido un predicado =, podríamos utilizarlo como clave de inicio-detención.
c1 = 1 and c2 = 2 and c4 = 4	Delimitación de rango (inicio-detención)	Delimitación de rango (detención)	Comparable mediante SARG de datos	Aquí, el predicado de c2 es un predicado =. Se puede utilizar como clave de detención. El predicado de c4 no se puede aplicar sobre el índice y se aplica como predicado comparable mediante SARG da datos durante la operación FETCH.
c2 = 2 and UDF_with_ external_ action(c4)	No aplicable	Comparable mediante SARG de índice	Residual	La columna inicial, c1, no tiene predicado y, por lo tanto, se puede aplicar el predicado de la columna c2 como predicado comparable mediante SARG de índice en el que se explora todo el índice. El predicado que implica la función definida por el usuario con acción externa (UDF_with_external_action) se aplica como predicado residual.

Tabla 28. Proceso de predicados para distintas consultas (continuación)

Predicados	Columna c1	Columna c2	Columna c3	Comentarios
c1 = 1 or c2 = 2	Comparable mediante SARG de índice	Comparable mediante SARG de índice	No aplicable	La presencia de un operador OR no nos permite utilizar este índice de varias columnas como claves de inicio-detención. Esto hubiera podido suceder si hubieran existido dos índices, uno con una columna inicial en c1 y otro con una columna inicial en c2, y el optimizador de DB2 hubiera elegido un plan de "operación OR en el índice". No obstante, en este caso los dos predicados se tratan como predicados comparables mediante SARG de índice.
c1 < 5 and (c2 = 2 or c3 = 3)	Delimitación de rango (detención)	Comparable mediante SARG de índice	Comparable mediante SARG de índice	Aquí, la columna inicial, c1, se aprovecha para impedir que la exploración de índice utilice el predicado con una clave de detención. El predicado OR de c2 y c3 se aplica como predicados comparables mediante SARG de índice.

El optimizador de DB2 emplea el mecanismo de reescritura de consultas para transformar muchos predicados complejos escritos por el usuario en consultas con un mejor rendimiento, como se muestra en la tabla siguiente:

Tabla 29. Predicados de reescritura de consultas

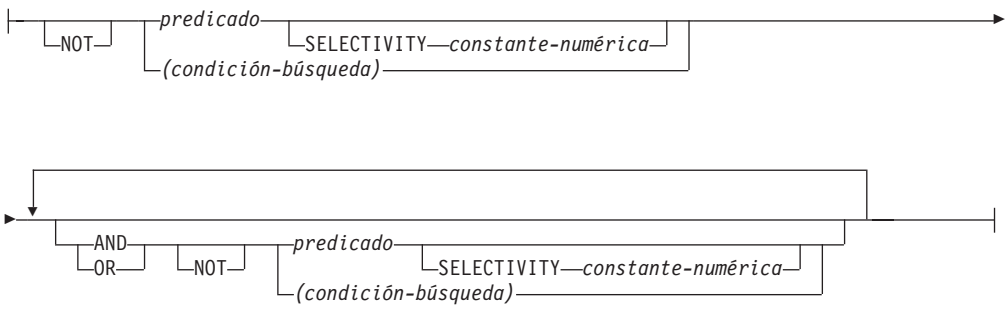
Predicado o consulta original	Predicados optimizados	Comentarios
c1 between 5 and 10	c1 = 5 and c1 = 10	Los predicados BETWEEN se reescriben transformándolos en los predicados de delimitación de rango equivalentes para que se puedan utilizar internamente como si el usuario hubiese especificado los predicados de delimitación de rango.
c1 not between 5 and 10	c1 < 5 or c1 > 10	La presencia del predicado OR no permite la utilización de una clave de inicio-detención a menos que el optimizador de DB2 elija un plan de operación OR en el índice.
SELECT * **FROM** t1 **WHERE EXISTS** (**SELECT** c1 **FROM** t2 **WHERE** t1.c1 = t2.c1)	**SELECT** t1.* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	La subconsulta puede transformarse en una unión.

Tabla 29. Predicados de reescritura de consultas (continuación)

Predicado o consulta original	Predicados optimizados	Comentarios
SELECT * **FROM** t1 **WHERE** t1.c1 **IN** (**SELECT** c1 **FROM** t2)	**SELECT** t1* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	Este caso es similar a la transformación del ejemplo de predicado EXISTS anterior.
c1 like 'abc%'	c1 = 'abc X X X ' and c1 = 'abc Y Y Y'	Si tenemos c1 como columna inicial de un índice, DB2 genera estos predicados para que se puedan aplicar como predicados de inicio y detención de delimitación de rango. Aquí, los caracteres X e Y son símbolos que representan el carácter de clasificación más bajo y más alto.
c1 like 'abc%def'	c1 = 'abc X X X ' and c1 = 'abc Y Y Y' and c1 like 'abc%def'	Este caso es como el anterior, a excepción de que debemos aplicar además el predicado original como predicado comparable mediante SARG de índice. Esto asegura que los caracteres se correspondan con corrección.

Condiciones de búsqueda

condición-búsqueda:



Una *condición de búsqueda* especifica una condición que es"verdadera", "falsa" o "desconocida" acerca de un valor, fila o grupo determinado.

El resultado de una condición de búsqueda se deriva por la aplicación de *operadores lógicos* (AND, OR, NOT) especificados al resultado de cada predicado especificado. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

AND y OR se definen en la Tabla 30, en la que P y Q son unos predicados cualesquiera:

Tabla 30. Tablas de evaluación para AND y OR

P	Q	P AND Q	P OR Q
Verdadero	Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso	Verdadero
Verdadero	Desconocido	Desconocido	Verdadero
Falso	Verdadero	Falso	Verdadero
Falso	Falso	Falso	Falso
Falso	Desconocido	Falso	Desconocido
Desconocido	Verdadero	Desconocido	Verdadero
Desconocido	Falso	Falso	Desconocido
Desconocido	Desconocido	Desconocido	Desconocido

NOT(verdadero) es falso, NOT(falso) es verdadero y NOT(desconocido) es desconocido.

En primer lugar se evalúan las condiciones de búsqueda entre paréntesis. Si el orden de evaluación no se especifica mediante paréntesis, NOT se aplica antes que AND y AND es aplica antes que OR. El orden en el que se evalúan los operadores del mismo nivel de prioridad no está definido, para permitir la optimización de condiciones de búsqueda.

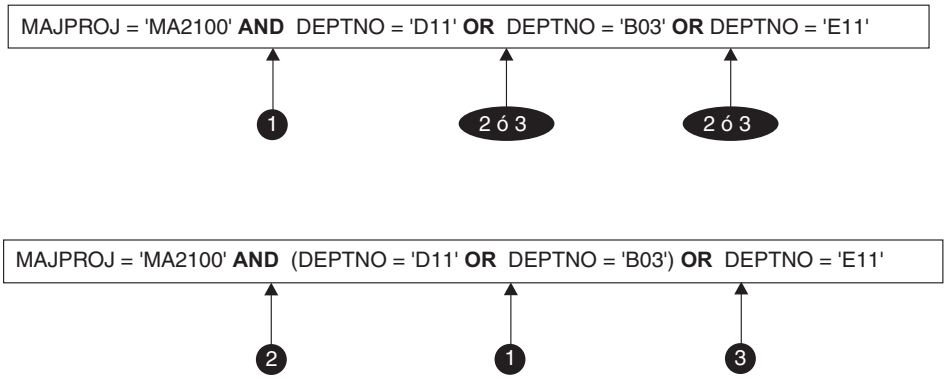


Figura 12. Orden de evaluación de las condiciones de búsqueda

SELECTIVITY *valor*

La cláusula SELECTIVITY se utiliza para indicar a DB2 qué porcentaje de selectividad prevista corresponde al predicado. SELECTIVITY se puede especificar sólo cuando el predicado es un predicado definido por el usuario.

Un predicado definido por el usuario consta de una invocación de función definida por el usuario, en el contexto de una especificación de predicado que coincide con la existente en la cláusula PREDICATES de CREATE FUNCTION. Por ejemplo, si la función `foo` está definida con PREDICATES WHEN=1..., es válido utilizar SELECTIVITY de este modo:

```
SELECT *
  FROM STORES
  WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

El valor de selectividad debe ser un valor literal numérico comprendido dentro del rango inclusivo 0-1 (SQLSTATE 42615). Si SELECTIVITY no se especifica, el valor por omisión es 0.01 (es decir, el predicado definido por el usuario debe descartar todas las filas de la tabla excepto un 1 por ciento. El valor por omisión de SELECTIVITY se puede modificar para una función determinada actualizando su columna SELECTIVITY en la vista SYSSTAT.ROUTINES. Se obtiene un error si la cláusula SELECTIVITY se especifica para un predicado no definido por el usuario (SQLSTATE 428E5).

Se puede utilizar una función definida por el usuario (UDF) como predicado definido por el usuario y, por tanto, puede permitir la utilización de índices si:

- La especificación de predicado está presente en la sentencia CREATE FUNCTION
- la UDF se invoca en una cláusula WHERE que se compara (sintácticamente) de la misma manera que se especifica en la especificación de predicado
- no existe ninguna negación (operador NOT)

Ejemplos

En la consulta siguiente, la especificación UDF `interna` de la cláusula WHERE cumple las tres condiciones y se considera que es un predicado definido por el usuario.

```
SELECT *
  FROM customers
  WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

Sin embargo, la presencia de `within` en la consulta siguiente no permite el uso de índices debido a la negación, y no se considera un predicado definido por el usuario.

```
SELECT *
  FROM customers
  WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

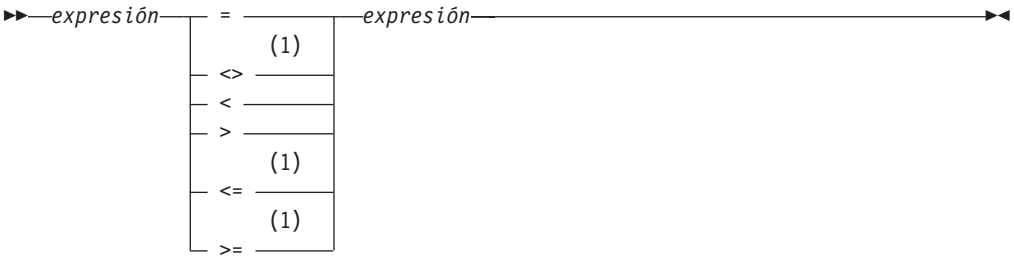
En el ejemplo siguiente, se identifican los clientes y tiendas que están a una determinada distancia entre sí. La distancia de una tienda a otra se calcula mediante el radio de la ciudad donde viven los clientes.

```
SELECT *
  FROM customers, stores
  WHERE distance(customers.loc, stores.loc) <
    CityRadius(stores.loc) SELECTIVITY 0.02
```

En la consulta anterior, se considera que el predicado contenido en la cláusula WHERE es un predicado definido por el usuario. El resultado producido por CityRadius se utiliza como argumento de búsqueda para la función productora de rangos.

Sin embargo, como el resultado devuelto por CityRadius se utiliza como función productora de rangos, el predicado definido por el usuario no podrá utilizar la extensión de índice definida para la columna stores.loc. Por lo tanto, la UDF sólo utilizará el índice definido en la columna customers.loc.

Predicado básico



Notas:

1 También se soportan los formatos siguientes de los operadores de comparación en predicados básicos y cuantificados: ^=, ^<, ^>, !=, !< y !>. En las páginas de códigos 437, 819 y 850, se da soporte a los formatos ¬=, ¬< y ¬>. Todos estos formatos específicos de producto de los operadores de comparación sólo están destinados a soportar las sentencias de SQL existentes que utilizan estos operadores y no se recomienda utilizarlos al escribir sentencias de SQL nuevas.

Un *predicado básico* compara dos valores.

Si el valor de cualquier operando es nulo, el resultado del predicado será desconocido. De lo contrario el resultado es verdadero o falso.

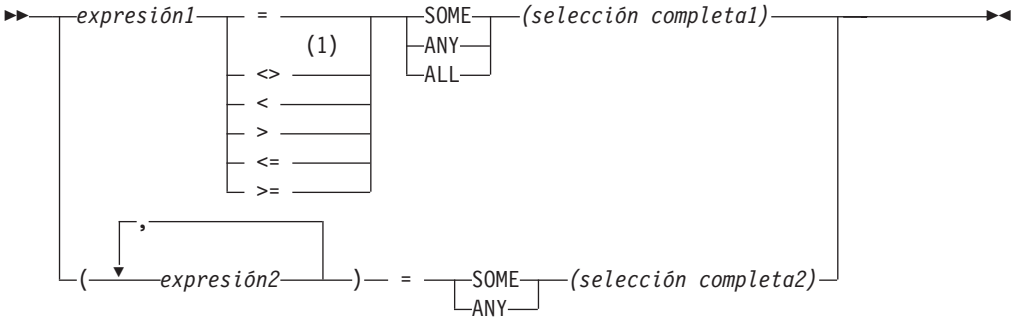
Para valores *x* e *y*:

Predicado	**Es verdadero si y sólo si...**
x **=** *y*	*x* es igual a *y*
x **<>** *y*	*x* es diferente de *y*
x **<** *y*	*x* es menor que *y*
x **>** *y*	*x* es mayor que *y*
x **>=** *y*	*x* es mayor o igual que *y*
x **<=** *y*	*x* es menor o igual que *y*

Ejemplos:

```
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

Predicado cuantificado



Notas:

1 También se soportan los formatos siguientes de los operadores de comparación en predicados básicos y cuantificados: ^=, ^<, ^>, !=, !< y !>. En las páginas de códigos 437, 819 y 850, se da soporte a los formatos ¬=, ¬< y ¬>. Todos estos formatos específicos de producto de los operadores de comparación sólo están destinados a soportar las sentencias de SQL existentes que utilizan estos operadores y no se recomienda utilizarlos al escribir sentencias de SQL nuevas.

Un *predicado cuantificado* compara un valor o valores con una colección de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda del operador del predicado (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

Cuando se especifica ALL:

- El resultado del predicado es verdadero si la selección completa no devuelve ningún valor o si la relación especificada es verdadera para cada valor que devuelva la selección completa.
- El resultado es falso si la relación especificada es falsa para un valor como mínimo que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es falsa para ninguno de los valores que devuelve la selección completa y una comparación como mínimo es desconocida debido a un valor nulo.

Cuando se especifica SOME o ANY:

- El resultado del predicado es verdadero si la relación especificada es verdadera para cada valor de una fila como mínimo que devuelve la selección completa.
- El resultado es falso si la selección completa no devuelve ninguna fila o si la relación especificada es falsa para como mínimo un valor de cada fila que devuelve la selección completa.
- El resultado es desconocido si la relación especificada no es verdadera para cualquiera de las filas y, como mínimo, una comparación es desconocida debido a un valor nulo.

Ejemplos: Utilice las tablas siguientes al hacer referencia a los ejemplos siguientes.

TBLAB:

COLA	COLB
1	12
2	12
3	13
4	14
-	-

TBLXY:

COLX	COLY
2	22
3	23

Figura 13. Ejemplos de tablas de predicado cuantificado

Ejemplo 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 2,3. La subselección devuelve (2,3). COLA en las filas 2 y 3 es igual al menos a uno de estos valores.

Ejemplo 2

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

Da como resultado 3,4. La subselección devuelve (2,3). COLA en las filas 3 y 4 es mayor que al menos uno de estos valores.

Ejemplo 3

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

Da como resultado 4. La subselección devuelve (2,3). COLA en la fila 4 es el único que es mayor que estos dos valores.

Ejemplo 4

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

Da como resultado 1,2,3,4, nulo. La subselección no devuelve ningún valor. Por lo tanto, el predicado es verdadero para todas las filas de TBLAB.

Ejemplo 5

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

La subselección devuelve todas las entradas de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

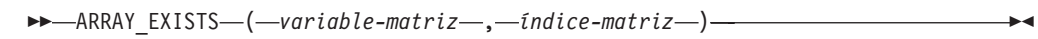
Ejemplo 6

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB) = ANY (SELECT COLX,COLY-10 FROM TBLXY)
```

La subselección devuelve COLX y COLY-10 de TBLXY. El predicado es verdadero para la subselección, por lo tanto el resultado es el siguiente:

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

ARRAY_EXISTS



El predicado ARRAY_EXISTS comprueba la existencia de un índice de matriz en una matriz.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

índice-matriz
El tipo de datos de *índice-matriz* se debe poder asignar al tipo de datos del índice de la matriz. Si *variable-matriz* es una matriz común, *índice matriz* se debe poder asignar a INTEGER (SQLSTATE 428H1).

El resultado es verdadero si *variable-matriz* incluye un índice de matriz que sea igual que *índice-matriz* convertido en el tipo de datos del índice de matriz de *variable-matriz*; de lo contrario, el resultado es falso.

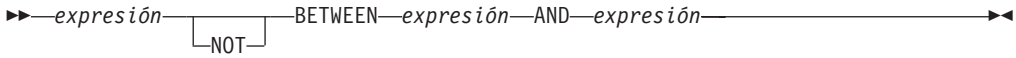
El resultado no puede ser desconocido; si cualquier argumento es nulo, el resultado es falso.

Ejemplo

- Supongamos que la variable de matriz RECENT_CALLS está definida como una matriz común de tipo de matriz PHONENUMBERS. La siguiente sentencia IF comprueba si la lista de llamadas recientes ha alcanzado ya la 40ª llamada guardada. En caso afirmativo, la variable booleana local EIGHTY_PERCENT se establece en true:

```
IF (ARRAY_EXISTS(RECENT_CALLS, 40)) THEN
  SET EIGHTY_PERCENT = TRUE;
END IF
```

Predicado BETWEEN



El predicado BETWEEN compara un valor con un rango de valores. Si los tipos de datos de los operandos no son iguales, todos los valores se convierten al tipo de datos que se generaría al aplicar las "Normas para tipos de datos de resultados", excepto si los tipos de datos de todos los operandos son numéricos, en cuyo caso no se convierte ningún valor.

El predicado BETWEEN:

```
valor1 BETWEEN valor2 AND valor3
```

es equivalente a la condición de búsqueda:

```
valor1 >= valor2 AND valor1 <= valor3
```

El predicado BETWEEN:

```
value1 NOT BETWEEN value2 AND value3
```

es equivalente a la condición de búsqueda:

```
NOT(valor1 BETWEEN valor2 AND valor3); es decir,
valor1 < valor2 OR valor1 > valor3.
```

El primer operando (expresión) no puede incluir una función que sea no determinista o que tenga una acción externa (SQLSTATE 42845).

Ejemplos

Ejemplo 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

Devuelve todos los salarios comprendidos entre 20.000 y 40.000 dólares.

Ejemplo 2

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

Suponiendo que :HV1 es 5000, da como resultado todos los salarios que son inferiores a 25.000 dólares y superiores a 40.000.

Predicados de cursor

Los predicados de cursor son palabras clave SQL que se pueden utilizar para determinar el estado de un cursor definido en el ámbito actual. Proporcionan una forma de indicar fácilmente si un cursor está abierto, cerrado o si hay filas asociadas con el cursor.

nombre-variable-cursor
Nombre de una variable de SQL o un parámetro de SQL de un tipo de cursor.

IS
Especifica que debe comprobarse una propiedad de predicado de cursor.

NOT
Especifica que se devolverá el valor opuesto al de la prueba de la propiedad de predicado del cursor.

FOUND
Especifica que se comprobará si el cursor contiene filas después de la ejecución de una sentencia FETCH. Si la última sentencia FETCH ejecutada ha sido correcta, y si se utiliza la sintaxis del predicado IS FOUND, el valor devuelto es TRUE. Si la última sentencia FETCH ejecutada ha dado como resultado una condición en la que no se han encontrado filas, el resultado es FALSE. El resultado se desconoce si:

- el valor de nombre-variable-cursor es nulo
- el cursor subyacente de nombre-variable-cursor no está abierto
- el predicado se evalúa antes de que se ejecute la primera acción FETCH sobre el cursor subyacente
- la última acción FETCH ha devuelto un error

El predicado IS FOUND puede resultar de utilidad cuando una parte de la lógica SQL PL hace un bucle y realiza una captación con cada iteración. El predicado se puede utilizar para determinar si faltan filas por captar. Proporciona una alternativa eficaz para utilizar un manejador de condiciones que comprueba la condición de error que aparece cuando no quedan más filas por captar.

Si se especifica la palabra clave NOT, de modo que la sintaxis es IS NOT FOUND, el valor del resultado es el opuesto.

OPEN
Especifica que se comprobará si el cursor está en un estado abierto. Si el cursor está abierto y si se utiliza la sintaxis del predicado IS OPEN, el valor devuelto es TRUE. Este puede resultar un predicado útil en aquellos casos en los que se pasan cursores como parámetros a las funciones y los procedimientos. Antes de intentar abrir el cursor, este predicado se puede utilizar para determinar si el cursor ya está abierto.

Si se especifica la palabra clave NOT, de modo que la sintaxis es IS NOT OPEN, el valor del resultado es el opuesto.

Notas

- Un predicado de cursor sólo puede utilizarse en sentencias en una sentencia de SQL compuesto (compilado) (SQLSTATE 42818).

Ejemplo

El script siguiente define un procedimiento SQL que contiene referencias a estos predicados así como a los objetos de requisito previo necesarios para compilar correctamente y llamar al procedimiento:

```
CREATE TABLE T1 (c1 INT, c2 INT, c3 INT)@

INSERT INTO T1 VALUES (1,1,1),(2,2,2),(3,3,3) @

CREATE TYPE myRowType AS ROW(c1 INT, c2 INT, c3 INT)@

CREATE TYPE myCursorType AS myRowType CURSOR@


CREATE PROCEDURE p(OUT count INT)
LANGUAGE SQL
BEGIN
  DECLARE C1 CURSOR;
  DECLARE lvarInt INT;

  SET count = -1;
  SET c1 = CURSOR FOR SELECT c1 FROM t1;

  IF (c1 IS NOT OPEN) THEN
     OPEN c1;
  ELSE
     set count = -2;
  END IF;

  SET count = 0;
  IF (c1 IS OPEN) THEN

    FETCH c1 INTO lvarInt;

    WHILE (c1 IS FOUND) DO
      SET count = count + 1;
      FETCH c1 INTO lvarInt;
    END WHILE;
  ELSE
     SET COUNT = 0;
  END IF;

END@

CALL p()@
```

Predicado EXISTS

El predicado EXISTS comprueba la existencia de ciertas filas.

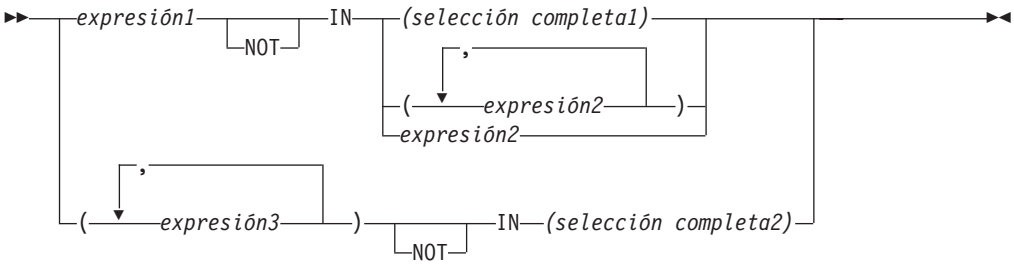
La selección completa puede especificar cualquier número de columnas y

- El resultado es verdadero sólo si el número de filas especificadas mediante la selección completa no es cero.
- El resultado es falso sólo si el número de filas especificadas es cero
- El resultado no puede ser desconocido.

Ejemplo

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

Predicado IN



El predicado IN compara un valor o valores con un conjunto de valores.

La selección completa debe identificar un número de columnas que sea el mismo que el número de expresiones especificadas a la izquierda de la palabra clave IN (SQLSTATE 428C4). La selección completa puede devolver cualquier número de filas.

- Un predicado IN del formato:

  ```
  expresión IN expresión
  ```

  es equivalente a un predicado básico del formato:

  ```
  expresión = expresión
  ```

- Un predicado IN del formato:

  ```
  expresión IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  expresión = ANY (selección completa)
  ```

- Un predicado IN del formato:

  ```
  expresión NOT IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  expresión <> ALL (selección completa)
  ```

- Un predicado IN del formato:

  ```
  expresión IN (expresióna, expresiónb, ..., expresiónk)
  ```

  es equivalente a:

  ```
  expresión = ANY (selección completa)
  ```

  donde selección completa en el formato de la cláusula-values es:

  ```
  VALUES (expresióna), (expresiónb), ..., (expresiónk)
  ```

- Un predicado IN del formato:

  ```
  (expresióna, expresiónb,..., expresiónk) IN (selección completa)
  ```

  es equivalente a un predicado cuantificado del formato:

  ```
  (expresióna, expresiónb,..., expresiónk) = ANY (selección completa)
  ```

  Tenga en cuenta que se hace referencia al operando situado en la parte izquierda del formato de estos predicados como una *expresión-valor-fila*.

Los valores para *expresión1* y *expresión2* o la columna de *selección completa1* del predicado IN deben ser compatibles. Cada valor de *expresión3* y su columna

correspondiente de *selección completa2* del predicado IN deben ser compatibles. Pueden utilizarse las normas para tipos de datos del resultado para determinar los atributos del resultado utilizados en la comparación.

Los valores para las expresiones del predicado IN (incluyendo las columnas correspondientes de una selección completa)pueden tener páginas de códigos diferentes. Si se precisa realizar una conversión, la página de códigos se determina aplicando las normas para las conversiones de series a la lista IN primero y, posteriormente, al predicado, utilizando la página de códigos derivada para la lista IN como segundo operando.

Ejemplos

Ejemplo 1: lo siguiente se evalúa como verdadero si el valor de la fila que se encuentra por debajo de la evaluación en al columna DEPTNO contiene D01, B01 o C01:

```
DEPTNO IN ('D01', 'B01', 'C01')
```

Ejemplo 2: lo siguiente sólo se evalúa como verdadero si el valor de EMPNO (número de empleado) del lado izquierdo coincide con el valor de EMPNO de un empleado del departamento E11:

```
EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

Ejemplo 3: dada la siguiente información, este ejemplo se evalúa en verdadero si el valor específico de la fila de la columna COL_1 coincide con cualquiera de los valores de la lista:

Tabla 31. Ejemplo de predicado IN

Expresiones	Tipo	Página de códigos
COL_1	columna	850
HV_2	variable del lenguaje principal	437
HV_3	variable del lenguaje principal	437
CON_1	constante	850

Cuando se evalúa el predicado:

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

las dos variables del lenguaje principal se convertirán a la página de códigos 850, en base a las normas para las conversiones de series.

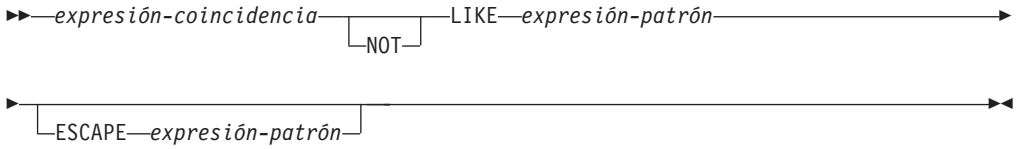
Ejemplo 4: lo siguiente se evalúa en verdadero si el año especificado en EMENDATE (fecha de una actividad del empleado en un proyecto finalizado) coincide con cualquiera de los valores especificados en la lista (el año actual o los dos años anteriores):

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),
                   YEAR(CURRENT DATE - 1 YEAR),
                   YEAR(CURRENT DATE - 2 YEARS))
```

Ejemplo 5: lo siguiente se evalúa en verdadero si tanto el valor de ID como el de DEPT del lado izquierdo coinciden con MANAGER y DEPTNUMB respectivamente para cualquiera de las filas de la tabla ORG.

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

Predicado LIKE



El predicado LIKE busca series que tienen un determinado patrón. El patrón se especifica mediante una serie en la que el signo de subrayado y de porcentaje pueden tener un significado especial. Los blancos finales de un patrón forman parte del mismo.

Si el valor de cualquiera de los argumentos es nulo, el resultado del predicado LIKE es desconocido.

Los valores de *expresión-coincidencia*, *expresión-patrón* y *expresión-escape* son expresiones de serie compatibles. Hay ligeras diferencias en los tipos de expresiones de series soportados por cada uno de los argumentos. Los tipos válidos de las expresiones se listan bajo la descripción de cada argumento.

Ninguna de las expresiones puede tener un tipo diferenciado. Sin embargo, pueden ser una función que convierta un tipo diferenciado en su tipo fuente.

expresión-coincidencia
Una expresión que especifica la serie que se debe examinar para ver si cumple con un determinado patrón de caracteres.

La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal (incluida una variable localizadora o una variable de referencia de archivo)
- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene cualquiera de las anteriores

expresión-patrón
Una expresión que especifica la serie que se debe comparar.

La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene cualquiera de las anteriores
- Un parámetro de procedimiento de SQL

con las siguientes restricciones:

- Ningún elemento en la expresión puede ser del tipo CLOB o DBCLOB. Además, no puede tratarse de una variable de referencia de archivo BLOB.
- La longitud real de *expresión-patrón* no puede superar los 32.672 bytes.

A continuación se muestran ejemplos de expresiones de serie o series válidas:

- Parámetros de función SQL definidos por el usuario
- Variables de transición activadores
- Variables locales en sentencias compuestas dinámicas

Una **descripción sencilla** del uso del predicado LIKE es que el patrón se utilice para especificar los criterios de cumplimiento correspondientes a los valores de *expresión-coincidencia*, donde:

- El carácter de subrayado (_) representa cualquier carácter único.
- El signo de porcentaje (%) representa una serie de cero o más caracteres.
- Cualquier otro carácter se representa a sí mismo.

Si la *expresión-patrón* tiene que incluir el carácter de subrayado o de porcentaje, la *expresión-escape* se utiliza para especificar un carácter que precede al carácter de subrayado o de porcentaje en el patrón.

A continuación se ofrece una **descripción rigurosa** del predicado LIKE. Tenga en cuenta que en esta descripción se pasa por alto el uso de la *expresión-escape*; su uso se explicará más adelante.

- Supongamos que *m* indica el valor de *expresión-coincidencia* y que *p* indica el valor de *expresión-patrón*. La serie *p* se interpreta como una secuencia el número mínimo de especificadores de subserie, de modo que cada carácter de *p* forma parte de exactamente un especificador de subserie. Un especificador de subserie es un carácter de subrayado, un signo de porcentaje o una secuencia no vacía de caracteres que no son el signo de subrayado ni de porcentaje.

  El resultado del predicado es desconocido si *m* o *p* es el valor nulo. De lo contrario, el resultado es verdadero (true) o falso (false). El resultado es true si *m* y *p* son ambos series vacías o existe un particionamiento de *m* en subseries como:

  – Una subserie de *m* es una secuencia de cero o más caracteres contiguos y cada carácter de *m* forma parte de exactamente una subserie.
  – Si el especificador de subserie número *n* es un carácter de subrayado, la subserie número *n* de *m* es cualquier carácter único.
  – Si el especificador de subserie número *n* es un carácter de porcentaje, la subserie número *n* de *m* es cualquier secuencia de cero o más caracteres.
  – Si el especificador de subserie número *n* no es ni un signo de subrayado ni uno de porcentaje, la subserie número *n* de *m* es igual a dicho especificador de subserie y tiene la misma longitud que dicho especificador de subserie.
  – El número de subseries de *m* es igual al número de especificadores de subserie.

  Por lo tanto, si *p* es una serie vacía y *m* no es una serie vacía, el resultado es false. Paralelamente, si *m* es una serie vacía y *p* no es una serie vacía (excepto para una serie que solo contenga signos de porcentaje), el resultado es false.

  El predicado *m* NOT LIKE *p* es equivalente a la condición de búsqueda NOT (*m* LIKE *p*).

Cuando se especifica la *expresión-escape*, la *expresión-patrón* no debe contener el carácter de escape especificado por la *expresión-escape*, excepto cuando va seguido inmediatamente del carácter de escape, el carácter de subrayado o el carácter de porcentaje (SQLSTATE 22025).

Si la *expresión-coincidencia* es una serie de caracteres en una base de datos MBCS, puede contener datos mixtos. En este caso, el patrón puede incluir tanto caracteres SBCS como no SBCS. Para bases de datos que no son Unicode, los caracteres especiales del patrón se interpretan del siguiente modo:

- Un signo de subrayado de media anchura SBCS hace referencia a un carácter SBCS.
- Un carácter de subrayado de anchura completa no SBCS hace referencia a un carácter no SBCS.
- Un signo de porcentaje de media anchura SBCS o de anchura completa no SBCS hace referencia a cero o más caracteres SBCS o no SBCS.

En una base de datos Unicode, no se suele hacer distinción entre caracteres de ″un solo byte″ y ″no de un solo byte″. Aunque el formato UTF-8 es una codificación de ″bytes mixtos″ de caracteres Unicode, no existe ninguna distinción real entre caracteres SBCS y no SBCS en UTF-8. Cada carácter es un carácter Unicode, independientemente del número de bytes en formato UTF-8.

En una columna gráfica Unicode, cada carácter no suplementario, incluidos el carácter de subrayado de media anchura (U&'\005F') y el carácter de signo de porcentaje de media anchura (U&'\0025'), tiene una anchura de dos bytes. En una base de datos Unicode, los caracteres especiales de un patrón se interpretan del siguiente modo:

- Para series de caracteres, un carácter de subrayado de media anchura (X'5F') o un carácter de subrayado de anchura completa (X'EFBCBF') hace referencia a un carácter Unicode y un carácter de signo de porcentaje de media anchura (X'25') o un carácter de signo de porcentaje de anchura completa (X'EFBC85') hace referencia a cero o más caracteres Unicode.
- Para series gráficas, un carácter de subrayado de media anchura (U&'\005F') o un carácter de subrayado de anchura completa (U&'\FF3F') hace referencia a un carácter Unicode y un carácter de signo de porcentaje de media anchura (U&'\0025') o un carácter de signo de porcentaje de anchura completa (U&'\FF05') hace referencia a cero o más caracteres Unicode.
- Para ser reconocidos como caracteres especiales cuando esté vigente una recopilación basada en UCA sensible a la configuración local, el carácter de subrayado y el carácter de signo de porcentaje no deben ir seguidos por signos (diacríticos) de combinación sin avance de espacio. Por ejemplo, el patrón U&'%\0300' (carácter de signo de porcentaje seguido de un acento grave de combinación sin avanzar espacio) se interpretará como búsqueda del % y no como una búsqueda de cero o más caracteres Unicode seguido de una letra con un acento grave.

Un carácter suplementario Unicode se almacena como dos elementos de código gráficos en una columna gráfica Unicode. Para comparar un carácter suplementario Unicode en una columna gráfica Unicode, utilice un carácter de subrayado si la base de datos utiliza la clasificación basada en la UCA sensible a la configuración local y dos caracteres de subrayado en caso contrario. Para comparar un carácter suplementario Unicode en una columna de caracteres Unicode, utilice un carácter de subrayado para todas las clasificaciones. Para comparar un carácter base con uno o más caracteres de combinación sin avance de espacio de cola, utilice un carácter de subrayado si la base de datos utiliza la clasificación basada en la UCA sensible a la configuración local. En

caso contrario, utilice tantos caracteres de subrayado como el número de caracteres de combinación sin avance de espacio además del carácter base.

expresión-escape
Este argumento opcional es una expresión que especifica un carácter que se utilizará para modificar el significado especial de los caracteres de subrayado (_) y de porcentaje (%) en la *expresión-patrón*. Esto permite utilizar el predicado LIKE para comparar valores que contienen los caracteres reales de porcentaje y de subrayado.

La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene cualquiera de las anteriores

teniendo en cuenta las siguientes restricciones:

- Ningún elemento en la expresión puede ser del tipo CLOB o DBCLOB. Además, no puede ser una variable de referencia a archivos BLOB.
- Para columnas de caracteres, el resultado de la expresión debe ser un carácter o una serie binaria que contenga exactamente un byte (SQLSTATE 22019).
- Para columnas gráficas, el resultado de la expresión debe ser un carácter (SQLSTATE 22019).
- El resultado de la expresión no debe ser una secuencia de caracteres de combinación sin avance de espacio (por ejemplo U&'\0301', acento agudo de combinación).

Cuando hay caracteres de escape en la serie de patrón, un carácter de subrayado, de porcentaje o de escape puede representar una aparición literal de sí mismo. Esto es cierto si el carácter en cuestión va precedido de un número impar de caracteres de escape sucesivos. De lo contrario, no es cierto.

En un patrón, una secuencia de caracteres de escape sucesivos se trata del siguiente modo:

- Supongamos que S es una secuencia y que no forma parte de una secuencia más larga de caracteres de escape sucesivos. Supongamos también que S contiene un total de n caracteres. Las normas que controlan S dependen del valor de n:
  - Si n es impar, S debe ir seguido de un signo de subrayado o de porcentaje (SQLSTATE 22025). S y el carácter que lo sigue representan (n-1)/2 apariciones literales del carácter de escape seguidas de una aparición literal del signo de subrayado o de porcentaje.
  - Si n es par, S representa n/2 apariciones literales del carácter de escape. A diferencia del caso en que n es impar, S puede finalizar el patrón. Si no finaliza el patrón, puede ir seguido de cualquier carácter (excepto, por supuesto, de un carácter de escape, que violaría la suposición de que S no forma parte de una secuencia más larga de caracteres de escape sucesivos). Si S va seguido de un signo de subrayado o de porcentaje, dicho carácter tiene su significado especial.

A continuación se ilustra el efecto de apariciones sucesivas del carácter de escape que, en este caso, es la barra inclinada invertida (\).

Serie de patrón	Patrón real
\%	Un signo de porcentaje
\\%	Una barra inclinada invertida seguida de cero o más caracteres arbitrarios
\\\%	Una barra inclinada invertida seguida de un signo de porcentaje

La página de códigos utilizada en la comparación se basa en la página de códigos del valor de *expresión-coincidencia*.

- El valor de *expresión-coincidencia* nunca se convierte.
- Si la página de códigos de *expresión-patrón* difiere de la página de códigos de *expresión-coincidencia*, el valor de *expresión-patrón* se convierte a la página de códigos de *expresión-coincidencia*, a no ser que alguno de los operandos esté definido como FOR BIT DATA (en cuyo caso no hay conversión).
- Si la página de códigos de *expresión-escape* difiere de la página de códigos de *expresión-coincidencia*, el valor de *expresión-escape* se convierte a la página de códigos de *expresión-coincidencia*, a no ser que alguno de los operandos esté definido como FOR BIT DATA (en cuyo caso no hay conversión).

Notas

- El número de blancos finales es significativo tanto en *expresión-coincidencia* como en *expresión-patrón*. Si las series no tienen la misma longitud, la serie más corta no se rellena con espacios en blanco. Por ejemplo, la expresión `'PADDED ' LIKE 'PADDED'` no daría lugar a una coincidencia.
- Si el patrón especificado en un predicado LIKE es un marcador de parámetro y se utiliza una variable del lenguaje principal de caracteres de longitud fija para sustituir el marcador de parámetro, el valor especificado para la variable del lenguaje principal debe tener la longitud correcta. Si no se especifica la longitud correcta, la operación select no devolverá los resultados previstos.

  Por ejemplo, si la variable del lenguaje principal se define como CHAR(10) y se asigna el valor WYSE% a dicha variable del lenguaje principal, la variable del lenguaje principal se rellena con blancos durante la asignación. El patrón utilizado es:

  ```
  'WYSE%     '
  ```

  El gestor de bases de datos busca todos los valores que comienzan por WYSE y terminan por cinco espacios en blanco. Si desea buscar sólo valores que comienzan por 'WYSE', asigne el valor 'WSYE%%%%%' a la variable del lenguaje principal.
- El patrón se compara cotejando la base de datos, a menos que cualquiera de los operandos se defina como FOR BIT DATA, en cuyo caso el patrón se compara utilizando una comparación binaria.

Ejemplos

- Busque la serie 'SYSTEMS' que aparezca en cualquier lugar dentro de la columna PROJNAME de la tabla PROJECT.

  ```
  SELECT PROJNAME FROM PROJECT
    WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
  ```

- Busque una serie cuyo primer carácter sea 'J' que tenga exactamente dos caracteres de longitud en la columna FIRSTNME de la tabla EMPLOYEE.

  ```
  SELECT FIRSTNME FROM EMPLOYEE
    WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
  ```

- Busque una serie de cualquier longitud, cuyo primer carácter sea 'J', en la columna FIRSTNME de la tabla EMPLOYEE.

```
SELECT FIRSTNME FROM EMPLOYEE
  WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
```

- En la tabla CORP_SERVERS, busque una serie en la columna LA_SERVERS que coincida con el valor del registro especial CURRENT SERVER.

```
SELECT LA_SERVERS FROM CORP_SERVERS
  WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
```

- Recupere todas las series que comienzan por la secuencia de caracteres '\' en la columna A de la tabla T.

```
SELECT A FROM T
  WHERE T.A LIKE '\_\\%' ESCAPE '\'
```

- Utilice la función escalar BLOB para obtener un carácter de escape de un byte que sea compatible con los tipos de datos de patrón y de coincidencia (ambos BLOB).

```
SELECT COLBLOB FROM TABLET
  WHERE COLBLOB LIKE :pattern_var ESCAPE BLOB(X'0E')
```

- En una base de datos Unicode definida con la clasificación no sensible a las mayúsculas y minúsculas UCA500R1_LEN_S1, busque todos los nombres que comiencen por 'Bill'.

```
SELECT NAME FROM CUSTDATA WHERE NAME LIKE 'Bill%'
```

  Se devolverán los nombres 'Bill Smith', 'billy simon' y 'BILL JONES'.

Predicado NULL

El predicado NULL comprueba la existencia de valores nulos.

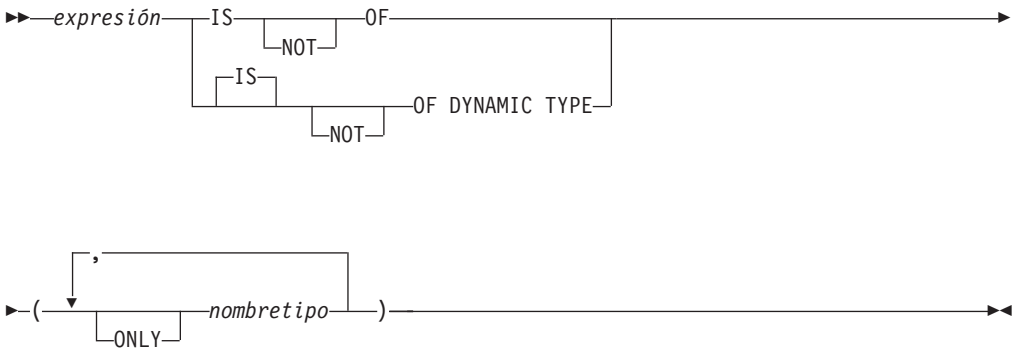
El resultado de un predicado NULL no puede ser desconocido. Si el valor de la expresión es nulo, el resultado es verdadero. Si el valor no es nulo, el resultado es falso. Si se especifica NOT, el resultado se invierte.

Un valor de tipo de fila no puede utilizarse como operando en un predicado NULL, excepto como calificador de un nombre de campo. Si *expresión* es un tipo de fila, se devuelve un error (SQLSTATE 428H2).

Ejemplos

```
PHONENO IS NULL

SALARY IS NOT NULL
```

Predicado TYPE



Un *predicado TYPE* compara el tipo de una expresión con uno o más tipos estructurados definidos por el usuario.

El tipo dinámico de una expresión que implica desreferenciar un tipo de referencia es el tipo real de la fila referenciada de la tabla o vista con tipo de destino. Puede diferenciarse del tipo de destino de una expresión que implica la referencia, denominado el tipo estático de la expresión.

Si el valor de *expresión* es nulo, el resultado del predicado será desconocido. El resultado del predicado será verdadero si el tipo dinámico de la *expresión* es un subtipo de uno de los tipos estructurados especificados por *nombre de tipo*; de lo contrario, el resultado será falso. Si ONLY precede cualquier *nombre de tipo*, no se tienen en cuenta los subtipos correspondientes de este tipo.

Si *nombre de tipo* no está calificado, se resuelve utilizando la vía de acceso de SQL. Cada *nombre de tipo* debe identificar un tipo definido por el usuario que esté en la jerarquía de tipos del tipo estático de *expresión* (SQLSTATE 428DU).

Debe utilizarse la función DEREF siempre que el predicado TYPE tenga una expresión que implique un valor de tipo de referencia. El tipo estático de esta forma de *expresión* es el tipo de destino de la referencia.

La sintaxis IS OF y OF DYNAMIC TYPE son alternativas equivalentes para el predicado TYPE. Asimismo, IS NOT OF y NOT OF DYNAMIC TYPE son alternativas equivalentes.

Ejemplos

Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla, ACTIVITIES, incluye una columna denominada WHO_RESPONSIBLE que está definida como REF(EMP) SCOPE EMPLOYEE. Lo siguiente es un predicado de tipo que devuelve un resultado verdadero cuando una fila correspondiente a WHO_RESPONSIBLE es un director ("manager"):

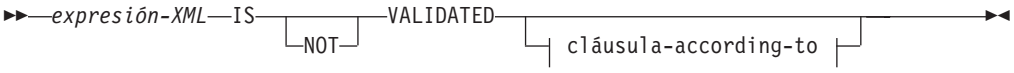
```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

Si una tabla contiene una columna EMPLOYEE de tipo EMP, EMPLOYEE puede contener valores de tipo EMP y también valores de sus subtipos, tales como MGR. El predicado siguiente

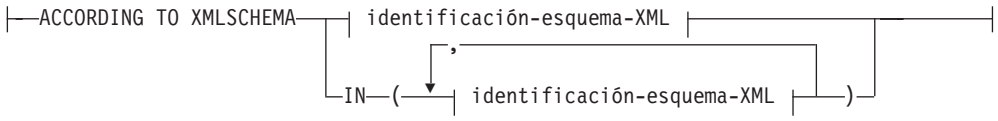
```
EMPL IS OF (MGR)
```

devuelve un resultado verdadero cuando EMPL no es nulo y es realmente un director.

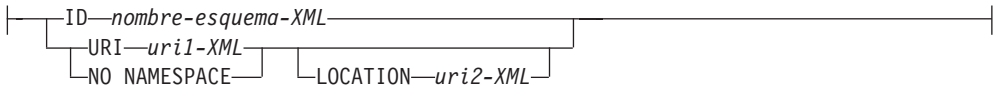
Predicado VALIDATED



cláusula-according-to:



Identificación-esquema-XML:



El predicado VALIDATED comprueba si se ha validado o no el valor especificado por *expresión-XML* utilizando la función XMLVALIDATE. Si el valor especificado es nulo, el resultado de la restricción de validación será desconocido; de lo contrario, el resultado de la restricción de validación será verdadero (true) o falso (false). El valor que especifique debe ser del tipo XML.

Si no se especifica la cláusula ACCORDING TO XMLSCHEMA, los esquemas de XML utilizados para la validación no afectarán al resultado de la restricción de validación.

Descripción

expresión-XML
Especifica el valor XML probado, donde *expresión-XML* puede constar de un documento XML, contenido XML, una secuencia de nodos XML, un *nombre-columna* de XML o un *nombre-correlación* de XML.

Si se especifica un *nombre-columna* de XML, el predicado evalúa si se han validado o no los documentos de XML asociados al nombre de columna especificado.

Consulte "CREATE TRIGGER" para obtener más información sobre la especificación de nombres de correlación del tipo XML como parte de los activadores.

IS VALIDATED o IS NOT VALIDATED
Especifica el estado de validación requerido para el operando *expresión-XML*.

Para que una restricción que especifica IS VALIDATED se evalúe como verdadera (true), el operando deberá haberse validado. Si una cláusula ACCORDING TO XMLSCHEMA opcional incluye uno o varios esquemas XML, el operando debe haberse validado utilizando uno de los esquemas XML identificados.

Para que una restricción que especifica IS NOT VALIDATED se evalúe como falsa (false), el operando deberá estar en estado validado. Si una cláusula ACCORDING TO XMLSCHEMA opcional incluye uno o varios esquemas XML, el operando debe haberse validado utilizando uno de los esquemas XML identificados.

cláusula-according-to
Especifica uno o varios esquemas XML frente a los que el operando debe o no haber sido validado. Únicamente pueden especificarse los esquemas XML registrados previamente con el depósito de esquemas XML.

ACCORDING TO XMLSCHEMA

ID *nombre-esquema-XML*
Especifica un identificador de SQL para el esquema XML. El nombre (incluido el calificador de esquema de SQL implícito o explícito) debe designar de forma exclusiva un esquema XML existente en el depósito de esquema XML en el servidor actual. Si no existe un esquema XML con este nombre en el esquema de SQL especificado explícita o implícitamente, se devuelve un error (SQLSTATE 42704).

URI *uri1-XML*
Especifica el URI del espacio de nombres de destino del esquema XML. El valor de *XML-uri1* especifica un URI como constante de serie de caracteres que no está vacía. El URI debe ser el espacio de nombres de destino de un esquema XML registrado (SQLSTATE 4274A) y, si no se ha especificado una cláusula LOCATION, debe identificar exclusivamente el esquema XML registrado (SQLSTATE 4274B).

NO NAMESPACE
Especifica que el esquema XML no tenga espacio de nombres de destino. El URI del espacio de nombres de destino es equivalente a una serie de caracteres vacía que no se puede especificar como URI de espacio de nombres de destino explícito.

LOCATION *uri2-XML*
Especifica el URI de ubicación del esquema XML del esquema XML. El valor de *XML-uri2* especifica un URI como constante de serie de caracteres que no está vacía. El URI de ubicación del esquema XML, combinado con el URI del espacio de nombres de destino, debe identificar un esquema XML registrado (SQLSTATE 4274A), y sólo debe existir ese esquema XML registrado (SQLSTATE 4274B).

Ejemplos

Ejemplo 1: supongamos que en la tabla T1 se ha definido la columna XMLCOL. Recuperar únicamente los valores XML que hayan sido validados por cualquier esquema XML.

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
```

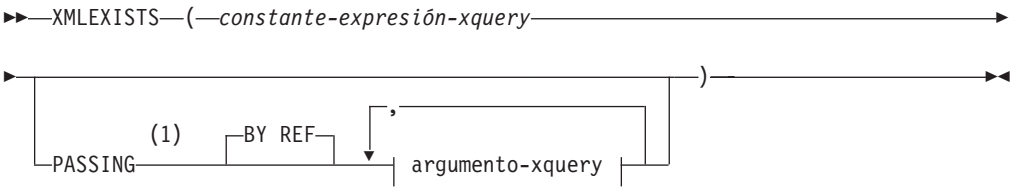
Ejemplo 2: supongamos que en la tabla T1 se ha definido la columna XMLCOL. Imponer la norma que hace que no se puedan insertar ni actualizar valores si no se han validado.

```
ALTER TABLE T1 ADD CONSTRAINT CK_VALIDATED
  CHECK (XMLCOL IS VALIDATED)
```

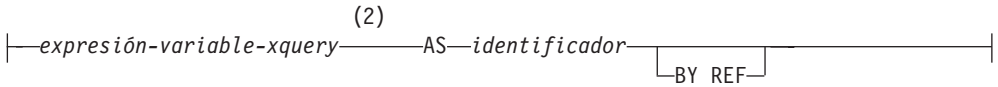
Ejemplo 3: Suponga que desea seleccionar únicamente las filas de la tabla T1 con la columna de XML XMLCOL que se haya validado con el esquema de XML URI http://www.posample.org.

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
  ACCORDING TO XMLSCHEMA URI
   'http://www.posample.org'
```

Predicado XMLEXISTS



argumento-xquery:



Notas:

1 El tipo de datos no puede ser DECFLOAT.

2 El tipo de datos de la expresión no puede ser DECFLOAT.

El predicado XMLEXISTS comprueba si una expresión XQuery devuelve una secuencia de uno o más elementos.

constante-expresión-xquery
Especifica una constante de serie de caracteres de SQL que se interpreta como una expresión XQuery. La serie de la constante se convierte directamente a UTF-8 sin la conversión a la base de datos o la página de códigos de la sección. La expresión XQuery se ejecuta mediante un conjunto opcional de valores de entrada XML, y devuelve una secuencia de salida que se comprueba para determinar el resultado del predicado XMLEXISTS. El valor para la *constante-expresión-xquery* no debe ser una serie vacía o una serie de caracteres en blanco (SQLSTATE 10505).

PASSING
Especifica los valores de entrada y el modo en que pasan a la expresión XQuery especificada por la *constante-expresión-xquery*. Por omisión, cada nombre de columna exclusivo en el ámbito en el que se invoca la función se pasa implícitamente a la expresión XQuery utilizando el nombre de la columna como nombre de variable. Si un *identificador* de un argumento-xquery especificado coincide con el nombre de columna con ámbito, entonces el argumento-xquery explícito se pasa a la expresión XQuery alterando temporalmente dicha columna implícita.

BY REF
Especifica que el mecanismo de pase por omisión es para cualquier *expresión-variable-xquery* de tipo de datos XML. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML.

Esta cláusula no afectará al modo en que se pasan los valores que no son XML. Los valores que no son XML crean una copia nueva del valor durante la conversión a XML.

argumento-xquery
Especifica un argumento que se pasará a la expresión XQuery especificada por *constante-expresión-xquery*. Un argumento especifica un valor y la forma en que ese valor se debe pasar. El argumento contiene una expresión SQL que se evalúa.

- Si el valor del resultado es del tipo XML, pasa a ser un *valor-xml-entrada*. Un valor XML nulo se convierte en una secuencia XML vacía.
- Si el valor del resultado no es del tipo XML, debe ser convertible al tipo de datos XML. Un valor nulo se convierte en una secuencia XML vacía. El valor convertido se transforma en un *valor-xml-entrada*.

Cuando se evalúa la *constante-expresión-xquery*, se presenta un valor igual a *valor-xml-entrada* a una variable XQuery y la cláusula AS especifica un nombre.

expresión-variable-xquery
Especifica una expresión SQL cuyo valor está disponible para la expresión XQuery especificada por *constante-expresión-xquery* durante la ejecución. La expresión no puede contener una referencia de secuencia (SQLSTATE 428F9) ni una función OLAP (SQLSTATE 42903). El tipo de datos de la expresión no puede ser DECFLOAT.

AS *identificador*
Especifica que el valor generado por *expresión-variable-xquery* se pasará a *constante-expresión-xquery* como una variable XQuery. El nombre de la variable será *identificador*. El signo de dólar inicial ($) que precede a los nombres de variable en el lenguaje XQuery no se incluye en el *identificador*. El identificador debe ser un nombre válido de variable XQuery y está restringido a un nombre XML NCName. El identificador no debe tener más de 128 bytes de longitud. Dos argumentos de la misma cláusula PASSING no pueden emplear el mismo identificador (SQLSTATE 42711).

BY REF
Indica que un valor de entrada XML se debe pasar por referencia. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML. Si no se especifica BY REF a continuación de una *expresión-variable-xquery*, los argumentos XML se pasan mediante el mecanismo de pase por omisión que se proporciona mediante la sintaxis situada tras la palabra clave PASSING. Esta opción no puede especificarse para valores que no son XML. Cuando se pasa un valor que no es XML, el valor se convierte a XML; este proceso crea una copia.

Notas

El predicado XMLEXISTS no puede formar parte de:

- Parte de la cláusula ON asociada a un operador JOIN o una sentencia MERGE (SQLSTATE 42972)
- Parte de la cláusula GENERATE KEY USING o RANGE THROUGH de la sentencia CREATE INDEX EXTENSION (SQLSTATE 428E3)
- Parte de la cláusula FILTER USING de la sentencia CREATE FUNCTION (escalar externa) o la cláusula FILTER USING de la sentencia CREATE INDEX EXTENSION (SQLSTATE 428E4)
- Parte de una restricción de comprobación o de una expresión de generación de columnas (SQLSTATE 42621)
- Una cláusula-group-by (SQLSTATE 42822)
- Un argumento de una función-columna (SQLSTATE 42607)

Las sentencias que restringen las subconsultas pueden restringir también un predicado XMLEXISTS que implique una subconsulta.

Ejemplo

```
SELECT c.cid FROM customer c
  WHERE XMLEXISTS('$d/*:customerinfo/*:addr[ *:city = "Aurora" ]'
    PASSING info AS "d")
```

Capítulo 3. Funciones

Resumen de las funciones

Una *función* es una operación que se indica mediante un nombre de función seguido por un par de paréntesis que contienen la especificación de los argumentos (es posible que no haya argumentos).

Las *funciones incorporadas* las proporciona el gestor de bases de datos; devuelven un resultado de un solo valor y se identifican como parte del esquema SYSIBM. Algunas funciones incorporadas son las funciones de columna (por ejemplo, AVG), las funciones con operadores (por ejemplo, "+"), las funciones de conversión (por ejemplo DECIMAL) y otras (como, por ejemplo, SUBSTR).

Las *funciones definidas por el usuario* se registran en una base de datos de SYSCAT.ROUTINES (utilizando la sentencia CREATE FUNCTION). Estas funciones nunca forman parte del esquema SYSIBM. Se proporciona un conjunto de estas funciones con el gestor de bases de datos en un esquema denominado SYSFUN y otro en un esquema denominado SYSPROC.

Las funciones se clasifican como funciones agregadas (de columna), funciones escalares, funciones de fila y funciones de tabla.

- El argumento de una *función agregada* es un conjunto de valores similares. Una función agregada devuelve un solo valor (posiblemente nulo) y puede especificarse en una sentencia de SQL donde sea posible utilizar una expresión.
- Los argumentos de una *función escalar* son valores escalares individuales, que pueden ser de tipos distintos y tener significados diferentes. Una función escalar devuelve un solo valor (posiblemente nulo) y puede especificarse en una sentencia de SQL donde sea posible utilizar una expresión.
- El argumento de una *función de fila* es un tipo estructurado. Una función de fila devuelve una fila de tipos de datos incorporados y sólo se puede especificar como función de transformación para un tipo estructurado.
- Los argumentos de una *función de tabla* son valores escalares individuales, que pueden ser de tipos distintos y tener significados diferentes. Una función de tabla devuelve una tabla a la sentencia de SQL y sólo puede especificarse en la cláusula FROM de una sentencia SELECT.

El nombre de la función, combinado con el esquema, proporciona el nombre totalmente calificado de la función. La combinación del esquema, el nombre de función y los parámetros de entrada constituye una *signatura de función*.

En algunos casos, el tipo de parámetro de entrada se especifica como un tipo de datos incorporados concreto y, en otros casos, se especifica mediante una variable general como *cualquier-tipo-numérico*. Si se especifica un tipo de datos concreto, una coincidencia exacta sólo se obtendrá con el tipo de datos especificado. Si se utiliza una variable general, cada uno de los tipos de datos asociados con dicha variable da como resultado una coincidencia exacta.

Puede que existan funciones adicionales, porque las funciones definidas por el usuario pueden crearse en esquemas distintos, usando como fuente una de las signaturas de función. También es posible crear funciones externas en las aplicaciones.

Funciones soportadas y vistas y rutinas administrativas SQL

Este tema lista las funciones incorporadas a las que se proporciona soporte clasificadas por tipos:

- Funciones agregadas (Tabla 32)
- Funciones de matriz (Tabla 33 en la página 323)
- Funciones escalares de conversión (Tabla 34 en la página 324)
- Funciones escalares de fecha y hora (Tabla 35 en la página 325)
- Funciones escalares diversas (Tabla 36 en la página 327)
- Funciones escalares numéricas (Tabla 37 en la página 328)
- Funciones escalares de particionamiento (Tabla 38 en la página 329)
- Funciones escalares de seguridad (Tabla 39 en la página 330)
- Funciones escalares de series (Tabla 40 en la página 330)
- Funciones de tabla (Tabla 41 en la página 332)
- Funciones XML (Tabla 42 en la página 332)

Para ver las listas de vistas y rutinas administrativas SQL soportadas clasificadas por funcionalidad, consulte "Vistas y rutinas administrativas SQL soportadas" en *Rutinas y vistas administrativas* . Estas rutinas y vistas están agrupadas como indicamos a continuación:

- Rutinas administrativas SQL de supervisión de actividad
- Procedimiento ADMIN_CMD almacenado y rutinas administrativas SQL asociadas
- Vistas y rutinas administrativas SQL de la configuración
- Vistas administrativas del entorno
- Rutinas administrativas SQL de instantáneas de salud
- Rutinas administrativas SQL de MQSeries
- Vistas y rutinas administrativas SQL de la seguridad
- Vistas y rutinas administrativas SQL de instantáneas
- Rutinas administrativas SQL de procedimientos SQL
- Rutinas administrativas SQL de redistribución paso a paso
- Rutinas administrativas SQL de la herramienta de administración del almacenamiento
- Vistas y rutinas administrativas SQL varias

Tabla 32. Funciones agregadas

Función	Descripción
"ARRAY_AGG" en la página 335	Agrega un conjunto de elementos a una matriz.
"AVG" en la página 337	Devuelve el promedio de un conjunto de números.
"CORRELATION" en la página 339	Devuelve el coeficiente de correlación de un conjunto de pares de números.
"COUNT" en la página 340	Devuelve el número de filas o de valores de un conjunto de filas o de valores.
"COUNT_BIG" en la página 341	Devuelve el número de filas o de valores de un conjunto de filas o de valores. El resultado puede ser mayor que el valor máximo de INTEGER.

Tabla 32. Funciones agregadas (continuación)

Función	Descripción
"COVARIANCE" en la página 343	Devuelve la covarianza de un conjunto de pares de números.
"GROUPING" en la página 344	Se utiliza con conjuntos-agrupaciones y super-grupos para indicar el subtotal de filas generadas por un conjunto de agrupaciones. El valor que se devuelve es 0 ó 1. Si se devuelve 1, el valor del argumento de la fila devuelta es nulo y se ha generado para un conjunto de agrupaciones. Esta fila generada proporciona un subtotal correspondiente a un conjunto de agrupaciones.
"MAX" en la página 346	Devuelve el valor máximo de un conjunto de valores.
"MIN" en la página 348	Devuelve el valor mínimo de un conjunto de valores.
"Funciones de regresión" en la página 349	La función agregada REGR_AVGX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 349	La función agregada REGR_AVGY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 349	La función agregada REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión.
"Funciones de regresión" en la página 349	La función agregada REGR_INTERCEPT o REGR_ICPT devuelve la intersección y de la línea de regresión.
"Funciones de regresión" en la página 349	La función agregada REGR_R2 devuelve el coeficiente de determinación de la regresión.
"Funciones de regresión" en la página 349	La función agregada REGR_SLOPE devuelve la inclinación de la línea.
"Funciones de regresión" en la página 349	La función agregada REGR_SXX devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 349	La función agregada REGR_SXY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"Funciones de regresión" en la página 349	La función agregada REGR_SYY devuelve las cantidades utilizadas para calcular las estadísticas de diagnóstico.
"STDDEV" en la página 352	Devuelve la desviación estándar de un conjunto de números.
"SUM" en la página 353	Devuelve la suma de un conjunto de números.
"VARIANCE" en la página 354	Devuelve la varianza de un conjunto de números.
"XMLAGG" en la página 355	Devuelve una secuencia XML que contiene un elemento por cada valor que no sea nuevo de un conjunto de valores XML.
"XMLGROUP" en la página 357	Devuelve un valor XML con un único nodo de documento XQuery que contiene un nodo de elemento de nivel superior.

Tabla 33. Funciones de matriz

Función	Descripción
"ARRAY_AGG" en la página 335	Agrega un conjunto de elementos a una matriz.
"ARRAY_DELETE" en la página 365	Suprime un elemento o rango de elementos de una matriz asociativa.

Tabla 33. Funciones de matriz (continuación)

Función	Descripción
"ARRAY_FIRST" en la página 366	Devuelve el valor del índice más bajo de la matriz.
"ARRAY_LAST" en la página 367	Devuelve el valor del índice más alto de la matriz.
"ARRAY_NEXT" en la página 368	Devuelve el siguiente valor de índice de matriz más alto para una matriz relativa al argumento de índice de la matriz especificada.
"ARRAY_PRIOR" en la página 369	Devuelve el siguiente valor de índice de matriz más bajo para una matriz relativa al argumento de índice de la matriz especificada.
"CARDINALITY" en la página 381	Devuelve un valor de tipo BIGINT que representa el número de elementos de una matriz.
"MAX_CARDINALITY" en la página 501	Devuelve un valor de tipo BIGINT que representa el número máximo de elementos que puede contener una matriz.
"TRIM_ARRAY" en la página 620	Devuelve un valor con el mismo tipo de matriz que *variable-matriz* pero con la cardinalidad reducida por el valor de *expresión-numérica*.
"UNNEST" en la página 705	Devuelve una tabla de resultados que incluye una fila para cada elemento de la matriz especificada.

Tabla 34. Funciones escalares de conversión

Función	Descripción
"BIGINT" en la página 375	Devuelve una representación en el formato de un entero de 64 bits de un valor en el formato de una constante entera.
"BLOB" en la página 380	Devuelve una representación BLOB de una serie de cualquier tipo.
"CHAR" en la página 383	Devuelve una representación CHARACTER de un valor.
"CLOB" en la página 392	Devuelve una representación CLOB de un valor.
"DATE" en la página 404	Devuelve una representación DATE de un valor.
"DBCLOB" en la página 411	Devuelve una representación DBCLOB de una serie.
"DECFLOAT" en la página 414	Devuelve la representación de coma flotante decimal de un valor.
"DECIMAL o DEC" en la página 419	Devuelve una representación DECIMAL de un valor.
"DOUBLE_PRECISION o DOUBLE" en la página 431	Devuelve la representación de coma flotante de un valor.
Funciones escalares EMPTY_BLOB, EMPTY_CLOB y EMPTY_DBCLOB	Devuelve un valor de longitud cero del tipo de datos asociado.
"FLOAT" en la página 442	Devuelve una representación FLOAT de un valor.
"GRAPHIC" en la página 447	Devuelve una representación GRAPHIC de una serie.
"INTEGER o INT" en la página 470	Devuelve una representación INTEGER de un valor.
"REAL" en la página 537	Devuelve la representación de coma flotante de precisión simple de un valor.

Tabla 34. Funciones escalares de conversión (continuación)

Función	Descripción
"SMALLINT" en la página 574	Devuelve una representación SMALLINT de un valor.
"TIME" en la página 595	Devuelve una representación TIME de un valor.
"TIMESTAMP" en la página 596	Devuelve una representación TIMESTAMP a partir de un valor o de un par de valores.
"TO_CLOB" en la página 609	Devuelve una representación CLOB de un tipo de serie de caracteres.
"VARCHAR" en la página 636	Devuelve una representación VARCHAR de un valor.
"VARGRAPHIC" en la página 652	Devuelve una representación VARGRAPHIC de un valor.

Tabla 35. Funciones escalares de fecha y hora

Función	Descripción
"ADD_MONTHS" en la página 363	Devuelve un valor de fecha y hora que representa la *expresión* más un número de meses especificado.
"DAY" en la página 405	Devuelve la parte del día del valor.
"DAYNAME" en la página 406	Devuelve una serie de caracteres que contiene el nombre del día (por ejemplo, viernes) para la parte del día de la expresión, basada en el nombre-entorno-local o el valor del registro especial CURRENT LOCALE LC_TIME.
"DAYOFWEEK" en la página 407	Devuelve el día de la semana a partir de un valor, donde el 1 es el domingo y el 7 es el sábado.
"DAYOFWEEK_ISO" en la página 408	Devuelve el día de la semana a partir de un valor, donde el 1 es el lunes y el 7 es el domingo.
"DAYOFYEAR" en la página 409	Devuelve el día del año a partir de un valor.
"DAYS" en la página 410	Devuelve una representación entera de una fecha.
"EXTRACT" en la página 439	Devuelve una parte de una fecha o indicación de fecha y hora basada en los argumentos.
"HOUR" en la página 457	Devuelve la parte de la hora de un valor.
"JULIAN_DAY" en la página 472	Devuelve un valor entero que representa el número de días desde el 1 de enero de 4712 A.C. hasta la fecha especificada en el argumento.
"LAST_DAY" en la página 473	Devuelve un valor de fecha y hora que representa el último día del mes del argumento.
"MICROSECOND" en la página 502	Devuelve la parte correspondiente a las milésimas de segundo de un valor.
"MIDNIGHT_SECONDS" en la página 503	Devuelve un valor entero que representa el número de segundos entre medianoche y un valor de hora especificado.
"MINUTE" en la página 505	Devuelve la parte del minuto de un valor.
"MONTH" en la página 507	Devuelve la parte del mes de un valor.
"MONTHNAME" en la página 508	Devuelve una serie de caracteres que contiene el nombre del mes (por ejemplo, enero) para la parte del mes de la expresión, basada en el nombre-entorno-local o el valor del registro especial CURRENT LOCALE LC_TIME.

Tabla 35. Funciones escalares de fecha y hora (continuación)

Función	Descripción
"MONTHS_BETWEEN" en la página 510	Devuelve una estimación del número de meses entre *expresión1* y *expresión2*.
"NEXT_DAY" en la página 514	Devuelve un valor de fecha y hora que representa el primer día de la semana, indicado mediante *expresión-serie*, que es posterior a la fecha indicada en *expresión*.
"QUARTER" en la página 533	Devuelve un entero que representa el trimestre del año en el que reside una fecha.
"ROUND" en la página 552	Devuelve un valor de fecha y hora, redondeado a la unidad especificada por una *serie-formato*.
"ROUND_TIMESTAMP" en la página 559	Devuelve una indicación de fecha y hora que es la *expresión* redondeada a la unidad especificada por la *serie-formato*.
"SECOND" en la página 569	Devuelve la segunda parte de un valor.
"TIMESTAMP_FORMAT" en la página 598	Devuelve una indicación de fecha y hora a partir de una serie de caracteres (*argumento1*) que se ha interpretado utilizando una plantilla de formato (*argumento2*).
"TIMESTAMP_ISO" en la página 605	Devuelve un valor de indicación de fecha y hora basado en un argumento de fecha, de hora o de indicación de fecha y hora. Si el argumento es una fecha, inserta ceros para todos los elementos de hora. Si el argumento es una hora, inserta el valor de CURRENT DATE para los elementos de fecha y ceros para el elemento de fracción de hora.
"TIMESTAMPDIFF" en la página 606	Devuelve un número estimado de intervalos de tipo *argumento1* basado en la diferencia entre dos indicaciones de fecha y hora. El segundo argumento es el resultado de restar dos tipos de indicaciones de fecha y hora y de convertir el resultado en CHAR.
"TO_CHAR" en la página 608	Devuelve una representación CHARACTER de una indicación de fecha y hora.
"TO_DATE" en la página 610	Devuelve una indicación de fecha y hora a partir de una serie de caracteres.
"TO_TIMESTAMP" en la página 612	Devuelve una indicación de fecha y hora basada en la interpretación de la serie de entrada utilizando el formato especificado.
"TRUNCATE o TRUNC" en la página 623	Devuelve un valor de fecha y hora, truncado a la unidad especificada por una *serie-formato*.
"TRUNC_TIMESTAMP" en la página 621	Devuelve una indicación de fecha y hora que es la *expresión* truncada en la unidad especificada por la *serie-formato*.
"VARCHAR_FORMAT" en la página 643	Devuelve una representación CHARACTER de una indicación de fecha y hora (*argumento1*) con el formato indicado en una plantilla *argumento2*).
"WEEK" en la página 658	Devuelve la semana del año a partir de un valor, donde la semana empieza el domingo.
"WEEK_ISO" en la página 659	Devuelve la semana del año a partir de un valor, donde la semana empieza el lunes.
"YEAR" en la página 702	Devuelve la parte del año de un valor.

Tabla 36. Funciones escalares diversas

Función	Descripción
"BITAND, BITANDNOT, BITOR, BITXOR y BITNOT" en la página 377	Estas funciones a nivel de bit operan en la representación de "complemento del dos" del valor entero de los argumentos de entrada y devuelven el resultado como un correspondiente valor entero de base 10 en un tipo de datos basándose en el tipo de datos de los argumentos de entrada.
"COALESCE" en la página 393	Devuelve el primer argumento que no es nulo.
"CURSOR_ROWCOUNT" en la página 402	Devuelve el número acumulado de todas las filas captadas por el cursor especificado desde que dicho cursor se abrió.
"DECODE" en la página 423	Compara cada *expresión2* especificada con la *expresión1*. Si *expresión1* es igual a *expresión2*, o *expresión1* y *expresión2* son nulas, se devuelve el valor de la *expresión-resultado* siguiente. Si ninguna *expresión2* coincide con *expresión1*, se devuelve el valor de *expresión-else*; de lo contrario, se devuelve un valor nulo.
"DEREF" en la página 428	Devuelve una instancia del tipo de destino del argumento del tipo de referencia.
"EVENT_MON_STATE" en la página 437	Devuelve el estado operativo de un determinado supervisor de sucesos.
"GREATEST" en la página 452	Devuelve el valor máximo de un conjunto de valores.
"HEX" en la página 455	Devuelve una representación hexadecimal de un valor.
"IDENTITY_VAL_LOCAL" en la página 458	Devuelve el valor asignado más reciente correspondiente a una columna de entidad.
"LEAST" en la página 476	Devuelve el valor mínimo de un conjunto de valores.
"LENGTH" en la página 480	Devuelve la longitud de un valor.
"MAX" en la página 500	Devuelve el valor máximo de un conjunto de valores.
"MIN" en la página 504	Devuelve el valor mínimo de un conjunto de valores.
"NULLIF" en la página 517	Devuelve un valor nulo si los argumentos son iguales; de lo contrario, devuelve el valor del primer argumento.
"NVL" en la página 518	Devuelve el primer argumento que no es nulo.
"RAISE_ERROR" en la página 535	Emite un error a la SQLCA. El sqlstate que se devolverá se indica mediante *argumento1*. El segundo argumento contiene el texto que debe devolverse.
"REC2XML" en la página 539	Devuelve una serie formateada codificada en XML que contiene nombres de columna y datos de columna.
"RID_BIT y RID" en la página 547	La función escalar RID_BIT devuelve el identificador de una fila (RID) en un formato de serie de caracteres. La función escalar RID devuelve el RID de una fila en formato de entero largo. La función RID no se soporta en entornos de bases de datos particionadas. Se prefiere la función RID_BIT a la función RID.
"TABLE_NAME" en la página 589	Devuelve un nombre no calificado de una tabla o vista, basado en el nombre de objeto especificado en *argumento1* y en el nombre de esquema opcional especificado en *argumento2*. El valor devuelto se utiliza para resolver los alias.

Tabla 36. Funciones escalares diversas (continuación)

Función	Descripción
"TABLE_SCHEMA" en la página 591	Devuelve la parte correspondiente al nombre de esquema de un nombre de tabla o de vista de dos partes (especificado por el nombre del objeto en *argumento1* y por el nombre de esquema opcional en *argumento2*). El valor devuelto se utiliza para resolver los alias.
"TYPE_ID" en la página 627	Devuelve el identificador interno de tipo de datos del tipo de datos dinámico del argumento. El resultado de esta función no se puede transportar entre bases de datos.
"TYPE_NAME" en la página 628	Devuelve el nombre no calificado del tipo de datos dinámico del argumento.
"TYPE_SCHEMA" en la página 629	Devuelve el nombre del esquema del tipo de datos dinámico del argumento.
"VALUE" en la página 635	Devuelve el primer argumento que no es nulo.

Tabla 37. Funciones escalares numéricas

Función	Descripción
"ABS o ABSVAL" en la página 361	Devuelve el valor absoluto de un número.
"ACOS" en la página 362	Devuelve el arcocoseno de un número, en radianes.
"ASIN" en la página 371	Devuelve el arcoseno de un número, en radianes.
"ATAN" en la página 372	Devuelve la arcotangente de un número, en radianes.
"ATANH" en la página 374	Devuelve la arcotangente hiperbólica de un número, en radianes.
"ATAN2" en la página 373	Devuelve la arcotangente de las coordenadas x e y como un ángulo expresado en radianes.
"CEILING o CEIL" en la página 382	Devuelve el valor del entero más pequeño que es mayor o igual que un número.
"COMPARE_DECFLOAT" en la página 396	Devuelve un valor SMALLINT que indica si los dos argumentos son iguales o están desordenados o si un argumento es mayor que el otro.
"COS" en la página 399	Devuelve el coseno de un número.
"COSH" en la página 400	Devuelve el coseno hiperbólico de un número.
"COT" en la página 401	Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.
"DECFLOAT_FORMAT" en la página 416	Devuelve un DECFLOAT(34) de una serie de caracteres.
"DEGREES" en la página 427	Devuelve el número de grados de un ángulo.
"DIGITS" en la página 430	Devuelve la representación en el formato de una serie de caracteres del valor absoluto de un número.
"EXP" en la página 438	Devuelve un valor que es la base del logaritmo natural (e) elevada a la potencia especificada por el argumento.
"FLOOR" en la página 443	Devuelve el valor del entero más grande que es menor o igual que un número.
"LN" en la página 482	Devuelve el logaritmo natural de un número.
"LOG10" en la página 490	Devuelve el logaritmo común (en base 10) de un número.

Tabla 37. Funciones escalares numéricas (continuación)

Función	Descripción
"MOD" en la página 506	Devuelve el resto del primer argumento dividido por el segundo argumento.
"MULTIPLY_ALT" en la página 512	Devuelve el producto de dos argumentos como un valor decimal. Esta función resulta útil cuando la suma de las precisiones del argumento es mayor que 31.
"NORMALIZE_DECFLOAT" en la página 516	Devuelve un valor de coma flotante decimal que es el resultado del argumento establecido en su formato más simple.
"POWER" en la página 530	Devuelve el resultado de elevar el primer argumento a la potencia del segundo argumento.
"QUANTIZE" en la página 531	Devuelve un número de coma flotante decimal que es igual en valor y signo al primer argumento y cuyo exponente es igual al exponente del segundo argumento.
"RADIANS" en la página 534	Devuelve el número de radianes de un argumento que se expresa en grados.
"RAND" en la página 536	Devuelve un número aleatorio.
"ROUND" en la página 552	Devuelve un valor numérico que se ha redondeado el número de posiciones decimales especificado.
"SIGN" en la página 571	Devuelve el signo de un número.
"SIN" en la página 572	Devuelve el seno de un número.
"SINH" en la página 573	Devuelve el seno hiperbólico de un número.
"SQRT" en la página 577	Devuelve la raíz cuadrada de un número.
"TAN" en la página 593	Devuelve la tangente de un número.
"TANH" en la página 594	Devuelve la tangente hiperbólica de un número.
"TO_NUMBER" en la página 611	Devuelve un DECFLOAT(34) de una serie de caracteres.
"TOTALORDER" en la página 613	Devuelve un valor SMALLINT de -1, 0 o 1 que indica el orden de comparación de dos argumentos.
"TRUNCATE o TRUNC" en la página 623	Devuelve un valor numérico que se ha truncado en el número de posiciones decimales especificado.

Tabla 38. Funciones escalares de particionamiento

Función	Descripción
"DATAPARTITIONNUM" en la página 403	Devuelve el número de secuencia (SYSDATAPARTITIONS.SEQNO) de la partición de datos donde reside la fila. El argumento es cualquier nombre de columna dentro de la tabla.
"DBPARTITIONNUM" en la página 412	Devuelve el número de partición de base de datos de la fila. El argumento es cualquier nombre de columna dentro de la tabla.
"HASHEDVALUE" en la página 453	Devuelve el índice de correlación de distribución (de 0 a 32767) de la fila. El argumento es un nombre de columna dentro de una tabla.

Tabla 39. Funciones escalares de seguridad

Función	Descripción
"SECLABEL" en la página 565	Devuelve una etiqueta de seguridad sin nombre.
"SECLABEL_BY_NAME" en la página 566	Devuelve una etiqueta de seguridad específica.
"SECLABEL_TO_CHAR" en la página 567	Acepta una etiqueta de seguridad y devuelve una serie que contiene todos los elementos de la etiqueta de seguridad.

Tabla 40. Funciones escalares de series

Función	Descripción
"ASCII" en la página 370	Devuelve el valor en código ASCII del carácter que hay más a la izquierda del argumento como un entero.
"CHARACTER_LENGTH" en la página 389	Devuelve la longitud de una expresión en la *unidad-serie* especificada.
"CHR" en la página 391	Devuelve el carácter que tiene el valor de código ASCII especificado por el argumento.
"COLLATION_KEY_BIT" en la página 394	Devuelve una serie VARCHAR FOR BIT DATA que representa la clave de clasificación de la *expresión-serie* especificada en el *nombre-clasificación* especificado.
"CONCAT" en la página 398	Devuelve una serie que es la concatenación de dos series.
"DECRYPT_BIN y DECRYPT_CHAR" en la página 425	Devuelve un valor que es el resultado de descifrar datos cifrados utilizando una serie de contraseña.
"DIFFERENCE" en la página 429	Devuelve la diferencia entre el sonido de las palabras en dos series de argumento según se determine mediante la función SOUNDEX. El valor 4 significa que las series suenan igual.
"ENCRYPT" en la página 434	Devuelve un valor que es el resultado de cifrar una expresión de serie de datos.
"GENERATE_UNIQUE" en la página 444	Devuelve una serie de caracteres de datos de bits que es exclusivo comparado con cualquier otra ejecución de la misma función.
"GETHINT" en la página 446	Devuelve la contraseña sugerida, si se encuentra una.
"INITCAP" en la página 463	Devuelve una serie con el primer carácter de cada palabra en mayúscula y el resto en minúsculas.
"INSERT" en la página 465	Devuelve una serie en la que *argumento3* bytes se han suprimido de *argumento1* (comenzando por *argumento2*) y en la que *argumento4* bytes se han insertado en *argumento1* (comenzando por *argumento2*).
"INSTR" en la página 469	Devuelve la posición inicial de una serie dentro de otra serie.
"LCASE" en la página 474	Devuelve una serie en la que todos los caracteres SBCS se han convertido a minúsculas.
"LCASE (sensible al entorno local)" en la página 475	Devuelve una serie en la que todos los caracteres se han convertido a minúsculas utilizando las normas del estándar Unicode asociadas al entorno local especificado.
"LOWER (sensible al entorno local)" en la página 494	Devuelve una serie en la que todos los caracteres se han convertido a minúsculas utilizando las normas del estándar Unicode asociadas al entorno local especificado.

Tabla 40. Funciones escalares de series (continuación)

Función	Descripción
"LEFT" en la página 477	Devuelve los caracteres situados más a la izquierda de una serie.
"LOCATE" en la página 483	Devuelve la posición inicial de una serie contenida en otra serie.
"LOCATE_IN_STRING" en la página 487	Devuelve la posición inicial de la primera ocurrencia de una serie dentro de otra serie.
"LOWER" en la página 493	Devuelve una serie en la que todos los caracteres se han convertido a caracteres en minúsculas.
"LPAD" en la página 496	Devuelve una serie que está rellenada en el lado izquierdo con el carácter especificado o con espacios en blanco.
"LTRIM" en la página 499	Elimina los blancos del principio de una expresión de serie.
"OCTET_LENGTH" en la página 519	Devuelve la longitud de una expresión en octetos (bytes).
"OVERLAY" en la página 520	Devuelve una serie en la que, a partir del *inicio* de la *serie-fuente* especificada, se ha suprimido la *longitud* de las unidades de código especificadas y se ha insertado la *serie-inserción*.
"POSITION" en la página 525	Devuelve la posición inicial del *argumento2* en *argumento1*.
"POSSTR" en la página 528	Devuelve la posición inicial de una serie contenida en otra serie.
"REPEAT" en la página 544	Devuelve una serie de caracteres compuesta por *argumento1* repetido *argumento2* veces.
"REPLACE" en la página 545	Sustituye todas las apariciones de *argumento2* en *argumento1* por *argumento3*.
"RIGHT" en la página 549	Devuelve los caracteres situados más a la derecha de una serie.
"RPAD" en la página 561	Devuelve una serie que está rellenada en el lado derecho con el carácter, serie o espacios en blanco especificados.
"RTRIM" en la página 564	Elimina los blancos del final de una expresión de serie.
"SOUNDEX" en la página 575	Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. Este resultado se puede comparar con el sonido de otras series.
"SPACE" en la página 576	Devuelve una serie de caracteres formada por el número de blancos especificado.
"STRIP" en la página 578	Elimina de una expresión de serie, los espacios en blanco iniciales o de cola o bien otros caracteres iniciales o de cola especificados.
"SUBSTR" en la página 580	Devuelve una subserie de una serie.
"SUBSTRB" en la página 583	Devuelve una subserie de una serie.
"SUBSTRING" en la página 586	Devuelve una subserie de una serie.
"TRANSLATE" en la página 615	Devuelve una serie en la que uno o más caracteres de una serie se han convertido en otros caracteres.
"TRIM" en la página 618	Elimina de una expresión de serie, los espacios en blanco iniciales o de cola o bien otros caracteres iniciales o de cola especificados.

Tabla 40. Funciones escalares de series (continuación)

Función	Descripción
"UCASE" en la página 630	La función UCASE es idéntica a la función TRANSLATE excepto en que sólo se especifica el primer argumento (*exp-serie-car*).
"UCASE (sensible al entorno local)" en la página 631	Devuelve una serie en la que todos los caracteres se han convertido a mayúsculas utilizando las normas del estándar Unicode asociadas al entorno local especificado.
"UPPER" en la página 632	Devuelve una serie en la que todos los caracteres se han convertido a mayúsculas.
"UPPER (sensible al entorno local)" en la página 633	Devuelve una serie en la que todos los caracteres se han convertido a mayúsculas utilizando las normas del estándar Unicode asociadas al entorno local especificado.

Tabla 41. Funciones de tabla

Función	Descripción
"BASE_TABLE" en la página 703	Devuelve tanto el nombre de objeto como el nombre de esquema del objeto encontrado después de resolverse todas las cadenas de alias.
"UNNEST" en la página 705	Devuelve una tabla de resultados que incluye una fila para cada elemento de la matriz especificada.
"XMLTABLE" en la página 707	Devuelve una tabla a partir de la evaluación de expresiones XQuery, posiblemente utilizando argumentos de entrada especificados como variables XQuery. Cada elemento de la secuencia de resultados de la expresión XQuery de fila representa una fila de la tabla de resultados.

Tabla 42. Funciones XML

Función	Descripción
"PARAMETER" en la página 524	Representa una posición en una sentencia de SQL en la que XQuery proporciona dinámicamente el valor como parte de la invocación de la función db2-fn:sqlquery.
"XMLAGG" en la página 355	Devuelve una secuencia XML que contiene un elemento por cada valor que no sea nuevo de un conjunto de valores XML.
"XMLATTRIBUTES" en la página 660	Construye los atributos XML a partir de los argumentos.
"XMLCOMMENT" en la página 662	Devuelve un valor XML con un único nodo de comentario XQuery con el argumento de entrada como contenido.
"XMLCONCAT" en la página 663	Devuelve una secuencia que contiene la concatenación de un número variable de argumentos de entrada de XML.
"" en la página 665	Devuelve un valor XML con un único nodo de documento XQuery con ninguno o varios nodos hijo.
"XMLELEMENT" en la página 667	Devuelve un valor XML que es un nodo de elemento XML.
"XMLFOREST" en la página 674	Devuelve un valor XML que es una secuencia de nodos de elemento XML.
"XMLGROUP" en la página 357	Devuelve un valor XML con un único nodo de documento XQuery que contiene un nodo de elemento de nivel superior.

Tabla 42. Funciones XML (continuación)

Función	Descripción
"XMLNAMESPACES" en la página 677	Construye las declaraciones de espacios de nombres a partir de los argumentos.
"XMLPARSE" en la página 679	Analiza el argumento como un documento XML y devuelve un valor XML.
"XMLPI" en la página 682	Devuelve un valor XML con un único nodo de instrucción de proceso XQuery.
"XMLQUERY" en la página 683	Devuelve un valor XML a partir de la evaluación de una expresión XQuery posiblemente utilizando los argumentos de entrada especificados como variables XQuery.
"XMLROW" en la página 686	Devuelve un valor XML con un único nodo de documento XQuery que contiene un nodo de elemento de nivel superior.
"XMLSERIALIZE" en la página 689	Devuelve un valor XML serializado de los tipos de datos especificados, generados a partir del argumento.
"XMLTABLE" en la página 707	Devuelve una tabla a partir de la evaluación de expresiones XQuery, posiblemente utilizando argumentos de entrada especificados como variables XQuery. Cada elemento de la secuencia de resultados de la expresión XQuery de fila representa una fila de la tabla de resultados.
"XMLTEXT" en la página 691	Devuelve un valor XML con un único nodo de texto XQuery cuyo contenido es el argumento de entrada.
"XMLVALIDATE" en la página 693	Devuelve una copia del valor XML de entrada aumentado con la información obtenida a partir de la validación de esquema XML, incluidos los valores por omisión.
"XMLXSROBJECTID" en la página 697	Devuelve un identificador de objeto XSR del esquema XML utilizado para validar el documento XML especificado en el argumento
"XSLTRANSFORM" en la página 698	Convierte datos XML a otros formatos, incluyendo la conversión de documentos XML que se ajustan a un esquema XML en documentos que se ajustan a otro esquema.

Funciones agregadas

El argumento de una función agregada es un conjunto de valores derivados de una expresión. La expresión puede incluir columnas, pero no puede incluir una *selección-completa-escalar*, otra función de columna ni una expresión XMLQUERY o XMLEXISTS (SQLSTATE 42607). El ámbito del conjunto es un grupo o una tabla resultante intermedia.

Si se especifica una cláusula GROUP BY en una consulta, y el resultado intermedio de las cláusulas FROM, WHERE, GROUP BY y HAVING es el conjunto vacío, las funciones agregadas no se aplican; el resultado de la consulta es el conjunto vacío; SQLCODE se establece en +100 y SQLSTATE se establece en '02000'.

Si *no* se especifica una cláusula GROUP BY en una consulta, y el resultado intermedio de las cláusulas FROM, WHERE y HAVING es el conjunto vacío, las funciones agregadas se aplican al conjunto vacío.

Por ejemplo, el resultado de la siguiente sentencia SELECT es el número de valores diferenciado de JOBCODE para los empleados en el departamento D01:

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

La palabra clave DISTINCT no se considera un argumento de una función, sino una especificación de una operación que se realiza antes de aplicar la función. Si se especifica DISTINCT, se eliminan los valores duplicados. Cuando se interpreta la cláusula DISTINCT para los valores de coma flotante decimal que sean numéricamente iguales, no se tiene en cuenta el número de dígitos significativos del valor. Por ejemplo, el número de coma flotante decimal 123.00 no es diferente del número de coma flotante decimal 123. La representación del número devuelto de la consulta será cualquiera de las representaciones que se encuentre (por ejemplo, 123.00 ó 123).

Si se especifica ALL implícita o explícitamente, no se eliminan los valores duplicados.

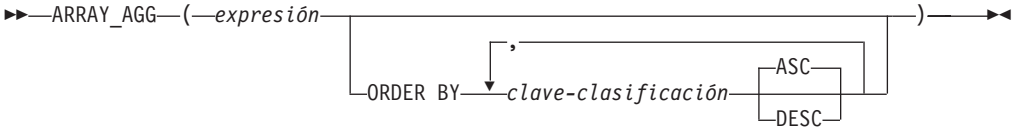
Por compatibilidad con otras implementaciones de SQL, se puede especificar UNIQUE como sinónimo para DISTINCT en funciones agregadas.

Se pueden utilizar expresiones en las funciones agregadas. Por ejemplo:

```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

Las funciones agregadas se pueden calificar con un nombre de esquema (por ejemplo, SYSIBM.COUNT(*)).

ARRAY_AGG



El esquema es SYSIBM.

La función ARRAY_AGG agrega un conjunto de elementos a una matriz. El tipo de datos de la expresión debe ser un tipo de datos que pueda especificarse en una sentencia CREATE TYPE (matriz) (SQLSTATE 429C2).

Si se especifica *clave-clasificación*, determina el orden de los elementos agregados en la matriz. Si no se especifica *clave-clasificación*, el orden de los elementos de la matriz no será determinante. Si no se especifica *clave-clasificación* y se especifica ARRAY_AGG más de una vez en la misma cláusula SELECT, se utilizará el mismo orden de elementos de la matriz para cada resultado de ARRAY_AGG.

Si una cláusula SELECT tiene varias ocurrencias de XMLAGG o ARRAY_AGG que especifiquen *clave-clasificación*, todas las claves de clasificación deberán ser idénticas (SQLSTATE 428GZ).

La función ARRAY_AGG sólo puede especificarse en un procedimiento de SQL de los siguientes contextos específicos (SQLSTATE 42887):

- La lista de selección de una sentencia SELECT INTO
- La lista de selección de una selección completa de la definición de un cursor que no pueda desplazarse
- La lista de selección de una subconsulta escalar a la derecha de una sentencia SET

ARRAY_AGG no puede utilizarse como parte de una función OLAP (SQLSTATE 42887). La sentencia SELECT que utiliza ARRAY_AGG no puede contener una cláusula ORDER BY o una cláusula DISTINCT y la cláusula SELECT o la cláusula HAVING no pueden contener una subconsulta o llamar a una función SQL (SQLSTATE 42887).

ARRAY_AGG no puede utilizarse para generar una matriz asociativa o una matriz con un tipo de datos de elemento de fila (SQLSTATE 42846).

Ejemplo:

- Dado el siguiente DDL:

```
CREATE TYPE PHONELIST AS DECIMAL(10, 0)ARRAY[10]

CREATE TABLE EMPLOYEE (
  ID          INTEGER NOT NULL,
  PRIORITY    INTEGER NOT NULL,
  PHONENUMBER DECIMAL(10, 0),
  PRIMARY KEY(ID, PRIORITY))
```

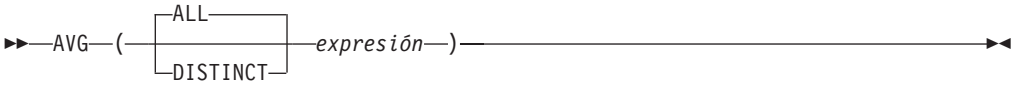
  Cree un procedimiento que utilice una sentencia SELECT INTO para devolver la lista priorizada de los números de contacto en los que se puede contactar con un empleado.

```
CREATE PROCEDURE GETPHONENUMBERS
  (IN  EMPID   INTEGER,
   OUT NUMBERS PHONELIST)
BEGIN
  SELECT ARRAY_AGG(PHONENUMBER ORDER BY PRIORITY)
    INTO NUMBERS
  FROM EMPLOYEE
  WHERE ID = EMPID;
END
```

Cree un procedimiento que utilice una sentencia SET para devolver la lista de los números de contacto de un empleado en orden aleatorio.

```
CREATE PROCEDURE GETPHONENUMBERS
   (IN  EMPID   INTEGER,
    OUT NUMBERS PHONELIST)
BEGIN
  SET NUMBERS =
    (SELECT ARRAY_AGG(PHONENUMBER)
    FROM EMPLOYEE
    WHERE ID = EMPID);
END
```

AVG



El esquema es SYSIBM.

La función AVG devuelve el promedio de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos internos) y su suma debe estar dentro del rango del tipo de datos del resultado, excepto para un tipo de datos de resultado decimal. Para los resultados decimales, la suma debe estar dentro del rango soportado por un tipo de datos decimal que tenga una precisión de 31 y una escala idéntica a la escala de los valores del argumento. El resultado puede ser nulo.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento, excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.
- El resultado es DECFLOAT(34) si el argumento es DECFLOAT(*n*).

Si el tipo de datos de los valores del argumento es decimal con la precisión *p* y la escala *s*, la precisión del resultado es 31 y la escala es 31-*p*+*s*.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes. Cuando se interpreta la cláusula DISTINCT para los valores de coma flotante decimal que sean numéricamente iguales, no se tiene en cuenta el número de dígitos significativos del valor. Por ejemplo, el número de coma flotante decimal 123.00 no es diferente del número de coma flotante decimal 123. La representación del número devuelto de la consulta será cualquiera de las representaciones que se encuentre (por ejemplo, 123.00 ó 123).

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor promedio del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

Si el tipo del resultado es entero, se pierde la parte correspondiente a la fracción del promedio.

Ejemplos:

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal AVERAGE (decimal(5,2)) en el nivel promedio de los trabajadores (PRSTAFF) de los proyectos del departamento (DEPTNO) 'D11'.

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

Da como resultado que AVERAGE se establece en 4,25 (es decir 17/4) cuando se utiliza la tabla de ejemplo.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal ANY_CALC (decimal(5,2)) en el promedio de cada valor de nivel exclusivo de los trabajadores (PRSTAFF) de proyectos del departamento (DEPTNO) 'D11'.

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

  El resultado es que ANY_CALC se establece en 4,66 (es decir 14/3) cuando se utiliza la tabla de ejemplo.

CORRELATION

El esquema es SYSIBM.

La función CORRELATION devuelve el coeficiente de correlación de un conjunto de pares de números.

Los valores del argumento deben ser números.

Si el argumento es de coma flotante decimal, el resultado es DECFLOAT(34); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo. Cuando no es nulo, el resultado está entre -1 y 1.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío o si STDDEV(*expresión1*) o STDDEV(*expresión2*) es igual a cero, el resultado es un valor nulo. De lo contrario, el resultado es el coeficiente de correlación para los pares de valores del conjunto. El resultado es equivalente a la expresión siguiente:

```
COVARIANCE(expresión1,expresión2)/
(STDDEV(expresión1)*
STDDEV(expresión2))
```

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

CORR puede especificarse en lugar de CORRELATION.

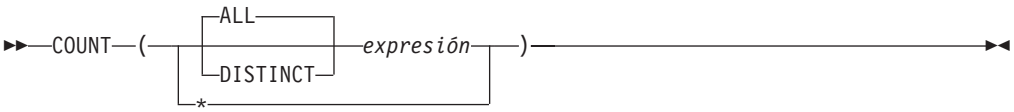
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal CORRLN (coma flotante de precisión doble) en la correlación entre salario y bonificación para los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT CORRELATION(SALARY, BONUS)
    INTO :CORRLN
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  CORRLN se establece en 9,99853953399538E-001 aproximadamente cuando se utiliza la tabla de ejemplo.

COUNT



El esquema es SYSIBM.

La función COUNT devuelve el número de filas o valores de un conjunto de filas o valores.

Si se especifica DISTINCT, el tipo de datos resultante de *expresión* no puede ser un BLOB, CLOB, DBCLOB, XML, un tipo distinto de cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907). En caso contrario, el tipo de datos resultante de la *expresión* puede ser cualquier tipo de datos.

El resultado de la función es un entero grande. El resultado no puede ser nulo.

El argumento de COUNT(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT(*expresión*) o COUNT(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE (int) en el número de filas en que el valor de la columna SEX es 'F'.

  ```
  SELECT COUNT(*)
    INTO :FEMALE
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

  El resultado es que FEMALE se establece en 13 cuando se utiliza la tabla de ejemplo.

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal FEMALE_IN_DEPT (int) en el número de departamentos (WORKDEPT) que tienen como mínimo una mujer como miembro.

  ```
  SELECT COUNT(DISTINCT WORKDEPT)
    INTO :FEMALE_IN_DEPT
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

  El resultado es que FEMALE_IN_DEPT se establece en 5 cuando se utiliza la tabla de ejemplo. (Hay como mínimo una mujer en los departamentos A00, C01, D11, D21 y E11.)

COUNT_BIG

El esquema es SYSIBM.

La función COUNT_BIG devuelve el número de filas o valores de un conjunto de filas o valores. Es similar a COUNT excepto que el resultado puede ser mayor que el valor máximo de entero.

Si se especifica DISTINCT, el tipo de datos resultante de *expresión* no puede ser un BLOB, CLOB, DBCLOB, XML, un tipo distinto de cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907). En caso contrario, el tipo de datos resultante de la *expresión* puede ser cualquier tipo de datos.

El resultado de la función es un decimal con precisión 31 y escala 0. El resultado no puede ser nulo.

El argumento de COUNT_BIG(*) es un conjunto de filas. El resultado es el número de filas del conjunto. Una fila que sólo incluye valores NULL se incluye en la cuenta.

El argumento de COUNT_BIG(DISTINCT *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos y duplicados. El resultado es el número de distintos valores no nulos del conjunto.

El argumento de COUNT_BIG(*expresión*) o COUNT_BIG(ALL *expresión*) es un conjunto de valores. La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluyendo los duplicados.

Ejemplos:

- Consulte los ejemplos de COUNT y sustituya COUNT_BIG por las apariciones de COUNT. Los resultados son los mismos excepto por el tipo de datos del resultado.
- Algunas aplicaciones pueden necesitar la utilización de COUNT pero necesitan dar soporte a valores mayores que el entero más grande. Esto se puede conseguir mediante la utilización de funciones con fuente definidas por el usuario y la definición de la vía de acceso de SQL. Las siguientes series de sentencias muestran cómo crear una función con fuente para dar soporte a COUNT(*) basándose en COUNT_BIG y devolver un valor decimal con una precisión de 15. La vía de acceso de SQL se establece de manera que se utilice la función con fuente basada en COUNT_BIG en las sentencias subsiguientes, tal como la consulta mostrada.

  ```
  CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
    SOURCE SYSIBM.COUNT_BIG();
  SET CURRENT PATH RICK, SYSTEM PATH;
  SELECT COUNT(*) FROM EMPLOYEE;
  ```

  Observe que la función con fuente se define sin parámetros para dar soporte a COUNT(*). Esto sólo es efectivo si utiliza COUNT como nombre de la función y

no califica la función con el nombre de esquema cuando se utiliza. Para conseguir el mismo efecto que COUNT(*) con un nombre distinto de COUNT, invoque la función sin parámetros. Por lo tanto, si RICK.COUNT se ha definido como RICK.MYCOUNT, la consulta se tendría que haber escrito de la siguiente manera:

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

Si la cuenta se efectúa en una columna específica, la función con fuente debe especificar el tipo de columna. Las sentencias siguientes crean una función con fuente que tomará cualquier columna CHAR como argumento y utilizará COUNT_BIG para realizar el recuento.

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE

El esquema es SYSIBM.

La función COVARIANCE devuelve la covarianza (del contenido) de un conjunto de pares de números.

Los valores del argumento deben ser números.

Si el argumento es de coma flotante decimal, el resultado es DECFLOAT(34); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la covarianza de los pares de valores del conjunto. El resultado es equivalente a lo siguiente:

1. Establezca que avgexp1 es el resultado de AVG(*expresión1*) y que avgexp2 es el resultado de AVG(*expresión2*).
2. El resultado de COVARIANCE(*expresión1*, *expresión2*) es AVG((*expresión1* - avgexp1) * (*expresión2* - avgexp2)

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

COVAR puede especificarse en lugar de COVARIANCE.

Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COVARNCE (coma flotante de precisión doble) en la covarianza entre salario y bonificación para los empleados del departamento (WORKDEPT) 'A00'.

```
SELECT COVARIANCE(SALARY, BONUS)
  INTO :COVARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  COVARNCE se establece en 1,68888888888889E+006 aproximadamente cuando se utiliza la tabla de ejemplo.

GROUPING

El esquema es SYSIBM.

Utilizada con conjuntos-agrupaciones y supergrupos, la función GROUPING devuelve un valor que indica si una fila devuelta en un conjunto de respuestas de GROUP BY es una fila generada por un conjunto de agrupaciones que excluye la columna representada por la *expresión* o no.

El argumento puede ser de cualquier tipo, pero debe ser un elemento de una cláusula GROUP BY.

El resultado de la función es un entero pequeño. Se establece en uno de los valores siguientes:

1 El valor de la *expresión* de la fila devuelta es un valor nulo y la fila se ha generado por el supergrupo. Esta fila generada puede utilizarse para proporcionar valores de subtotales para la expresión GROUP BY.

0 El valor no es el de arriba.

Ejemplo:

La siguiente consulta:

```
SELECT SALES_DATE, SALES_PERSON,
    SUM(SALES) AS UNITS_SOLD,
    GROUPING(SALES_DATE) AS DATE_GROUP,
    GROUPING(SALES_PERSON) AS SALES_GROUP
  FROM SALES
  GROUP BY CUBE (SALES_DATE, SALES_PERSON)
  ORDER BY SALES_DATE, SALES_PERSON
```

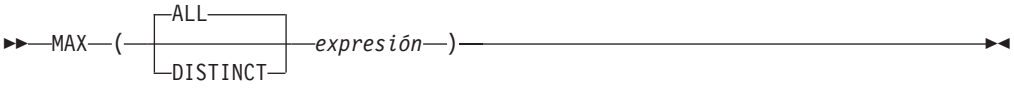
da como resultado:

```
SALES_DATE SALES_PERSON    UNITS_SOLD  DATE_GROUP  SALES_GROUP
---------- --------------- ----------- ----------- -----------
12/31/1995 GOUNOT                    1           0           0
12/31/1995 LEE                       6           0           0
12/31/1995 LUCCHESSI                 1           0           0
12/31/1995 -                         8           0           1
03/29/1996 GOUNOT                   11           0           0
03/29/1996 LEE                      12           0           0
03/29/1996 LUCCHESSI                 4           0           0
03/29/1996 -                        27           0           1
03/30/1996 GOUNOT                   21           0           0
03/30/1996 LEE                      21           0           0
03/30/1996 LUCCHESSI                 4           0           0
03/30/1996 -                        46           0           1
03/31/1996 GOUNOT                    3           0           0
03/31/1996 LEE                      27           0           0
03/31/1996 LUCCHESSI                 1           0           0
03/31/1996 -                        31           0           1
04/01/1996 GOUNOT                   14           0           0
04/01/1996 LEE                      25           0           0
04/01/1996 LUCCHESSI                 4           0           0
04/01/1996 -                        43           0           1
-          GOUNOT                   50           1           0
```

```
-          LEE                    91          1          0
-          LUCCHESSI              14          1          0
-          -                     155          1          1
```

Una aplicación puede reconocer una fila de subtotales de SALES_DATE por el hecho de que el valor de DATE_GROUP es 0 y el valor de SALES_GROUP es 1. Una fila de subtotales SALES_PERSON puede reconocerse por el hecho de que el valor de DATE_GROUP es 1 y el valor de SALES_GROUP es 0. Una fila de total general puede reconocerse por el valor 1 de DATE_GROUP y SALES_GROUP.

MAX



El esquema es SYSIBM.

La función MAX devuelve el valor máximo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo interno que no sea una serie LOB.

El tipo de datos resultante de *expresión* no puede ser un BLOB, CLOB, DBCLOB, un tipo distinto de cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es el valor máximo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal MAX_SALARY (decimal(7,2)) en el valor del salario máximo mensual (SALARY/12).

  ```
  SELECT MAX(SALARY) / 12
    INTO :MAX_SALARY
    FROM EMPLOYEE
  ```

  El resultado es que MAX_SALARY se establece en 4395,83 cuando se utiliza esta tabla de ejemplo.
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal LAST_PROJ(char(24)) en el nombre de proyecto (PROJNAME) que es el último en el orden de clasificación.

  ```
  SELECT MAX(PROJNAME)
    INTO :LAST_PROJ
    FROM PROJECT
  ```

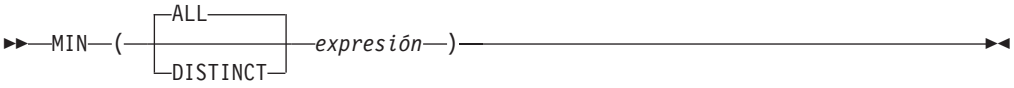
  Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING' cuando se utiliza la tabla de ejemplo.
- De manera parecida al ejemplo anterior, establezca la variable del lenguaje principal LAST_PROJ (char(40)) en el nombre del proyecto que es el último en el

orden de clasificación cuando se concatena un nombre de proyecto con la variable del lenguaje principal PROJSUPP. PROJSUPP es '_Support'; tiene un tipo de datos char(8).

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
  INTO :LAST_PROJ
  FROM PROJECT
```

Da como resultado que LAST_PROJ se establece en 'WELD LINE PLANNING_SUPPORT' cuando se utiliza la tabla de ejemplo.

MIN



La función MIN devuelve el valor mínimo de un conjunto de valores.

Los valores del argumento pueden ser de cualquier tipo interno que no sea una serie LOB.

El tipo de datos resultante de *expresión* no puede ser un BLOB, CLOB, DBCLOB, un tipo distinto de cualquiera de estos tipos ni un tipo estructurado (SQLSTATE 42907).

El tipo de datos, la longitud y la página de códigos del resultado son iguales que el tipo de datos, la longitud y la página de códigos de los valores del argumento. El resultado se considera un valor derivado y puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si la función se aplica a un conjunto vacío, el resultado de la función es un valor nulo. De lo contrario, el resultado es el valor mínimo del conjunto.

La especificación de DISTINCT no tiene ningún efecto en el resultado y, por lo tanto, no es aconsejable. Se incluye para la compatibilidad con otros sistemas relacionados.

Ejemplos:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal COMM_SPREAD (decimal(7,2)) en la diferencia entre la comisión máxima y mínima (COMM) de los miembros del departamento (WORKDEPT) 'D11'.

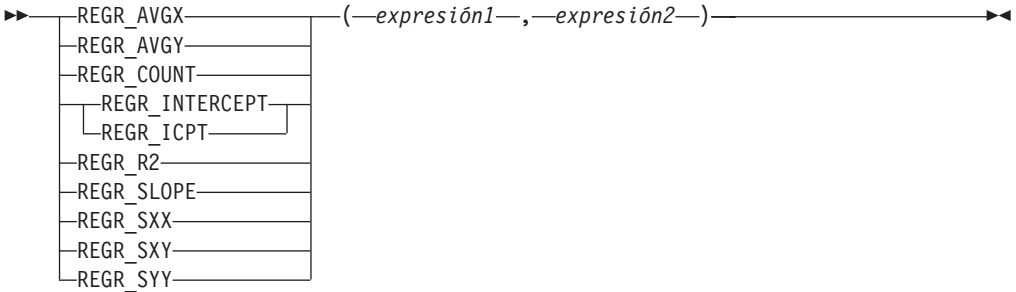
  ```
  SELECT MAX(COMM) - MIN(COMM)
    INTO :COMM_SPREAD
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
  ```

  El resultado es que COMM_SPREAD se establece en 1118 (es decir, 2580 - 1462) cuando se utiliza la tabla de ejemplo.
- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal (FIRST_FINISHED (char(10)) en la fecha de finalización estimada (PRENDATE) del primer proyecto que se ha de terminar.

  ```
  SELECT MIN(PRENDATE)
    INTO :FIRST_FINISHED
    FROM PROJECT
  ```

  Da como resultado que FIRST_FINISHED se establece en '1982-09-15' cuando se utiliza la tabla de ejemplo.

Funciones de regresión



El esquema es SYSIBM.

Las funciones de regresión soportan la adecuación de una línea de regresión mínimo-cuadrados-normales del formato y = a * x + b para un conjunto de pares de números. El primer elemento de cada par (*expresión1*) se interpreta como un valor de la variable dependiente (es decir, un ″valor y″). El segundo elemento de cada par (*expresión2*) se interpreta como un valor de la variable independiente (es decir, un ″valor x″).

La función REGR_COUNT devuelve el número de pares de números no nulos utilizados para acomodar la línea de regresión (vea más abajo).

La función REGR_INTERCEPT (o REGR_ICPT) devuelve la intersección y de la línea de regresión (″b″ en la ecuación anterior).

La función REGR_R2 devuelve el coeficiente de determinación (″cuadrado-R″ o ″mejor-adecuación″) para la regresión.

La función REGR_SLOPE devuelve la inclinación de la línea (″a″ en la ecuación anterior).

Las funciones REGR_AVGX, REGR_AVGY, REGR_SXX, REGR_SXY y REGR_SYY devuelven cantidades que pueden utilizarse para calcular varias estadísticas de diagnóstico necesarias para la evaluación de la calidad y la validez estadística del modelo de regresión (vea más abajo).

Los valores del argumento deben ser números.

El tipo de datos del resultado de REGR_COUNT es un entero. Para las funciones restantes, si uno de los dos argumentos es DECFLOAT(*n*), el tipo de datos del resultado es DECFLOAT(34); en caso contrario, el tipo de datos del resultado es de coma flotante de precisión doble. Si cualquiera de los argumentos es un valor de coma flotante decimal especial, se aplicarán las normas para las operaciones aritméticas generales para la coma flotante decimal. Consulte el apartado “General arithmetic operation rules for decimal floating-point” (Operaciones aritméticas generales para coma flotante decimal) en “Normas generales de operaciones aritméticas para coma flotante decimal” en la página 236.

El resultado puede ser nulo. Cuando no es nulo, el resultado de REGR_R2 está comprendido entre 0 y 1 y el resultado de REGR_SXX y REGR_SYY no es negativo.

Cada función se aplica al conjunto de pares (*expresión1*, *expresión2*) derivado de los valores del argumento por la eliminación de todos los pares para los que *expresión1* o *expresión2* es nulo.

Si el conjunto no está vacío y VARIANCE(*expresión2*) es positivo, REGR_COUNT devuelve el número de pares no nulos del conjunto y las demás funciones devuelven los resultados que se definen de la siguiente manera:

```
REGR_SLOPE(expresión1,expresión2) =
COVARIANCE(expresión1,expresión2)/VARIANCE(expresión2)

REGR_INTERCEPT(expresión1, expresión2) =
AVG(expresión1) - REGR_SLOPE(expresión1,
expresión2) * AVG(expresión2)

REGR_R2(expresión1,
expresión2) =
POWER(CORRELATION(expresión1, expresión2), 2) si
VARIANCE(expresión1)>0
REGR_R2(expresión1, expresión2) = 1 si
VARIANCE(expresion1)=0

REGR_AVGX(expresión1, expresión2) = AVG(expresión2)

REGR_AVGY(expresión1,
expresión2) = AVG(expresión1)

REGR_SXX(expresión1,
expresión2) =
REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión2)

REGR_SYY(expresión1, expresión2) =
REGR_COUNT(expresión1, expresión2) * VARIANCE(expresión1)

REGR_SXY(expresión1, expresión2) =
REGR_COUNT(expresión1, expresión2) * COVARIANCE(expresión1, expresión2)
```

Si el conjunto no está vacío y VARIANCE(*expresión2*) es igual a cero, la línea de regresión tiene una inclinación infinita o no está definida. En este caso, las funciones REGR_SLOPE, REGR_INTERCEPT y REGR_R2 devuelven cada una un valor nulo y las demás funciones devuelven valores tal como se ha definido arriba. Si el conjunto está vacío, REGR_COUNT devuelve cero y las demás funciones devuelven un valor nulo.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

Las funciones de regresión se calculan simultáneamente durante un solo paso a través de los datos. En general, es más eficaz utilizar las funciones de regresión para calcular las estadísticas necesarias para un análisis de regresión que realizar cálculos equivalentes utilizando las funciones normales de columna como AVERAGE, VARIANCE, COVARIANCE, etcétera.

Las estadísticas de diagnóstico normales que acompañan a un análisis de regresión-lineal se pueden calcular en términos de las funciones anteriores. Por ejemplo:

R2 ajustada
1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))

Error estándar
SQRT((REGR_SYY-(POWER(REGR_SXY,2)/REGR_SXX))/
(REGR_COUNT-2))

Suma total de cuadrados
REGR_SYY

Suma de cuadrados de regresión
POWER(REGR_SXY,2) / REGR_SXX

Suma de cuadrados residuales
(Suma total de cuadrados)-(Suma de cuadrados de regresión)

t estadística de inclinación
REGR_SLOPE * SQRT(REGR_SXX) / (Error estándar)

t estadística para intersección y
REGR_INTERCEPT/((Error estándar)*SQRT((1/REGR_COUNT)+
(POWER(REGR_AVGX,2)/REGR_SXX))

Ejemplo:

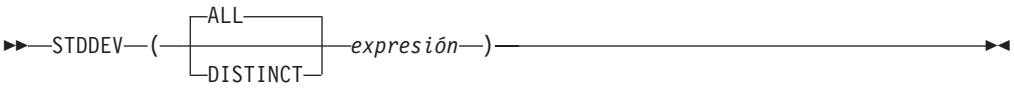
- Utilizando la tabla EMPLOYEE, calcule la línea de regresión de cuadrados-mínimos-normales que expresa la bonificación de un empleado del departamento (WORKDEPT) 'A00' como una función lineal del salario del empleado. Establezca las variables del lenguaje principal SLOPE, ICPT, RSQR (coma flotante de precisión doble) en la inclinación, intersección y coeficiente de determinación de la línea de regresión, respectivamente. Establezca también las variables del lenguaje principal AVGSAL y AVGBONUS en el salario medio y la bonificación media, respectivamente, de los empleados del departamento 'A00', y establezca la variable del lenguaje principal CNT (entero) en el número de empleados del departamento 'A00' para los que están disponibles los datos de salario y de bonificación. Almacene las demás estadísticas de regresión en las variables del lenguaje principal SXX, SYY y SXY.

```
SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
REGR_SXY(BONUS,SALARY)
INTO :SLOPE, :ICPT,
:RSQR, :CNT,
:AVGSAL, :AVGBONUS,
:SXX, :SYY,
:SXY
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
```

Al utilizar la tabla de ejemplo, las variables del lenguaje principal se establecen en los siguientes valores aproximados:

```
SLOPE:  +1.71002671916749E-002
ICPT:  +1.00871888623260E+002
RSQR:  +9.99707928128685E-001
CNT:  3
AVGSAL:  +4.28333333333333E+004
AVGBONUS:  +8.33333333333333E+002
SXX:  +2.96291666666667E+008
SYY:  +8.66666666666667E+004
SXY:  +5.06666666666667E+006
```

STDDEV



El esquema es SYSIBM.

La función STDDEV devuelve la desviación estándar (/*n*) de un conjunto de números. La fórmula que se utiliza para calcular STDDEV es:

```
STDDEV = SQRT(VARIANCE)
```

donde SQRT(VARIANCE) es la raíz cuadrada de la varianza.

Los valores del argumento deben ser números.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes. Cuando se interpreta la cláusula DISTINCT para los valores de coma flotante decimal que sean numéricamente iguales, no se tiene en cuenta el número de dígitos significativos del valor. Por ejemplo, el número de coma flotante decimal 123.00 no es diferente del número de coma flotante decimal 123. La representación del número devuelto de la consulta será cualquiera de las representaciones que se encuentre (por ejemplo, 123.00 ó 123).

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la desviación estándar de los valores del conjunto.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

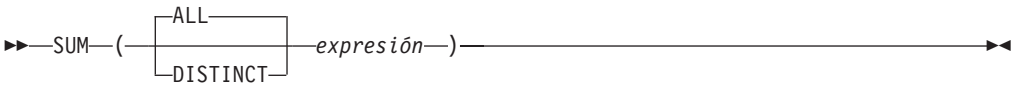
Ejemplo:

- Utilizando la tabla EMPLOYEE, establecer la variable del lenguaje principal en DEV (coma flotante de doble precisión) en la desviación estándar de los sueldos de los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT STDDEV(SALARY)
    INTO :DEV
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  DEV se establece en un número con un valor aproximado de 9938.00.

SUM



El esquema es SYSIBM.

La función SUM devuelve la suma de un conjunto de números.

Los valores del argumento deben ser números (sólo tipos internos) y su suma debe estar dentro del rango del tipo de datos del resultado.

El tipo de datos del resultado es el mismo que el tipo de datos de los valores del argumento, excepto que:

- El resultado es un entero grande si los valores del argumento son enteros pequeños.
- El resultado es de coma flotante de precisión doble si los valores del argumento son de coma flotante de precisión simple.
- El resultado es DECFLOAT(34) si el argumento es DECFLOAT(*n*).

Si el tipo de datos de los valores del argumento es decimal, la precisión del resultado es 31 y la escala es la misma que la de los valores del argumento. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, también se eliminan los valores duplicados redundantes. Cuando se interpreta la cláusula DISTINCT para los valores de coma flotante decimal que sean numéricamente iguales, no se tiene en cuenta el número de dígitos significativos del valor. Por ejemplo, el número de coma flotante decimal 123.00 no es diferente del número de coma flotante decimal 123. La representación del número devuelto de la consulta será cualquiera de las representaciones que se encuentre (por ejemplo, 123.00 ó 123).

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la suma de los valores del conjunto.

El orden en el que los valores se agregan no está definido, pero cada resultado intermedio debe estar dentro del rango del tipo de datos del resultado.

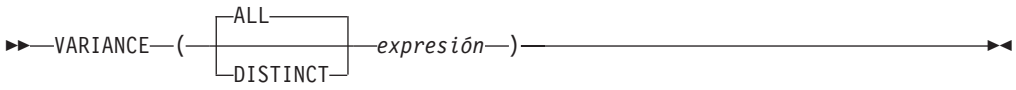
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal JOB_BONUS (decimal(9,2)) en el total de bonificaciones (BONUS) pagadas a los conserjes (JOB='CLERK').

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

  El resultado es que JOB_BONUS se establece en 2800 cuando se utiliza la tabla de ejemplo.

VARIANCE



El esquema es SYSIBM.

La función VARIANCE devuelve la varianza de un conjunto de números.

Los valores del argumento deben ser números.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es de coma flotante de precisión doble. El resultado puede ser nulo.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos. Si se especifica DISTINCT, se eliminan los valores duplicados redundantes. Cuando se interpreta la cláusula DISTINCT para los valores de coma flotante decimal que sean numéricamente iguales, no se tiene en cuenta el número de dígitos significativos del valor. Por ejemplo, el número de coma flotante decimal 123.00 no es diferente del número de coma flotante decimal 123. La representación del número devuelto de la consulta será cualquiera de las representaciones que se encuentre (por ejemplo, 123.00 ó 123).

Si la función se aplica a un conjunto vacío, el resultado es un valor nulo. De lo contrario, el resultado es la varianza de los valores del conjunto.

El orden en el que los valores se añaden es indefinido, pero cada resultado intermedio debe estar en el rango del tipo de datos del resultado.

VAR puede especificarse en lugar de VARIANCE.

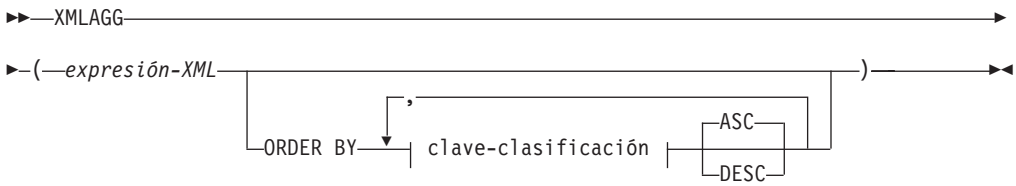
Ejemplo:

- Utilizando la tabla EMPLOYEE, establezca la variable del lenguaje principal VARNCE (coma flotante de precisión doble) en la varianza de los salarios para los empleados del departamento (WORKDEPT) 'A00'.

  ```
  SELECT VARIANCE(SALARY)
    INTO :VARNCE
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  Da como resultado que VARNCE se establece en 98763888,88 aproximadamente cuando se utiliza la tabla de ejemplo.

XMLAGG



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLAGG devuelve una secuencia XML que contiene un elemento por cada valor que no sea nulo en un conjunto de valores XML.

expresión-XML
Especifica una expresión de tipo de datos XML.

ORDER BY
Especifica el orden de las filas del mismo conjunto de agrupación que se procesan en la agregación. Si se omite la cláusula ORDER BY o si ésta no puede distinguir el orden de los datos de la columna, las filas del mismo conjunto de agrupación se ordenan de forma arbitraria.

clave-clasificación
La clave de clasificación puede ser un nombre de columna o una *expresión-clave-clasificación*. Observe que si la clave de clasificación es una constante, no hace referencia a la posición de la columna de salida(como en la cláusula ORDER BY normal) sino que es simplemente una constante, que no implica ninguna clave de clasificación.

El tipo de datos del resultado es XML.

La función se aplica al conjunto de valores derivados de los valores del argumento por la eliminación de los valores nulos.

Si el argumento *expresión-XML* puede ser nulo, el resultado puede ser nulo. Si el conjunto de valores está vacío, el resultado es el valor nulo. En cualquier otro caso, devuelve una secuencia XML que contiene un elemento por cada valor del conjunto.

Nota:

1. **Soporte en expresiones OLAP**: XMLAGG no puede utilizarse como función de columna de una función agregada OLAP (SQLSTATE 42601).

Ejemplo:

Nota: XMLAGG no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Construir un elemento de departamento para cada departamento, con una lista de empleados ordenados por apellido.

```
SELECT XMLSERIALIZE(
  CONTENT XMLELEMENT(
    NAME "Department", XMLATTRIBUTES(
      E.WORKDEPT AS "name"
```

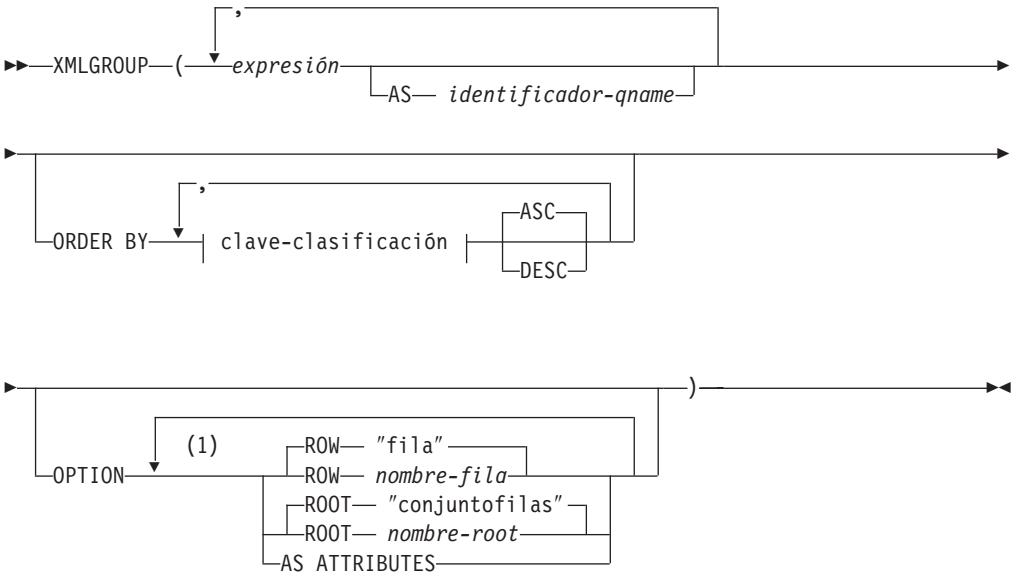
```
      ),
      XMLAGG(
        XMLELEMENT(
          NAME "emp", E.LASTNAME
        )
        ORDER BY E.LASTNAME
      )
    )
    AS CLOB(110)
  )
  AS "dept_list"
  FROM EMPLOYEE E
  WHERE E.WORKDEPT IN ('C01','E21')
  GROUP BY WORKDEPT
```

Esta consulta genera el resultado siguiente:

```
dept_list
----------------------...
<Department name="C01">
  <emp>KWAN</emp>
  <emp>NICHOLLS</emp>
  <emp>QUINTANA</emp>
</Department>
<Department name="E21">
  <emp>GOUNOT</emp>
  <emp>LEE</emp>
  <emp>MEHTA</emp>
  <emp>SPENSER</emp>
</Department>
```

XMLGROUP

La función XMLGROUP devuelve un valor XML con un único nodo de documento XQuery que contiene un nodo de elemento de nivel superior. Esta es una expresión agregada que devolverá un documento XML con una única raíz desde un grupo de filas en el que cada fila está correlacionada a un subelemento de fila.



Notas:

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

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

expresión
El contenido de cada nodo de elemento XML generado (o el valor de cada atributo generado) se especifica mediante una expresión. El tipo de datos *expresión* no puede ser un tipo estructurado (SQLSTATE 42884). La expresión puede ser cualquier expresión SQL. Si la expresión no es una referencia de columna simple, debe especificarse un *identificador-qname*.

AS *identificador-qname*
Especifica el nombre de elemento XML o nombre de atributo como identificador SQL. El *identificador-qname* debe tener el formato de un nombre de calificador XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. Si el nombre está calificado, el prefijo de espacio de nombres deberá declararse dentro del ámbito (SQLSTATE 42635). Si no se especifica *identificador-qname*, *expresión* debe ser un nombre de columna (SQLSTATE 42703). El nombre de elemento o nombre de atributo se crea a partir del nombre de columna, utilizando la correlación con elusión de caracteres ("fully escaped") desde un nombre de columna a un QName.

OPTION
Especifica opciones adicionales para construir el valor XML. Si no se especifica ninguna cláusula OPTION, se aplica el comportamiento por omisión.

ROW *nombre-fila*
Especifica el nombre del elemento al que está correlacionado cada fila. Si no se especifica esta opción, el nombre de elemento por omisión es ″fila″.

ROOT *nombre-root*
Especifica el nombre del nodo de elemento root. Si no se especifica esta opción, el nombre de elemento root por omisión es ″conjuntofilas″.

AS ATTRIBUTES
Especifica que cada expresión está correlacionada a un valor de atributo que tenga nombre de columna o *identificador-qname* que sirva como nombre de atributo.

ORDER BY
Especifica el orden de las filas del mismo conjunto de agrupación que se procesan en la agregación. Si se omite la cláusula ORDER BY o si ésta no puede distinguir el orden de los datos de la columna, las filas del mismo conjunto de agrupación se ordenan de forma arbitraria.

clave-clasificación
La clave de clasificación puede ser un nombre de columna o una *expresión-clave-clasificación*. Observe que si la clave de clasificación es una constante, no hace referencia a la posición de la columna de salida(como en la cláusula ORDER BY normal) sino que es simplemente una constante, que no implica ninguna clave de clasificación.

Notas

El comportamiento por omisión define una correlación simple entre un conjunto de resultados y un valor XML. Se aplican algunas notas adicionales sobre el comportamiento de las funciones:

- Por omisión, cada fila se transforma en un elemento XML denominado ″fila″ y cada columna se transforma en un elemento anidado en el que el nombre de columna sirve como nombre de elemento.
- El comportamiento de manejo de nulos por omisión es NULL ON NULL. Un valor de NULL de una columna se correlaciona con la ausencia del subelemento. Si todos los valores de columna son NULL, no se generará ningún elemento de filas.
- El esquema de codificación binario para los tipos de datos BLOB y FOR BIT DATA es la codificación base64Binary.
- Por omisión, los elementos que se corresponden con las filas de un grupo son hijos de un elemento root denominado ″conjuntofilas″.
- El orden de los subelementos de filas en el elemento root será el mismo que el orden en el que se devuelven las filas en el conjunto de resultados de la consulta.
- Un nodo de documento se añadirá implícitamente al elemento root para hacer que el resultado de XML sea un documento XML con una sola raíz bien formado.

Ejemplos

Suponga que existe la siguiente tabla T1 con las columnas de enteros C1 y C2 que contienen datos numéricos almacenados en un formato relacional.

```
C1          C2
----------- -----------
          1           2
          -           2
```

```
1           -
-           -

4 registro(s) seleccionado(s).
```

- El siguiente ejemplo muestra un fragmento de salida y consulta XMLGroup con comportamiento por omisión, que utiliza un único elemento de nivel superior para representar la tabla:

```
SELECT XMLGROUP(C1, C2)FROM T1

<conjuntofilas>
   <row>
      <C1>1</C1>
      <C2>2</C2>
   </fila>
   <row>
      <C2>2</C2>
   </fila>
   <row>
      <C1>1</C1>
   </fila>
</rowset>

  1 record(s) selected.
```

- El siguiente ejemplo muestra un fragmento de salida y consulta XMLGroup con correlación céntrica de atributos. En vez de aparecer como elementos anidados como en el ejemplo anterior, los datos relacionales se correlacionan a los atributos de elementos:

```
SELECT XMLGROUP(C1, C2 OPTION AS ATTRIBUTES) FROM T1

<conjuntofilas>
   <row C1="1" C2="2"/>
   <row C2="2"/>
   <row C1="1"/>
</rowset>

  1 record(s) selected.
```

- El ejemplo siguiente muestra una consulta XMLGroup y un fragmento de salida con el elemento raíz <rowset> por omisión sustituido por <document> y el elemento <row> por omisión sustituido por <entry>. Las columnas C1 y C2 se devuelven como elementos <column1> y <column2> y el conjunto de retorno se ordena por columna C1:

```
SELECT XMLGROUP(
   C1 AS "column1", C2 AS "column2"
   ORDER BY C1 OPTION ROW "entry" ROOT "document")
FROM T1

<document>
   <entry>
      <column1>1</column1>
      <column2>2</column2>
   </entry>
   <entry>
      <column1>1</column1>
   </entry>
   <entry>
      <column2>2</column2>
    </entry>
</document>
```

Funciones escalares

Una función escalar se puede utilizar siempre que se pueda utilizar una expresión. No obstante, las restricciones que se aplican al uso de expresiones y funciones agregadas, también se aplican cuando una expresión o función agregada se utiliza dentro de una función escalar. Por ejemplo, el argumento de una función escalar puede ser una función agregada sólo si se permite una función agregada en el contexto en el que se utiliza la función escalar.

Las restricciones sobre el uso de funciones agregadas no se aplican a las funciones escalares, ya que una función escalar se aplica a un único valor en vez de aplicarse a un conjunto de valores.

El resultado de la siguiente sentencia SELECT contiene un mismo número de filas igual al número de empleados que hay en el departamento D01:

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

Las funciones escalares puede esta calificadas mediante un nombre de esquema (por ejemplo, SYSIBM.CHAR(123)).

En una base de datos Unicode, todas las funciones escalares que acepten una serie de caracteres o gráfica aceptarán todos los tipos de serie para los que se soporte la conversión.

ABS o ABSVAL

El esquema es SYSIBM.

La versión SYSFUN de la función ABS (o ABSVAL) continúa estando disponible.

Devuelve el valor absoluto del argumento. El argumento puede ser de cualquier tipo de datos numérico interno.

El resultado tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo. Si el argumento es el valor negativo máximo para SMALLINT, INTEGER o BIGINT, el resultado es un error de desbordamiento.

Notas

Resultados que implican valores especiales de DECFLOAT: para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:

- ABS(NaN) y ABS(-NaN) devuelven NaN.
- ABS(Infinity) y ABS(-Infinity) devuelven Infinity.
- ABS(sNaN) y ABS(-sNaN) devuelven sNaN.

Ejemplo

```
ABS(-51234)
```

devuelve un INTEGER con un valor de 51234.

ACOS

El esquema es SYSIBM. (La versión SYSFUN de la función ACOS continúa estando disponible).

Devuelve el arcocoseno del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

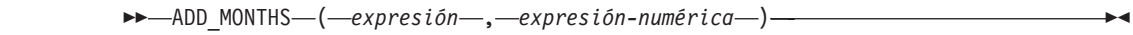
Ejemplo:

Supongamos que la variable del lenguaje principal ACOSINE es una variable del lenguaje principal DECIMAL(10,9) con un valor de 0,070737202.

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

Esta sentencia devuelve el valor aproximado 1,49.

ADD_MONTHS



El esquema es SYSIBM.

La función ADD_MONTHS devuelve un valor de fecha y hora que representa la *expresión* más un número de meses especificado.

expresión
Expresión que especifica la fecha inicial. La expresión debe devolver un valor de uno de los siguientes tipos de datos incorporados: valor DATE o valor TIMESTAMP.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. La *expresión-numérica* indica el número de meses que se añadirán a la fecha inicial especificada en *expresión*. Está permitido el uso de valores numéricos negativos.

El resultado de la función tiene el mismo tipo de datos que *expresión*, a menos que *expresión* sea una serie, en cuyo caso el tipo de datos del resultado es DATE. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Si *expresión* es el último día del mes o si el mes resultante tiene menos días que el componente de día de *expresión*, el resultado es el último día del mes resultante. En cualquier otro caso, el resultado tiene el mismo componente de día que *expresión*. La función no modifica la información de horas, minutos, segundos o segundos fraccionados incluida en *expresión*.

Ejemplos

- Supongamos que hoy es 31 de enero de 2007. Establecer la variable del lenguaje principal ADD_MONTH con el último día de enero más un mes.

  ```
  SET :ADD_MONTH = ADD_MONTHS(LAST_DAY(CURRENT_DATE), 1);
  ```

  La variable del lenguaje principal ADD_MONTH se establece con el valor que representa el final de febrero, 2007-02-28.
- Supongamos que DATE es un variable del lenguaje principal con el valor 27 de julio de 1965. Establecer la variable del lenguaje principal ADD_MONTH con el valor de ese día más tres meses.

  ```
  SET :ADD_MONTH = ADD_MONTHS(:DATE,3);
  ```

  La variable del lenguaje principal ADD_MONTH se establece con el valor que representa ese día más tres meses, 1965-10-27.
- Es posible conseguir resultados similares con la función ADD_MONTHS y la aritmética de fecha. En los ejemplos siguientes se muestran las similitudes y las diferencias.

  ```
  SET :DATEHV = DATE('2008-2-28') + 4 MONTHS;
  SET :DATEHV = ADD_MONTHS('2008-2-28', 4);
  ```

  En ambos casos, la variable del lenguaje principal DATEHV se establece con el valor '2008–06–28'.

  Ahora consideremos el mismo ejemplo pero con la fecha '2008–2–29' como argumento.

```
SET :DATEHV = DATE('2008-2-29') + 4 MONTHS;
```

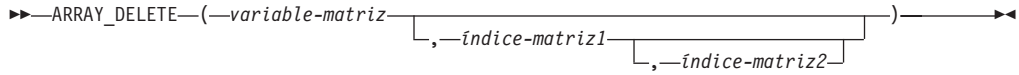
La variable del lenguaje principal DATEHV se establece con el valor '2008–06–29'.

```
SET :DATEHV = ADD_MONTHS('2008-2-29', 4);
```

La variable del lenguaje principal DATEHV se establece con el valor '2008–06–30'.

En este caso, la función ADD_MONTHS devuelve el último día del mes, que es el 30 de junio de 2008, en lugar del 29 de junio de 2008. El motivo es que el 29 de febrero es el último día del mes. Por lo tanto, la función ADD_MONTHS devuelve el último día de junio.

ARRAY_DELETE



El esquema es SYSIBM.

La función ARRAY_DELETE elimina elementos de una matriz.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

índice-matriz1
Expresión que da lugar a un valor que se puede asignar al tipo de datos del índice de matriz. Si la *variable-matriz* es una matriz común, el *índice-matriz1* debe tener un valor nulo.

índice-matriz2
Expresión que da lugar a un valor que se puede asignar al tipo de datos del índice de matriz. Si la *variable-matriz* es una matriz común, el *índice-matriz2* debe tener un valor nulo. Si se especifica el *índice-matriz2* y no es un valor nulo, el *índice-matriz1* debe tener un valor que no sea nulo y que sea menor que el valor del *índice-matriz2* (SQLSTATE 42815).

El resultado de la función tiene el mismo tipo de datos que la *variable-matriz*. Si no se especifican los argumentos opcionales o su valor es nulo, se eliminan todos los elementos de la *variable-matriz* y la cardinalidad del valor de la matriz resultante es 0. Si solo se especifica el *índice-matriz1* con un valor no nulo, se elimina el valor del elemento de matriz en el índice *índice-matriz1*. Si también se especifica el *índice-matriz2* con un valor no nulo, se eliminan los elementos comprendidos entre el valor de índice *índice-matriz1* y el valor de índice *índice-matriz2*, ambos incluidos.

El resultado puede ser nulo; si la *variable-matriz* es nula, el resultado es el valor nulo.

Notas

- La función ARRAY_DELETE puede utilizarse únicamente a la derecha de una sentencia de asignación en contextos en los que se da soporte a las matrices.

Ejemplos

- Eliminar todos los elementos de la variable de matriz común RECENT_CALLS de tipo de matriz PHONENUMBERS.

```
SETRECENT_CALLS = ARRAY_DELETE(RECENT_CALLS)
```

- Un proveedor ha dejado de ofrecer algunos de sus productos. Eliminar los elementos de la variable de matriz asociativaFLOOR_TILES de tipo de matriz PRODUCTS comprendidos entre el valor de índice 'PK5100' y el valor de índice 'PS2500'.

```
SETFLOOR_TILES = ARRAY_DELETE(FLOOR_TILES,'PK5100','PS2500')
```

ARRAY_FIRST

El esquema es SYSIBM.

La función ARRAY_FIRST devuelve el valor de índice mínimo de la matriz.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

El tipo de datos del resultado es el tipo de datos del índice de matriz, que es INTEGER en el caso de las matrices comunes. Si *variable-matriz* no tiene el valor nulo y la cardinalidad de la matriz es mayor que cero, el valor del resultado es el valor de índice de matriz mínimo, que es 1 en el caso de una matriz común.

El resultado puede ser nulo; si *variable-matriz* tiene el valor nulo o la cardinalidad de la matriz es cero, el resultado es el valor nulo.

Ejemplos

- Devolver el primer valor de índice de la variable de matriz común SPECIALNUMBERS a la variable de SQL E_CONSTIDX.

  ```
  SET E_CONSTIDX = ARRAY_FIRST(SPECIALNUMBERS)
  ```

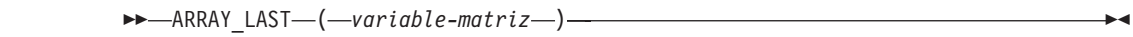
  El resultado es 1.
- Dada la variable de matriz asociativa PHONELIST con estos valores de índice y números de teléfono: 'Home' es '4163053745', 'Work' es '4163053746' y 'Mom' es '416-4789683', asignar el valor del índice mínimo de la matriz a la variable de serie de caracteres denominada X.

  ```
  SET X = ARRAY_FIRST(PHONELIST)
  ```

  El valor de 'Particular' se asigna a X. Acceder al valor del elemento asociado con el valor de índice 'Particular' y asignarlo a la variable de SQL NUMBER_TO_CALL:

  ```
  SET NUMBER_TO_CALL = PHONELIST[X]
  ```

ARRAY_LAST



El esquema es SYSIBM.

La función ARRAY_LAST devuelve el valor de índice máximo de la matriz.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

El tipo de datos del resultado es el tipo de datos del índice de matriz, que es INTEGER en el caso de las matrices comunes. Si *variable-matriz* no tiene el valor nulo y la cardinalidad de la matriz es mayor que cero, el valor del resultado es el valor de índice de matriz máximo, que es la cardinalidad de la matriz en el caso de una matriz común.

El resultado puede ser nulo; si variable-matriz tiene el valor nulo o la cardinalidad de la matriz es cero, el resultado es el valor nulo.

Ejemplos

- Devolver el último valor de índice de la variable de matriz común SPECIALNUMBERS a la variable de SQL PI_CONSTIDX.

  ```
  SET PI_CONSTIDX = ARRAY_LAST(SPECIALNUMBERS)
  ```

  El resultado es 10.

- Dada la variable de matriz asociativa PHONELIST con estos valores de índice y números de teléfono: 'Home' es '4163053745', 'Work' es '4163053746' y 'Mom' es '4164789683', asignar el valor del índice máximo de la matriz a la variable de tipo serie de caracteres denominada X.

  ```
  SET X = ARRAY_LAST(PHONELIST)
  ```

  El valor de 'Work' se asigna a X. Acceder al valor del elemento asociado con el valor de índice 'Work' y asignarlo a la variable de SQL NUMBER_TO_CALL:

  ```
  SET NUMBER_TO_CALL = PHONELIST[X]
  ```

ARRAY_NEXT

El esquema es SYSIBM.

La función ARRAY_NEXT devuelve el siguiente valor de índice de matriz más alto para una matriz relativa al argumento de índice de la matriz especificada.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

índice-matriz
Especifica un valor que se puede asignar al tipo de datos del índice de la matriz. Los valores válidos son cualquier valor válido para el tipo de datos.

El resultado es el siguiente valor de índice de matriz más alto definido en la matriz relativa al valor de *índice-matriz* especificado. Si *índice-matriz* es inferior al valor de índice de matriz mínimo en la matriz, el resultado es el primer valor de índice de matriz definido en la matriz.

El tipo de datos del resultado de la función es el tipo de datos del índice de matriz. El resultado puede ser nulo; si algún argumento es nulo, la cardinalidad del primer argumento es cero o el valor de *índice-matriz* es mayor o igual que el valor del último índice de la matriz, el resultado es el valor nulo.

Ejemplos

- Devolver el siguiente valor de índice después de la 9ª posición de índice en la variable de matriz común SPECIALNUMBERS a la variable de SQL NEXT_CONSTIDX.

  ```
  SET NEXT_CONSTIDX = ARRAY_NEXT(SPECIALNUMBERS,9)
  ```

  El resultado es 10.
- Dada la variable de matriz asociativa PHONELIST con valores de índice y números de teléfono: 'Casa' es '4163053745', 'Trabajo' es '4163053746' y 'Mamá' es '416-4789683', asigne el valor del índice en la matriz que es el siguiente índice después del valor de índice 'Papá', que no existe para el valor de matriz, a la variable de serie de caracteres llamada X:

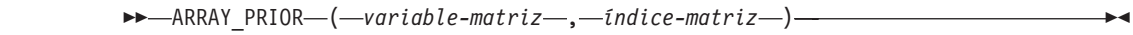
  ```
  SET X = ARRAY_NEXT(PHONELIST, 'Papá')
  ```

  El valor de 'Casa' se asigna a X, puesto que el valor 'Papá' es un valor inferior a cualquier otro valor de índice para la variable de matriz. Asigne el valor del índice en la matriz que es el índice siguiente al valor de índice'Trabajo':

  ```
  SET X = ARRAY_NEXT(PHONELIST, 'Trabajo')
  ```

  El valor nulo se asigna a X.

ARRAY_PRIOR



El esquema es SYSIBM.

La función ARRAY_PRIOR devuelve el valor de índice de matriz más bajo siguiente para una matriz relativa al argumento de índice de la matriz especificada.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

índice-matriz
Especifica un valor que se puede asignar al tipo de datos del índice de la matriz. Los valores válidos son cualquier valor válido para el tipo de datos.

El resultado es el siguiente valor de índice de matriz más bajo definido en la matriz relativa al valor de *índice-matriz* especificado. Si *índice-matriz* es superior al valor de índice de matriz máximo en la matriz, el resultado es el último valor de índice de matriz definido en la matriz.

El tipo de datos del resultado de la función es el tipo de datos del índice de matriz. El resultado puede ser nulo; si algún argumento es nulo, la cardinalidad del primer argumento es cero o el valor de *índice-matriz* es inferior o igual que el valor del primer índice de la matriz, el resultado es el valor nulo.

Ejemplos

- Devolver el previo valor de índice antes de la 2ª posición de índice en la variable de matriz común SPECIALNUMBERS a la variable de SQL PREV_CONSTIDX.

  ```
  SET PREV_CONSTIDX = ARRAY_PRIOR(SPECIALNUMBERS,2)
  ```

  El resultado es 1.
- Dada la variable de matriz asociativa PHONELIST con valores de índice y números de teléfono: 'Casa' es '4163053745', 'Trabajo' es '4163053746' y 'Mamá' es '416-4789683', asigne el valor del índice en la matriz que es el previo índice antes del valor de índice 'Trabajo' a la variable de serie de caracteres llamada X:

  ```
  SET X = ARRAY_PRIOR(PHONELIST, 'Trabajo')
  ```

  El valor de 'Mamá' se asigna a X. Asigne el valor del índice en la matriz que es el previo índice antes del valor de índice'Casa':

  ```
  SET X = ARRAY_PRIOR(PHONELIST, 'Casa')
  ```

  El valor null se asigna a X.

ASCII

El esquema es SYSFUN.

Devuelve el valor en código ASCII del carácter que hay más a la izquierda del argumento como un entero.

El argumento puede ser de cualquier tipo de serie de caracteres interno. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. Para un VARCHAR, la longitud máxima es de 4.000 bytes y para un CLOB, la longitud máxima es de 1.048.576 bytes. LONG VARCHAR se convierte a CLOB para que lo procese la función.

El resultado de la función siempre es INTEGER.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

ASIN

El esquema es SYSIBM. (La versión SYSFUN de la función ASIN continúa estando disponible).

Devuelve el arcoseno del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATAN

El esquema es SYSIBM. (La versión SYSFUN de la función ATAN continúa estando disponible).

Devuelve la tangente del arco del argumento como un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATAN2

El esquema es SYSIBM. (La versión SYSFUN de la función ATAN2 continúa estando disponible).

Devuelve la tangente del arco de las coordenadas x e y como un ángulo expresado en radianes. Las coordenadas x e y se especifican por el primer y el segundo argumento, respectivamente.

El primer argumento y el segundo pueden ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Los dos se convierten a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

ATANH

El esquema es SYSIBM.

Devuelve la arcotangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

BIGINT

De numérico a entero superior

De serie a entero superior

De fecha y hora a entero superior

El esquema es SYSIBM.

La función BIGINT devuelve una representación de enteros de 64 bits de:

- Un número
- Una representación de serie de un número
- Un valor de fecha y hora

De numérico a entero superior

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado es el mismo número que el que se generaría si se asignara el argumento a una variable o columna de enteros superiores. La parte fraccionaria del argumento se trunca. Si la parte completa del argumento no está dentro del rango de los enteros superiores, se devuelve un error (SQLSTATE 22003).

De serie a entero superior

expresión-serie
Una expresión que devuelve un valor de serie de caracteres o la representación de serie gráfica Unicode de un número con una longitud no superior a la longitud máxima de una constante de caracteres.

El resultado es el mismo número que el que generaría CAST(*expresión-serie* AS BIGINT). Los espacios en blanco iniciales y finales se eliminan y la serie resultante debe ajustarse a las normas para formar una constante de coma flotante decimal, de entero, de decimal o de coma flotante (SQLSTATE 22018). Si la parte completa del argumento no está dentro del rango de los enteros superiores, se devuelve un error (SQLSTATE 22003). El tipo de datos de *expresión-serie* no debe ser CLOB ni DBCLOB (SQLSTATE 42884).

De fecha y hora a entero superior

expresión-fecha-hora
Una expresión que sea uno de los tipos de datos siguientes:

- DATE. El resultado es un valor BIGINT que representa la fecha como *aaaammdd*.

- TIME. El resultado es un valor BIGINT que representa la hora como *hhmmss*.
- TIMESTAMP. El resultado es un valor BIGINT que representa la indicación de fecha y hora como *aaaammddhhmmss*. La parte de segundos fraccionarios del valor de la indicación de fecha y hora no se incluye en el resultado.

El resultado de la función es un entero superior. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".

Ejemplos

- En la tabla ORDERS_HISTORY, cuente el número de órdenes y devuelva el resultado como un valor de entero superior.

```
SELECT BIGINT (COUNT_BIG(*))
  FROM ORDERS_HISTORY
```

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros superiores para procesarla más en la aplicación.

```
SELECT BIGINT (EMPNO) FROM EMPLOYEE
```

- Supongamos que la columna RECEIVED (cuyo tipo de datos es TIMESTAMP) tiene un valor interno equivalente a '1988-12-22-14.07.21.136421'.

```
BIGINT(RECEIVED)
```

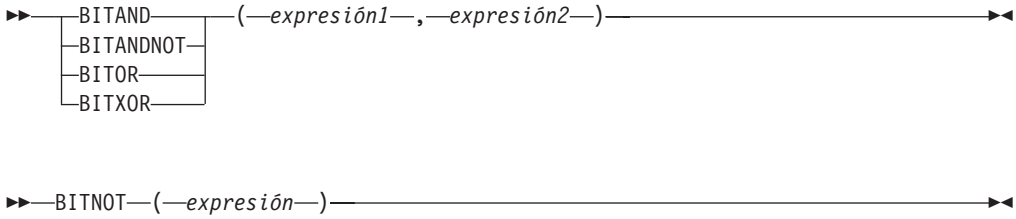
  da como resultado el valor `19 881 222 140 721`.

- Supongamos que la columna STARTTIME (cuyo tipo de datos es TIME) tiene un valor interno equivalente a '12:03:04'.

```
BIGINT(STARTTIME)
```

  da como resultado el valor `120 304`.

BITAND, BITANDNOT, BITOR, BITXOR y BITNOT



El esquema es SYSIBM.

Estas funciones a nivel de bit operan en la representación de ″complemento del dos″ del valor entero de los argumentos de entrada y devuelven el resultado como un correspondiente valor entero de base 10 en un tipo de datos basándose en el tipo de datos de los argumentos de entrada.

Tabla 43. Funciones de manipulación de bits

Función	Descripción	Un bit en la representación de complemento del dos del resultado es:
BITAND	Realiza una operación AND a nivel de bit.	1 sólo si los bits correspondientes en ambos argumentos son 1.
BITANDNOT	Borra cualquier bit del primer argumento que esté en el segundo argumento.	Cero si el bit correspondiente del segundo argumento es 1; de lo contrario, el resultado se copia del bit correspondiente del primer argumento.
BITOR	Realiza una operación OR a nivel de bit.	1 a menos que los bits correspondientes de ambos argumentos sean cero.
BITXOR	Realiza una operación OR exclusiva a nivel de bit.	1 a menos que los bits correspondientes de ambos argumentos sean iguales.
BITNOT	Realiza una operación NOT a nivel de bit.	Contrario del bit correspondiente del argumento.

Los argumentos deben ser valores enteros representados por los tipos de datos SMALLINT, INTEGER, BIGINT o DECFLOAT. Los argumentos de tipo DECIMAL, REAL o DOUBLE se convierten en DECFLOAT. El valor se trunca a un número entero.

Las funciones de manipulación de bit pueden operar en un máximo de 16 bits para SMALLINT, 32 bits para INTEGER, 64 bits para BIGINT y 113 bits para DECFLOAT. El rango de valores DECFLOAT soportado incluye enteros de -2^{112} a 2^{112} -1 y los valores especiales tales como NaN o INFINITY no se soportan (SQLSTATE 42815). Si los dos argumentos tienen tipos de datos diferentes, el argumento que soporta menos bits se convierte en un valor con el tipo de datos del argumento que soporta más bits. Esta conversión afecta a los bits que están

establecidos para valores negativos. Por ejemplo, -1 como un valor SMALLINT tiene 16 bits establecidos en 1, que, cuando se transforma en un valor INTEGER, tiene 32 bits establecidos en 1.

El resultado de las funciones con dos argumentos tiene el tipo de datos del argumento que está más arriba en la lista de prioridades de tipo de datos para promoción. Si alguno de los argumentos es DECFLOAT, el tipo de datos del resultado es DECFLOAT(34). Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

El resultado de la función BITNOT tiene el mismo tipo de datos que el argumento de entrada, excepto en que DECIMAL, REAL, DOUBLE o DECFLOAT(16) devuelve DECFLOAT(34). Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Debido a las diferencias en la representación interna entre los tipos de datos y en las diferentes plataformas de hardware, la utilización de funciones (tales como HEX) o construcciones en lenguaje de sistema principal para ver o comparar representaciones internas de los resultados y argumentos de la función BIT depende del tipo de datos y no es portable. El procedimiento independiente de la plataforma y del tipo de datos para ver o comparar los resultados y argumentos de la función BIT consiste en utilizar los valores enteros reales.

Se recomienda utilizar la función BITXOR para conmutar bits en un valor. Utilice la función BITANDNOT para borrar bits. BITANDNOT(val, pattern) funciona de forma más eficiente que BITAND(val, BITNOT(pattern)).

Ejemplos:

Los ejemplos siguientes se basan en una tabla ITEM con una columna PROPERTIES de tipo INTEGER.

- Devolver todos los elementos para los que se ha establecido el tercer bit de propiedad.

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 4) = 4
```

- Devolver todos los elementos para los que se ha establecido el cuarto y sexto bit de propiedad.

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 40) <> 0
```

- Borrar la duodécima propiedad del elemento cuyo ID es 3412.

```
UPDATE ITEM
    SET PROPERTIES = BITANDNOT(PROPERTIES, 2048)
  WHERE ITEMID = 3412
```

- Establecer la quinta propiedad del elemento cuyo ID es 3412.

```
UPDATE ITEM
    SET PROPERTIES = BITOR(PROPERTIES, 16)
  WHERE ITEMID = 3412
```

- Conmutar la undécima propiedad del elemento cuyo ID es 3412.

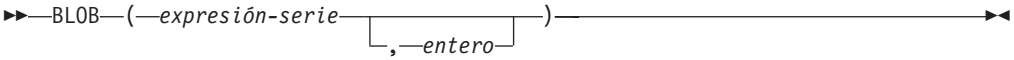
```
UPDATE ITEM
    SET PROPERTIES = BITXOR(PROPERTIES, 1024)
  WHERE ITEMID = 3412
```

- Conmutar todos los bits de un valor de 16 bits que sólo tiene activo el segundo bit.

```
VALUES BITNOT(CAST(2 AS SMALLINT))
```

devuelve -3 (con un tipo de datos de SMALLINT).

BLOB



El esquema es SYSIBM.

La función BLOB devuelve una representación BLOB de una serie de cualquier tipo.

expresión-serie
Una *expresión-serie* cuyo valor puede ser una serie de caracteres, una serie gráfica o una serie binaria.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos BLOB resultante. Si no se especifica *entero*, el atributo de longitud del resultado es el mismo que la longitud de la entrada, excepto cuando la entrada es gráfica. En este caso, el atributo de longitud del resultado es el doble de la longitud de la entrada.

El resultado de la función es un BLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Suponiendo una tabla con una columna BLOB denominada TOPOGRAPHIC_MAP y una columna VARCHAR denominada MAP_NAME, localice los mapas que contienen la serie 'Pellow Island' y devuelva una sola serie binaria con el nombre del mapa concatenado delante del mapa real.

```
SELECT BLOB(MAP_NAME  CONCAT ':  ') CONCAT TOPOGRAPHIC_MAP
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE BLOB('%Pellow Island%')
```

CARDINALITY

El esquema es SYSIBM.

La función CARDINALITY devuelve un valor del tipo BIGINT que representa el número de elementos de una matriz.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

Para una matriz común el valor devuelto por la función CARDINALITY es el subíndice más alto para el que la matriz tiene un elemento asignado. Esto incluye elementos a los que se les ha asignado el valor nulo.Para una matriz asociativa, el valor devuelto por la función CARDINALITY es el número real de valores exclusivos de índice de matriz que se han definido en la *variable-matriz*.

La función devuelve 0 si la matriz está vacía. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Devolver el número de llamadas que se ha almacenado en la lista de llamadas recientes hasta ahora:

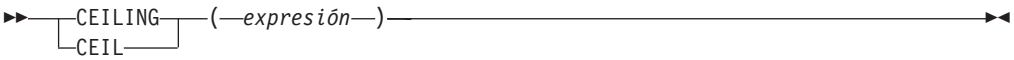
  ```
  SET HOWMANYCALLS = CARDINALITY(RECENT_CALLS)
  ```

  La variable de SQL `HOWMANYCALLS` contiene el valor 3.
- Suponga que la variable de matriz asociativa CAPITALS de tipo de matriz CAPITALSARRAY contiene todas las capitales de las 10 provincias y 3 territorios en Canadá, así como la capital del país, Ottawa. Para devolver la cardinalidad de la variable de matriz:

  ```
  SET NUMCAPITALS = CARDINALITY(CAPITALS)
  ```

  La variable de SQL `NUMCAPITALS` contiene el valor 14.

CEILING o CEIL



El esquema es SYSIBM. (La versión SYSFUN de la función CEILING continúa estando disponible).

Devuelve el valor del entero más pequeño que es mayor o igual que el argumento.

El argumento puede ser de cualquier tipo numérico interno. El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento, con la excepción de que la escala es 0 si el argumento es DECIMAL. Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,5) devuelve DECIMAL(5,0).

El resultado puede ser nulo si el argumento puede ser nulo o si el argumento no es un número de coma flotante decimal y la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

Notas

Resultados que implican valores especiales de DECFLOAT: para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:

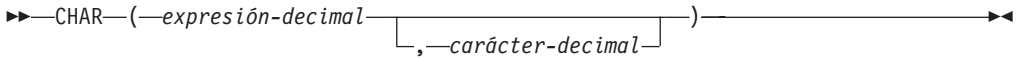
- CEILING(NaN) devuelve NaN.
- CEILING(-NaN) devuelve -NaN.
- CEILING(Infinity) devuelve Infinity.
- CEILING(-Infinity) devuelve -Infinity.
- CEILING(sNaN) devuelve NaN y un aviso.
- CEILING(-sNaN) devuelve -NaN y un aviso.

Sintaxis alternativa: puede especificarse CEIL en lugar de CEILING.

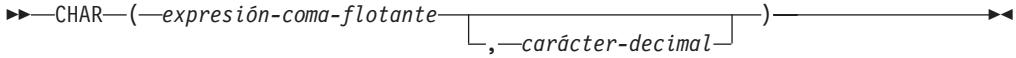
CHAR

De entero a carácter

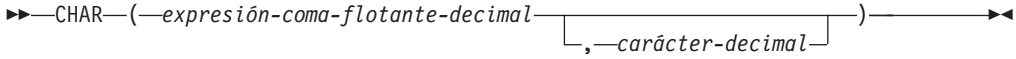
De decimal a carácter



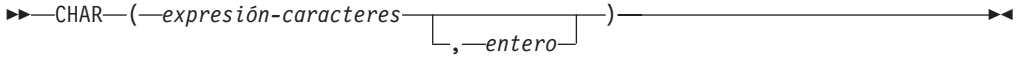
De coma flotante a carácter



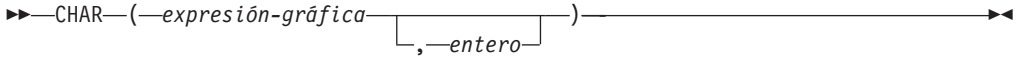
De coma flotante decimal a carácter



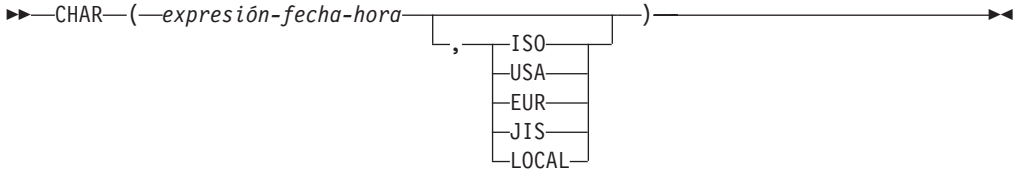
De carácter a carácter



De gráfico a carácter



De fecha y hora a carácter



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función. La signatura SYSFUN.CHAR(*expresión-coma-flotante*) continúa estando disponible. En este caso, el carácter decimal es sensible la configuración local y, por lo tanto, devuelve un punto o una coma, dependiendo del idioma del servidor de la base de datos.

La función CHAR devuelve una representación de serie de caracteres de longitud fija de:

- Un número entero, si el primer argumento es SMALLINT, INTEGER o BIGINT
- Un número decimal, si el primer argumento es un número decimal
- Un número de coma flotante de precisión doble, si el primer argumento es DOUBLE o REAL
- Un número de coma flotante decimal, si el primer argumento es un DECFLOAT
- Una serie de caracteres, si el primer argumento es cualquier tipo de serie de caracteres
- Una serie gráfica (sólo para bases de datos Unicode), si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora, si el primer argumento es DATE, TIME o TIMESTAMP

En una base de datos Unicode, cuando en la serie de salida un carácter de varios bytes aparece truncado en parte:

- Si la salida era una serie de caracteres, el carácter parcial se sustituye por uno o varios blancos
- Si la salida era una serie gráfica, el carácter parcial se sustituye por una serie vacía

No confíe en ninguno de estos comportamientos, porque podrían cambiar en los releases futuros.

El resultado de la función es una serie de caracteres de longitud fija. Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

De entero a carácter

expresión-entero
Una expresión que devuelve un valor que es un tipo de datos de entero (SMALLINT, INTEGER o BIGINT).

El resultado es una representación de serie de caracteres de longitud fija de *expresión-entero* en forma de constante de entero SQL. El resultado consta de *n* caracteres, que representan los dígitos significativos del argumento, precedido por un signo menos si el argumento es negativo. El resultado está justificado por la izquierda.

- Si el primer argumento es un entero pequeño, la longitud del resultado es de 6.
- Si el primer argumento es un entero grande, la longitud del resultado es de 11.
- Si el primer argumento es un entero superior, la longitud del resultado es de 20.

Si el número de bytes en el resultado es menor que la longitud definida del resultado, éste se rellena por la derecha con espacios en blanco de un solo byte.

La página de códigos del resultado es la página de códigos de la sección.

De decimal a carácter

expresión-decimal
Una expresión que devuelve un valor que es de un tipo de datos decimal. Si se necesita una precisión y escala diferentes, puede utilizarse primero la función escalar DECIMAL para realizar el cambio.

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud fija de la *expresión-decimal* con el formato de una constante decimal SQL. La longitud del resultado es 2+*p*, donde *p* es la precisión de la *expresión-decimal*. Los ceros iniciales no se incluyen. Los ceros finales se incluyen. Si *expresión-decimal* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito o un carácter decimal. Si la escala de *expresión-decimal* es cero, el carácter decimal no se devuelve. Si el número de bytes en el resultado es menor que la longitud definida del resultado, éste se rellena por la derecha con espacios en blanco de un solo byte.

La página de códigos del resultado es la página de códigos de la sección.

De coma flotante a carácter

expresión-coma-flotante
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud fija de la *expresión-coma-flotante* con el formato de una constante de coma flotante SQL. La longitud del resultado es 24. El resultado es el número menor de caracteres que puedan representar el valor de la *expresión-coma-flotante*, de manera que la mantisa conste de un solo dígito que no sea cero seguido de un punto y una secuencia de dígitos. Si *expresión-coma-flotante* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si *expresión-coma-flotante* es cero, el resultado es 0E0. Si el número de bytes en el resultado es menor que 24, el resultado se rellena por la derecha con espacios en blanco de un solo byte.

La página de códigos del resultado es la página de códigos de la sección.

De coma flotante decimal a carácter

expresión-coma-flotante-decimal
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante decimal (DECFLOAT).

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud fija de la *expresión-coma-flotante-decimal* con el formato de una constante de coma flotante decimal SQL. El atributo de longitud del resultado es 42. El resultado es el número menor de caracteres que puedan representar el valor de *expresión-coma-flotante-decimal*. Si *expresión-coma-flotante-decimal* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si *expresión-coma-flotante-decimal* es cero, el resultado es 0.

Si el valor de *expresión-coma-flotante-decimal* es el valor especial Infinity, sNaN o NaN, se devuelven las series 'INFINITY', 'SNAN' y 'NAN', respectivamente. Si el valor especial es negativo, el primer carácter del resultado es un signo menos. El valor especial de coma flotante decimal sNaN no genera un aviso cuando se convierte en una serie. Si el número de caracteres en el resultado es menor que 42, éste se rellena por la derecha con espacios en blanco de un solo byte.

La página de códigos del resultado es la página de códigos de la sección.

De carácter a carácter

expresión-caracteres
Una expresión que devuelve un valor que es una serie de caracteres incorporados (CHAR, VARCHAR o CLOB).

entero
El atributo de longitud de la serie de caracteres de longitud fija resultante. El valor debe estar entre 0 y 254.

Si el segundo argumento no se especifica:

- Si la *expresión-caracteres* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento. Si la longitud real del primer argumento (sin contar los espacios en blanco finales) es superior a 254, se devuelve un error (SQLSTATE 22001).

La longitud real del resultado es la misma que el atributo de longitud del resultado. Si la longitud de la *expresión-caracteres* es menor que el atributo de longitud del resultado, el resultado se rellena con espacios en blanco hasta llenar toda la longitud del resultado. Si la longitud de la *expresión-caracteres* es superior al atributo de longitud del resultado, éste se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y la *expresión-caracteres* no sea CLOB.

Si la longitud de la expresión de caracteres es menor que el atributo de longitud del resultado, el resultado se rellena con blancos hasta la longitud del resultado. Si la longitud de la expresión de caracteres es mayor que el atributo de longitud del resultado, el resultado se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y la expresión gráfica no sea CLOB.

De gráfico a carácter

expresión-gráfica
Expresión que devuelve un valor que es un tipo de datos de serie gráfica incorporado. (GRAPHIC, VARGRAPHIC o DBCLOB).

entero
El atributo de longitud de la serie de caracteres de longitud fija resultante. El valor debe estar entre 0 y 254.

Si el segundo argumento no se especifica:

- Si la *expresión-gráfica* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que MIN (254, 3* atributo de longitud del primer argumento). Si la longitud real del primer argumento (incluidos los espacios en blanco finales) es superior a 254, se devuelve un error (SQLSTATE 22001).

La longitud real del resultado es la misma que el atributo de longitud del resultado. Si la longitud de la *expresión-gráfica* es menor que el atributo de longitud del resultado, el resultado se rellena con espacios en blanco hasta llenar toda la longitud del resultado. Si la longitud de la *expresión-gráfica* es mayor que el atributo de longitud del resultado, se trunca sin devolver ningún aviso.

De fecha y hora a carácter

expresión-fecha-hora
Una expresión que sea uno de los tipos de datos siguientes:

DATE El resultado es la representación de serie de caracteres de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIME El resultado es la representación de serie de caracteres de la hora en el formato especificado por el segundo argumento. La longitud del resultado es 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIMESTAMP
El resultado es la representación de serie de caracteres de la indicación de fecha y hora. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(0), la longitud del resultado es 19. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(*n*), donde *n* es un número entre 1 y 12, la longitud del resultado es 20+*n*. De lo contrario, la longitud del resultado es 26. El segundo argumento no es aplicable y no se debe especificar (SQLSTATE 42815).

La página de códigos del resultado es la página de códigos de la sección.

Notas:

- La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones cuando el primer argumento es numérico o bien una serie y se especifica el argumento de longitud. Para obtener más información, consulte la "especificación CAST".
- Se permite una serie binaria como primer argumento de la función, y la serie de longitud fija resultante es una serie de caracteres FOR BIT DATA, rellenada con espacios en blanco, si es necesario.
- **De decimal a carácter y ceros iniciales**: en las versiones anteriores a la versión 9.7, el resultado de la entrada decimal en esta función incluye ceros iniciales y

un carácter decimal final. El parámetro de configuración de base de datos *dec_to_char_fmt* puede establecerse en “V95” para que esta función devuelva el resultado de la versión 9.5 para la entrada decimal. El valor por omisión de *dec_to_char_fmt* para las bases de datos nuevas es “NEW”, que tiene los resultados de retorno de esta función que coinciden con las reglas de conversión estándar de SQL y es coherente con los resultados de la función VARCHAR.

Ejemplos

- Supongamos que la columna PRSTDATE tiene un valor interno equivalente a 1988-12-25. La función siguiente devuelve el valor ‘12/25/1988’.

  ```
  CHAR(PRSTDATE, USA)
  ```

- Supongamos que la columna STARTING tiene un valor interno equivalente a 17:12:30 y que la variable del lenguaje principal HOUR_DUR (decimal(6,0)) es una duración en horas con un valor de 050000 (es decir, 5 horas). La función siguiente devuelve el valor ‘5:12 PM’.

  ```
  CHAR(STARTING, USA)
  ```

  La función siguiente devuelve el valor‘10:12 PM’.

  ```
  CHAR(STARTING + :HOUR_DUR, USA)
  ```

- Supongamos que la columna RECEIVED (TIMESTAMP) tiene un valor interno equivalente a la combinación de las columnas PRSTDATE y STARTING. La función siguiente devuelve el valor ‘1988-12-25-17.12.30.000000’.

  ```
  CHAR(RECEIVED)
  ```

- La columna LASTNAME está definida como VARCHAR(15). La función siguiente devuelve los valores de esta columna como series de caracteres de longitud fija de 10 bytes de longitud. Los valores de LASTNAME con una longitud superior a los 10 bytes (excluidos los blancos de cola) se truncan y se devuelve un aviso.

  ```
  SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
  ```

- La columna EDLEVEL está definida como SMALLINT. La función siguiente devuelve los valores de esta columna como series de caracteres de longitud fija. Se devuelve un valor de EDLEVEL de 18 como valor de CHAR(6) de ’18’ seguido por cuatro espacios en blanco.

  ```
  SELECT CHAR(EDLEVEL) FROM EMPLOYEE
  ```

- La columna SALARY se define como DECIMAL con una precisión de 9 y una escala de 2. El valor actual (18357.50) debe visualizarse con una coma como separador decimal (18357,50). La función siguiente devuelve el valor ‘18357,50’ seguido por tres espacios en blanco.

  ```
  CHAR(SALARY, ',')
  ```

- Los valores de la columna SALARY deben restarse de 20000.25 y visualizarse con el carácter decimal por omisión. La función siguiente devuelve el valor ‘-0001642.75’ seguido por tres espacios en blanco.

  ```
  CHAR(20000.25 - SALARY)
  ```

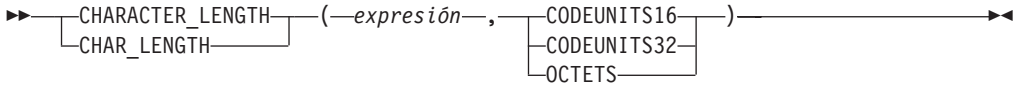
- Supongamos que la variable del lenguaje principal SEASONS_TICKETS está definida como INTEGER y tiene un valor de 10000. La función siguiente devuelve el valor ‘10000.00’.

  ```
  CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
  ```

- Supongamos que la variable del lenguaje principal DOUBLE_NUM está definida como DOUBLE y tiene un valor de -987.654321E-35. La función siguiente devuelve el valor ‘-9.87654321E-33’ seguido por nueve espacios en blanco finales porque el tipo de datos del resultado es CHAR(24).

  ```
  CHAR(:DOUBLE_NUM)
  ```

CHARACTER_LENGTH



El esquema es SYSIBM.

La función CHARACTER_LENGTH devuelve la longitud de la *expresión* en la unidad de la serie especificada.

expresión
Una expresión que devuelve un valor de un carácter incorporado o serie gráfica.

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie del resultado. CODEUNITS16 especifica que el resultado debe expresarse en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que el resultado debe expresarse en unidades de código UTF-32 de 32 bits. OCTETS especifica que el resultado debe expresarse en bytes.

Si la unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32 y la *expresión* es una serie binaria o datos de bits, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie se especifica como OCTETS y la *expresión* es una serie binaria, se devuelve un error (SQLSTATE 42815). Para obtener más información acerca de CODEUNITS16, CODEUNITS32 y OCTETS, consulte "Unidades de serie en las funciones incorporadas" en "Series de caracteres".

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La longitud de las series de caracteres y de gráficos incluye espacios en blanco finales. La longitud de las series de longitud variable es la longitud real y no la longitud máxima.

Ejemplos:

- Supongamos que NAME es una columna VARCHAR(128) codificada en Unicode UTF-8 que contiene el valor 'Jürgen'. Las dos consultas siguientes devuelven el valor 6:

```
SELECT CHARACTER_LENGTH(NAME, CODEUNITS32)
FROM T1 WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME, CODEUNITS16)
FROM T1 WHERE NAME = 'Jürgen'
```

  Las dos consultas siguientes devuelven el valor 7:

```
SELECT CHARACTER_LENGTH(NAME, OCTETS)
FROM T1 WHERE NAME = 'Jürgen'

SELECT LENGTH(NAME)
FROM T1 WHERE NAME = 'Jürgen'
```

- Los ejemplos siguientes funcionan con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

Supongamos que la variable UTF8_VAR contiene la representación UTF-8 de la serie.

```
SELECT CHARACTER_LENGTH(UTF8_VAR, CODEUNITS16),
  CHARACTER_LENGTH(UTF8_VAR, CODEUNITS32),
  CHARACTER_LENGTH(UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 6, 5 y 9, respectivamente.

Supongamos que la variable UTF16_VAR contiene la representación UTF-16BE de la serie.

```
SELECT CHARACTER_LENGTH(UTF16_VAR, CODEUNITS16),
   CHARACTER_LENGTH(UTF16_VAR, CODEUNITS32),
   CHARACTER_LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 6, 5 y 12, respectivamente.

CHR

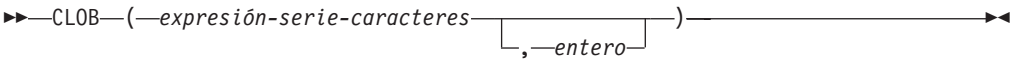
El esquema es SYSFUN.

Devuelve el carácter que tiene el valor del código ASCII especificado por el argumento. Si el valor de *expresión* es 0, el resultado es el carácter en blanco (X'20').

El argumento puede ser INTEGER o SMALLINT. El valor del argumento debe estar entre 0 y 255; de lo contrario, el valor de retorno es el carácter que tiene el valor de código ASCII correspondiente a 255.

El resultado de la función es CHAR(1). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

CLOB



El esquema es SYSIBM.

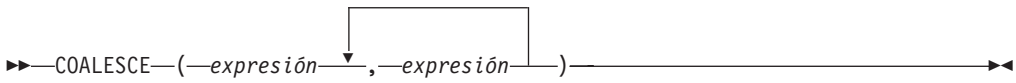
La función CLOB devuelve una representación CLOB de un tipo de serie de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a un tipo de datos de serie de caracteres antes de que se ejecute la función.

expresión-serie-caracteres
Expresión que devuelve un valor que es una serie de caracteres. La expresión no puede ser una serie de caracteres definida como FOR BIT DATA (SQLSTATE 42846).

entero
Un valor entero que especifica el atributo de longitud del tipo de datos CLOB resultante. El valor debe estar entre 0 y 2 147 483 647. Si no se especifica ningún valor para *entero*, la longitud del resultado es la misma que la longitud el primer argumento.

El resultado de la función es CLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

COALESCE



El esquema es SYSIBM.

COALESCE devuelve el primer argumento que no es nulo.

Los argumentos se evalúan en el orden en que se especifican, y el resultado de la función es el primer argumento que no es nulo. El resultado sólo puede ser nulo si todos los argumentos pueden ser nulos, y el resultado sólo es nulo si todos los argumentos son nulos. El argumento seleccionado se convierte, si es necesario, a los atributos del resultado.

Los argumentos deben ser compatibles. Puede ser de un tipo de datos interno o definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos de datos diferenciados definidos por el usuario).

Ejemplos:

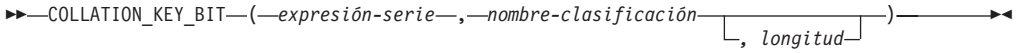
- Cuando se seleccionan todos los valores de todas las filas de la tabla DEPARTMENT, si falta el director del departamento (MGRNO) (es decir, es nulo), se ha de devolver un valor de 'ABSENT'.

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- Cuando se selecciona el número de empleado (EMPNO) y el salario (SALARY) de todas las filas de la tabla EMPLOYEE, si falta el salario (es decir, es nulo), se ha de devolver un valor de cero.

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

COLLATION_KEY_BIT



El esquema es SYSIBM.

La función COLLATION_KEY_BIT devuelve una serie VARCHAR FOR BIT DATA que representa la clave de clasificación de *expresión-serie* en el *nombre-clasificación* especificado.

En los resultados de COLLATION_KEY_BIT para dos series se pueden comparar los binarios para determinar el orden en el *nombre-clasificación* especificado. Para que la comparación sea significativa, los resultados utilizados deben ser del mismo *nombre-clasificación*.

expresión-serie
Expresión que devuelve una serie CHAR, VARCHAR, GRAPHIC o VARGRAPHIC para la que se debe determinar la clave de clasificación. Si *expresión-serie* es CHAR o VARCHAR, la expresión no debe ser FOR BIT DATA (SQLSTATE 429BM).

Si *expresión-serie* no está en UTF-16, esta función realiza la conversión de página de códigos de *expresión-serie* a UTF-16. Si el resultado de la conversión de página de códigos contiene como mínimo un carácter de sustitución, esta función devolverá una clave de clasificación de la serie UTF-16 con el carácter o los caracteres de sustitución y el distintivo de aviso SQLWARN8 de SQLCA se establecerá en 'W'.

nombre-clasificación
Constante de tipo carácter que especifica la clasificación a utilizar al determinar la clave de clasificación. el valor de *nombre-clasificación* no es sensible a las mayúsculas y minúsculas y debe ser una de las "clasificaciones basadas en algoritmos de clasificación Unicode" de *Globalization Guide* o "clasificaciones según el idioma para datos Unicode" de *Globalization Guide* (SQLSTATE 42616).

longitud
Expresión que especifica el atributo de longitud del resultado en bytes. Si se especifica, *longitud* debe ser un entero entre 1 y 32.672 (SQLSTATE 42815).

Si no se especifica un valor para *longitud*, la longitud del resultado se determina del modo siguiente:

Tabla 44. Determinación de la longitud del resultado

Tipo de datos del argumento de la serie	Longitud de tipo de datos de resultado
CHAR(*n*) o VARCHAR(*n*)	Mínimo de 12*n* bytes y 32.672 bytes
GRAPHIC(*n*) o VARGRAPHIC(*n*)	Mínimo de 12*n* bytes y 32.672 bytes

Independientemente de si se especifica *longitud*, si la longitud de la clave de clasificación es mayor que la longitud del tipo de datos de resultado, se devuelve un error (SQLSTATE 42815). La longitud de resultado real de la clave de clasificación es aproximadamente seis veces la longitud de *expresión-serie* después de que ésta se haya convertido a UTF-16.

Si *serie-expresión* es una serie vacía, el resultado es una clave de clasificación válida que puede tener una longitud distinta de cero.

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

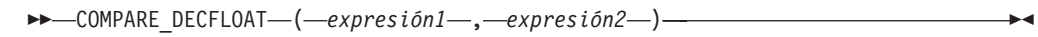
La consulta siguiente ordena los empleados por los apellidos utilizando la clasificación según idioma para alemán de la página de códigos 923:

```
SELECT FIRSTNME, LASTNAME
  FROM EMPLOYEE
  ORDER BY COLLATION_KEY_BIT (LASTNAME, 'SYSTEM_923_DE')
```

La siguiente consulta utiliza una comparación culturalmente correcta para buscar los departamentos de empleados de la provincia de Quebec:

```
SELECT E.WORKDEPT
  FROM EMPLOYEE AS E INNER JOIN SALES AS S
  ON COLLATION_KEY_BIT(E.LASTNAME, 'UCA400R1_LFR') =
    COLLATION_KEY_BIT(S.SALES_PERSON, 'UCA400R1_LFR')
  WHERE S.REGION = 'Quebec'
```

COMPARE_DECFLOAT



El esquema es SYSIBM.

La función COMPARE_DECFLOAT devuelve un valor SMALLINT que indica si los dos argumentos son iguales o desordenados o si un argumento es superior a otro.

expresión1
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es DECFLOAT(34), se convierte de forma lógica a DECFLOAT(34) para procesarse.

expresión2
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es DECFLOAT(34), se convierte de forma lógica a DECFLOAT(34) para procesarse.

El valor de *expresión1* se compara con el valor de *expresión2* y el resultado se devuelve de acuerdo con las normas siguientes:

- Si ambos argumentos son finitos, la comparación será algebraica y seguirá el procedimiento para la resta de coma flotante decimal. Si la diferencia es exactamente cero con cualquiera de los dos signos, los argumentos son iguales. Si una diferencia distinta de cero es positiva, el primer argumento es mayor que el segundo argumento. Si una diferencia distinta de cero es negativa, el primer argumento es menor que el segundo.
- El cero positivo y el cero negativo se comparan como iguales.
- El infinito positivo se compara como igual al infinito positivo.
- El infinito positivo se compara como superior a cualquier número finito.
- El infinito negativo se compara como igual al infinito negativo.
- El infinito negativo se compara como inferior a cualquier número finito.
- La comparación numérica es exacta. El resultado se determina para operandos finitos como si el rango y la precisión fueran ilimitados. No se puede producir ninguna condición de desbordamiento o subdesbordamiento.
- Si uno de los dos argumentos es NaN o sNaN (positivo o negativo), el resultado está desordenado.

El valor de resultado es el siguiente:

- 0 si los argumentos son exactamente iguales
- 1 si *expresión1* es menor que *expresión2*
- 2 si *expresión1* es mayor que *expresión2*
- 3 si los argumentos no están ordenados

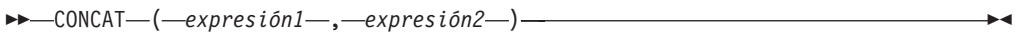
El resultado de la función es un valor SMALLINT. Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Los ejemplos siguientes muestran los valores que devuelve la función COMPARE_DECFLOAT, si se da una variedad de valores de coma flotante decimal de entrada:

```
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.17)) = 0
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.170)) = 2
COMPARE_DECFLOAT(DECFLOAT(2.170), DECFLOAT(2.17)) = 1
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(0.0)) = 2
COMPARE_DECFLOAT(INFINITY,INFINITY) = 0
COMPARE_DECFLOAT(INFINITY,-INFINITY) = 2
COMPARE_DECFLOAT(DECFLOAT(-2),INFINITY) = 1
COMPARE_DECFLOAT(NAN,NAN) = 3
COMPARE_DECFLOAT(DECFLOAT(-0.1),SNAN) = 3
```

CONCAT



El esquema es SYSIBM.

La función CONCAT combina dos argumentos para formar una expresión de serie.

expresión1
Una expresión que devuelve un valor de cualquier tipo de datos de serie, tipo de datos numérico o tipo de datos de fecha y hora.

expresión2
Una expresión que devuelve un valor de cualquier tipo de datos de serie, tipo de datos numérico o tipo de datos de fecha y hora. Sin embargo, algunos tipos de datos no están soportados en combinación con el tipo de datos de *expresión1*, como se describe a continuación.

Los argumentos pueden ser cualquier combinación de valores de serie (excepto series binarias), numéricos y de fecha y hora. Cuando un argumento no es un valor de serie, se convierte de forma implícita en VARCHAR. Una serie binaria sólo se puede concatenar con otra serie binaria. No obstante, a través del proceso convertible de resolución de función, una serie binaria se puede concatenar con una serie de caracteres definida como FOR BIT DATA cuando el primer argumento es la serie binaria.

Las concatenaciones que implican tanto un argumento de serie de caracteres como un argumento de serie gráfica sólo reciben soporte en una base de datos Unicode. El argumento de caracteres primero se convertirá en el tipo de datos gráficos antes de la concatenación. Las series de caracteres definidas como FOR BIT DATA no se pueden convertir en el tipo de datos gráfico.

El resultado de la función es una serie que consta del primer argumento seguido del segundo argumento. El tipo de datos y la longitud del resultado vienen determinados por los tipos de datos y las longitudes de los argumentos, después de que se realicen las conversiones que procedan. Para obtener más información, consulte la tabla “Tipo de datos y longitud de los operandos concatenados” en el tema “Expresiones”.

Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Notas

- No se realiza ninguna comprobación para detectar los datos mixtos formados defectuosamente al realizar la concatenación.
- La función CONCAT es idéntica al operador CONCAT. Para obtener más información, consulte “Expresiones”.

Ejemplo

- Concatenar la columna FIRSTNME con la columna LASTNAME.

```
SELECT CONCAT(FIRSTNME, LASTNAME)
FROM EMPLOYEE
WHERE EMPNO = '000010'
```

  Devuelve el valor “CHRISTINEHAAS”.

COS

El esquema es SYSIBM. (La versión SYSFUN de la función COS continúa estando disponible).

Devuelve el coseno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

COSH

El esquema es SYSIBM.

Devuelve el coseno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

COT

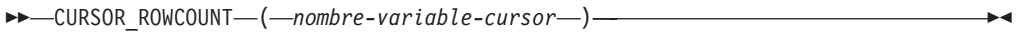
El esquema es SYSIBM. (La versión SYSFUN de la función COT continúa estando disponible).

Devuelve la cotangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo numérico incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

CURSOR_ROWCOUNT



El esquema es SYSIBM.

La función CURSOR_ROWCOUNT devuelve el número acumulado de todas las filas captadas por el cursor especificado desde que dicho cursor se abrió.

nombre-variable-cursor
Nombre de una variable de SQL o un parámetro de SQL de un tipo de cursor. El cursor subyacente del *nombre-variable-cursor* debe estar abierto (SQLSTATE 24501).

El resultado es 0 si no se ha realizado ninguna acción FETCH sobre el cursor subyacente del nombre-variable-cursor antes de la evaluación de la función.

Esta función sólo puede utilizarse en una sentencia de SQL compuesto (compilado).

El tipo de datos del resultado es BIGINT. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos

En el ejemplo siguiente se muestra cómo utilizar la función para recuperar el número de filas asociadas con el cursor *curEmp* y asignarlo a una variable denominada *rows_fetched*:

```
SET rows_fetched = CURSOR_ROWCOUNT(curEmp);
```

DATAPARTITIONNUM

El esquema es SYSIBM.

La función DATAPARTITIONNUM devuelve el número de secuencia (SYSDATAPARTITIONS.SEQNO) de la partición de datos donde reside la fila. Las particiones de datos se clasifican por rango y los números de secuencia empiezan en 0. Por ejemplo, la función DATAPARTITIONNUM devuelve 0 para una fila que reside en la partición de datos con el rango más bajo.

El argumento debe ser el nombre calificado o no calificado de cualquier columna de la tabla. Puesto que se devuelve información a nivel de fila, el resultado es el mismo, sin tener en cuenta qué columna se especifica. La columna puede tener cualquier tipo de datos.

Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas normas que una vista.

El tipo de datos del resultado es INTEGER y nunca es nulo.

Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Ya que la función acepta cualquier tipo de datos como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos diferentes definidos por el usuario.

La función DATAPARTITIONNUM no se puede utilizar en las restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881). No se puede utilizar la función DATAPARTITIONNUM en la definición de una tabla de consulta materializada (MQT) (SQLSTATE 428EC).

Ejemplos

- *Ejemplo 1:* recuperar el número de secuencia de la partición de datos en la que reside la fila para EMPLOYEE.EMPNO.

```
SELECT DATAPARTITIONNUM (EMPNO)
FROM EMPLOYEE
```

- *Ejemplo 2:* para convertir un número se secuencia que DATAPARTITIONNUM devuelve (por ejemplo, 0) en un nombre de partición de datos que pueda utilizarse en otras sentencias de SQL (como ALTER TABLE...DETACH PARTITION), puede consultar la vista de catálogo SYSCAT.DATAPARTITIONS. Incluya el valor de SEQNO obtenido de DATAPARTITIONNUM en la cláusula WHERE, tal como se muestra en el ejemplo siguiente.

```
SELECT DATAPARTITIONNAME
   FROM SYSCAT.DATAPARTITIONS
   WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
```

  genera el valor 'PART0'.

DATE

El esquema es SYSIBM.

La función DATE devuelve una fecha de un valor.

El argumento debe ser un valor DATE, TIMESTAMP, un número positivo menor que o igual a 3.652.059, una representación de serie válida de una fecha o indicación de fecha y hora o una serie de longitud 7 que no sea un CLOB ni un DBCLOB.

Sólo las bases de datos Unicode dan soporte a un argumento que es una representación de serie gráfica de una fecha o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Si el argumento es una serie de longitud 7, debe representar una fecha válida en el formato *aaaannn*, donde *aaaa* son los dígitos que indican el año y *nnn* los son dígitos entre 001 y 366, que indican un día de dicho año.

El resultado de la función es un valor DATE. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

- Si el argumento es un valor DATE, TIMESTAMP o una representación de serie válida de una fecha o indicación de fecha y hora:
  - El resultado es la parte correspondiente a la fecha del valor.
- Si el argumento es un número:
  - El resultado es la fecha de *n*-1 días después de 1 de enero de 0001, donde *n* es la parte integral del número.
- Si el argumento es una serie con una longitud de 7:
  - El resultado es la fecha representada por la serie.

Ejemplos

Supongamos que la columna RECEIVED (cuyo tipo de datos es TIMESTAMP) tiene un valor interno equivalente a '1988-12-25-17.12.30.000000'.

- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE(RECEIVED)
  ```
- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE('1988-12-25')
  ```
- Este ejemplo da como resultado una representación interna de '1988-12-25'.
  ```
  DATE('25.12.1988')
  ```
- Este ejemplo da como resultado una representación interna de '0001-02-04'.
  ```
  DATE(35)
  ```

DAY

El esquema es SYSIBM.

La función DAY devuelve la parte correspondiente al día de un valor.

El argumento debe ser un valor DATE, TIMESTAMP, una duración de fecha, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

- Si el argumento es un valor DATE, TIMESTAMP o una representación de serie válida de una fecha o indicación de fecha y hora:
  - El resultado es la parte correspondiente al día del valor, que es un entero entre 1 y 31.
- Si el argumento es una duración de fecha o duración de la indicación de fecha y hora:
  - El resultado es la parte correspondiente al día del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal END_DAY (smallint) en el día en que está planificado que el proyecto WELD LINE PLANNING (PROJNAME) finalice (PRENDATE).

  ```
  SELECT DAY(PRENDATE)
    INTO :END_DAY
    FROM PROJECT
    WHERE PROJNAME = 'WELD LINE PLANNING'
  ```

  Da como resultado que END_DAY se establece en 15 cuando se utiliza la tabla de ejemplo.
- Supongamos que la columna DATE1 (cuyo tipo de datos es DATE) tiene un valor interno equivalente a 2000-03-15 y la columna DATE2 (cuyo tipo de datos es DATE) tiene un valor interno equivalente a 1999-12-31.

  ```
  DAY(DATE1 - DATE2)
  ```

  Da como resultado el valor 15.

DAYNAME

El esquema es SYSIBM. La versión SYSFUN de la función DAYNAME continúa estando disponible.

La función DAYNAME devuelve una serie de caracteres que contiene el nombre del día (por ejemplo, viernes) para la parte del día de la *expresión*, basada en el *nombre-entorno-local* o el valor del registro especial CURRENT LOCALE LC_TIME.

expresión
Expresión que devuelve un valor de uno de los tipos de datos incorporados siguientes: DATE, TIMESTAMP o una representación de serie de caracteres de una fecha o una indicación de fecha y hora que no sea CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local utilizado para determinar el idioma del resultado. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado es una serie de caracteres de longitud variable. El atributo de longitud es 100. Si la serie resultante supera el atributo de longitud del resultado, el resultado se truncará. Si el argumento *expresión* puede ser nulo, el resultado puede ser nulo; si el argumento *expresión* es nulo, el resultado es el valor nulo. La página de códigos del resultado es la página de códigos de la sección.

Notas

- **Calendario juliano y gregoriano**: esta función tiene en cuenta la transición del calendario juliano al calendario gregoriano el 15 de octubre de 1582. Sin embargo, la versión SYSFUN de la función DAYNAME adopta el calendario gregoriano para todos los cálculos.
- **Determinismo:** DAYNAME es una función determinista. No obstante, cuando no se especifica de forma explícita *nombre-entorno-local*, la invocación de la función depende del valor del registro especial CURRENT LOCALE LC_TIME. Esta invocación que depende del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales (SQLSTATE 42621 o 428EC).

Ejemplo

- Suponga que la variable TMSTAMP se ha definido como TIMESTAMP y tiene el valor siguiente: 2007-03-09-14.07.38.123456. Los ejemplos siguientes muestran varias invocaciones de la función y los valores de serie resultantes. En cada caso el tipo de resultado es VARCHAR(100).

```
Invocación de función                          Resultado
-------------------------                     ----------
DAYNAME (TMSTAMP, 'CLDR 1.5:en_US')            Friday
DAYNAME (TSMTAMP, 'CLDR 1.5:de_DE')            Freitag
DAYNAME (TMSTAMP, 'CLDR 1.5:fr_FR')            vendredi
```

DAYOFWEEK

El esquema es SYSFUN.

La función DAYOFWEEK devuelve el día de la semana del argumento como un valor entero comprendido entre 1 y 7, donde 1 representa el domingo.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFWEEK_ISO

El esquema es SYSFUN.

Devuelve el día de la semana del argumento, en forma de valor entero comprendido dentro del rango 1-7, donde 1 representa el lunes.

El argumento debe ser un valor DATE, TIMESTAMP o bien una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYOFYEAR

El esquema es SYSFUN.

Devuelve el día del año del argumento como un valor entero en el rango de 1 a 366.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

DAYS

El esquema es SYSIBM.

La función DAYS devuelve una representación de entero de una fecha.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado es 1 más que el número de días desde el 1 de enero de 0001 hasta *D*, donde *D* es la fecha que podría darse si se aplicase la función DATE al argumento.

Ejemplos

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal EDUCATION_DAYS (int) en el número de días transcurridos (PRENDATE - PRSTDATE) estimados para el proyecto (PROJNO) 'IF2000'.

```
SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
  INTO :EDUCATION_DAYS
  FROM PROJECT
  WHERE PROJNO = 'IF2000'
```

  El resultado de EDUCATION_DAYS se define en 396.

- Utilizando la tabla PROJECT, establezca la variable del lenguaje principal TOTAL_DAYS (int) en la suma de los días transcurridos (PRENDATE - PRSTDATE) estimados para todos los proyectos del departamento (DEPTNO) 'E21'.

```
SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
  INTO :TOTAL_DAYS
  FROM PROJECT
  WHERE DEPTNO = 'E21'
```

  Da como resultado que TOTAL_DAYS se establece en 1584 cuando se utiliza la tabla de ejemplo.

DBCLOB

El esquema es SYSIBM.

La función DBCLOB devuelve una representación DBCLOB de un tipo de serie gráfica.

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter:

- Se deja tal cual, si el argumento proporcionado es una serie de caracteres
- O se convierte al carácter en blanco (X'0020'), si el argumento proporcionado es una serie gráfica

No confíe en estos comportamientos, porque podrían cambiar en los releases futuros.

El resultado de la función es DBCLOB. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

expresión-gráfica
Una expresión que devuelve un valor que es una serie gráfica.

entero
Un valor entero que especifica el atributo de longitud del tipo de datos DBCLOB resultante. El valor debe estar entre 0 y 1 073 741 823. Si no se especifica *entero*, la longitud del resultado es la misma que la longitud del primer argumento.

DBPARTITIONNUM

El esquema es SYSIBM.

La función DBPARTITIONNUM devuelve el número de partición de base de datos para una fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el número de partición de la base de datos para cada fila del conjunto de resultados.

El argumento debe ser el nombre calificado o no calificado de cualquier columna de la tabla. Puesto que se devuelve información a nivel de fila, el resultado es el mismo, sin tener en cuenta qué columna se especifica. La columna puede tener cualquier tipo de datos.

Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas normas que una vista.

La fila (y la tabla) específica para la que la función DBPARTITIONNUM devuelve el número de partición de base de datos se determina a partir del contexto de la sentencia SQL que utiliza la función.

El número de partición de base de datos devuelto en las variables y las tablas de partición se deriva de los valores de transición actuales de las columnas de claves de distribución. Por ejemplo, en un activador BEFORE INSERT, la función devuelve el número de partición de base de datos proyectado que corresponda a los valores actuales de las nuevas variables de transición. No obstante, es posible que los valores de las columnas de claves de distribución se modifiquen mediante un activador BEFORE INSERT subsiguiente. Por lo tanto, el número de partición de base de datos final de la fila cuando se inserta en la base de datos puede ser distinto del valor proyectado.

El tipo de datos del resultado es INTEGER y nunca es nulo. Si no hay ningún archivo `db2nodes.cfg`, el resultado es 0.

Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Ya que la función acepta cualquier tipo de datos como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos diferentes definidos por el usuario.

La función DBPARTITIONNUM no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Para mantener la compatibilidad con las versiones anteriores de los productos DB2, se puede especificar NODENUMBER en lugar de DBPARTITIONNUM.

Ejemplos:

- Cuente el número de instancias en las que la fila para un empleado de la tabla EMPLOYEE está en una partición de base de datos distinta a la descripción del departamento del empleado de la tabla DEPARTMENT.

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- Unir las tablas EMPLOYEE y DEPARTMENT de manera que las filas de las dos tablas se encuentren en la misma partición de base de datos.

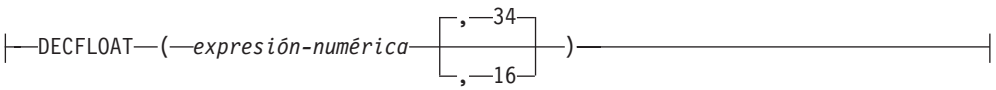
```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- Utilizando un activador anterior en la tabla EMPLOYEE, registrar el número de empleado y el número de partición de base de datos proyectado de cualquier fila nueva de la tabla EMPLOYEE en una tabla denominada EMPINSERTLOG1.

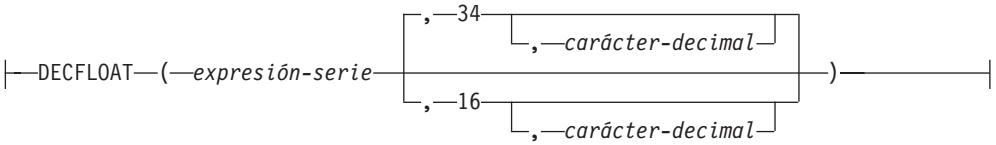
```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

DECFLOAT

De numérico a coma flotante decimal:



De carácter a coma flotante decimal:



El esquema es SYSIBM.

La función DECFLOAT devuelve una representación de coma flotante decimal de un número o representación de serie de un número.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

expresión-serie
Una expresión que devuelve un valor de serie de caracteres o la representación de serie gráfica Unicode de un número con una longitud no superior a la longitud máxima de una constante de caracteres. El tipo de datos de expresión-serie no debe ser CLOB ni DBCLOB (SQLSTATE 42884). Los blancos iniciales o de cola se eliminan de la serie. La subserie resultante debe ajustarse a las normas para formar una constante de coma flotante decimal, de coma flotante, de decimal y de entero (SQLSTATE 22018) y no debe tener más de 42 bytes (SQLSTATE 42820).

34 ó 16
Especifica el número de dígitos de precisión para el resultado. El valor por omisión es 34.

carácter-decimal
Especifica la constante de caracteres de un solo byte utilizada para delimitar los dígitos decimales en *expresión-caracteres* de la parte correspondiente a los enteros del número. El carácter no puede ser un dígito, el signo más (+), el signo menos (-) ni un blanco y puede aparecer como máximo una vez en *expresión-caracteres*

El resultado es el mismo número que sería el resultado de `CAST(`*`expresión-serie`*` AS DECFLOAT(`*`n`*`))` o `CAST(`*`expresión-numérica`*` AS DECFLOAT(`*`n`*`))`. Los blancos iniciales o de cola se eliminan de la serie.

El resultado de la función es un número de coma flotante decimal con el número de dígitos de precisión especificado de forma implícita o explícita. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Si es necesario, la fuente se redondea a la precisión del destino. El registro especial CURRENT DECFLOAT ROUNDING MODE determina la modalidad de redondeo.

Notas

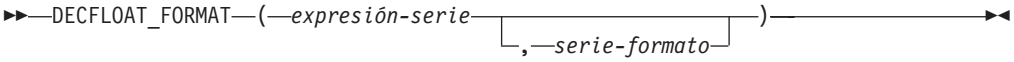
- La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".
- Todos los valores numéricos se interpretan como constantes de coma flotante, decimal o entero y después convertirse a la coma flotante decimal. El uso de una constante de coma flotante puede dar como resultado un redondeo de errores y por tanto se desaconseja vivamente. En su lugar, utilice la serie para la versión de coma flotante decimal de la función DECFLOAT.

Ejemplos

- Utilice la función DECFLOAT a fin de forzar que se devuelva un tipo de datos DECFLOAT en una lista de selección para la columna EDLEVEL (tipo de datos = SMALLINT) en la tabla EMPLOYEE. La columna EMPNO debe aparecer también en la lista de selección.

```
SELECT EMPNO, DECFLOAT(EDLEVEL,16)
FROM EMPLOYEE
```

DECFLOAT_FORMAT



El esquema es SYSIBM.

La función DECFLOAT_FORMAT devuelve un valor DECFLOAT(34) que se basa en la interpretación de la serie de entrada utilizando el formato especificado.

expresión-serie
Expresión que devuelve un valor que es un tipo de datos CHAR y VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función. Los blancos iniciales o de cola se eliminan de la serie. Si no se especifica *serie-formato*, la subserie resultante debe ajustarse a las normas para formar una constante de coma flotante decimal, de coma flotante, decimal o de entero SQL (SQLSTATE 22018) y no debe tener más de 42 bytes (SQLSTATE 42820); en caso contrario, la subserie resultante debe contener los componentes de un número que correspondan al formato especificado en *serie-formato* (SQLSTATE 22018).

serie-formato
Expresión que devuelve un valor que es un tipo de datos de serie de caracteres incorporados (excepto CLOB). En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se evalúe la función. La longitud real no debe superar los 254 bytes (SQLSTATE 22018). El valor es una plantilla para que pueda interpretarse la *expresión-serie* y luego convertirse en un valor de DECFLOAT. Los elementos de formato especificados como prefijo sólo pueden utilizarse al principio de la plantilla. Los elementos de formato especificados como sufijo sólo pueden utilizarse al final de la plantilla. Los elementos de formato son sensibles a mayúsculas y minúsculas. La plantilla no debe contener más de uno de los elementos de formato MI, S o PR (SQLSTATE 22018).

Tabla 45. Elementos de formato para la función DECFLOAT_FORMAT

Elemento de formato	Descripción
0 ó 9	Cada 0 y cada 9 representa un dígito.
MI	Sufijo: si *expresión-serie* representa un número negativo, se espera un signo menos (-) final. Si *expresión-serie* representa un número positivo, se puede especificar un espacio en blanco final.
S	Prefijo: si *expresión-serie* representa un número negativo, se espera un signo menos (-) inicial. Si *expresión-serie* representa un número positivo, se puede especificar un signo más (+) inicial o un espacio en blanco inicial.
PR	sufijo: si *expresión-serie* representa un número negativo, se espera un carácter de menor que (<) inicial y un carácter de mayor que (>) final. Si *expresión-serie* representa un número positivo, se puede especificar un espacio inicial y un espacio final.
$	Prefijo: se debe especificar un signo de dólar inicial ($).
,	Especifica la ubicación esperada de una coma. Esta coma se utiliza como separador de grupos.

Tabla 45. Elementos de formato para la función DECFLOAT_FORMAT (continuación)

Elemento de formato	Descripción
.	Especifica la ubicación esperada del punto. Este punto se utiliza como separador decimal.

Si no se especifica *serie-formato*, *expresión-serie* debe ajustarse a las normas para formar una constante de coma flotante decimal, de coma flotante, decimal o de entero SQL (SQLSTATE 22018) y tener una longitud que no supere los 42 bytes (SQLSTATE 42820).

El resultado es DECFLOAT(34). Si el primer argumento o el segundo puede ser nulo, el resultado puede ser nulo; si el primer argumento o el segundo es nulo, el resultado es el valor nulo.

Notas

- Alternativas de la sintaxis: TO_NUMBER es sinónimo de DECFLOAT_FORMAT.

Ejemplos

- `DECFLOAT_FORMAT( '123.45' )`

  devuelve 123.45
- `DECFLOAT_FORMAT( '-123456.78' )`

  devuelve -123456.78
- `DECFLOAT_FORMAT( '+123456.78' )`

  devuelve 123456.78
- `DECFLOAT_FORMAT( '1.23E4' )`

  devuelve 12300
- `DECFLOAT_FORMAT( '123.4', '9999.99' )`

  devuelve 123.40
- `DECFLOAT_FORMAT( '001,234', '000,000' )`

  devuelve 1234
- `DECFLOAT_FORMAT( '1234 ', '9999MI' )`

  devuelve 1234
- `DECFLOAT_FORMAT( '1234-', '9999MI' )`

  devuelve -1234
- `DECFLOAT_FORMAT( '+1234', 'S9999' )`

  devuelve 1234
- `DECFLOAT_FORMAT( '-1234', 'S9999' )`

  devuelve -1234
- `DECFLOAT_FORMAT( ' 1234 ', '9999PR' )`

  devuelve 1234
- `DECFLOAT_FORMAT( '<1234>', '9999PR' )`

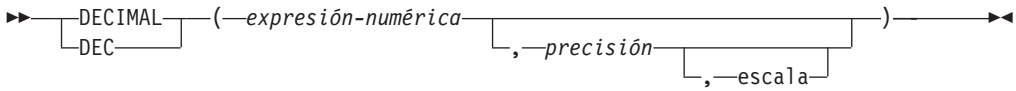
  devuelve -1234

- **DECFLOAT_FORMAT**('$123,456.78', '$999,999.99')

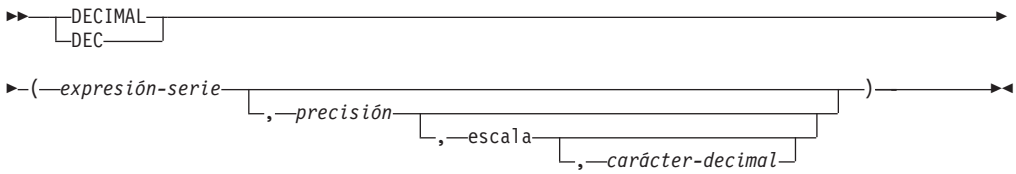
  devuelve 123456.78

DECIMAL o DEC

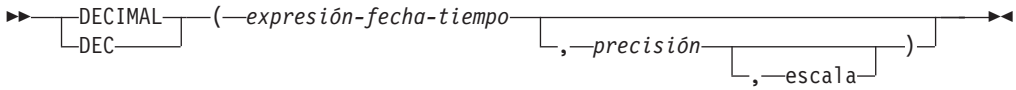
De numérico a decimal :



De serie a decimal:



De fecha y hora a decimal:



El esquema es SYSIBM.

La función DECIMAL devuelve una representación decimal de:

- Un número
- Una representación de serie de un número
- Un valor de fecha y hora

De numérico a decimal

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

precisión
Una constante de enteros con un valor en el rango de 1 a 31.

El valor por omisión para *precisión* depende del tipo de datos de *expresión-numérica*:

- 31 para coma flotante decimal
- 15 para coma flotante y decimal
- 19 para entero superior
- 11 para entero grande
- 5 para entero pequeño.

escala
Una constante de entero con un valor entre 0 y el valor de *precisión*. El valor por omisión es cero.

El resultado es el mismo número que se generaría si el primer argumento se asignara a una columna o variable decimal con una precisión con el valor *precisión* y una escala con el valor *escala*. Se devuelve un error si el

número de dígitos decimales significativos necesarios para representar la parte entera del número es superior al valor de *precisión - escala* (SQLSTATE 22003).

De serie a decimal

expresión-serie
Una *expresión* que devuelve un valor que es una serie de caracteres o una representación de serie gráfica Unicode de un número con una longitud no superior a la longitud máxima de una constante de caracteres. El tipo de datos de *expresión-serie* no debe ser CLOB o DBCLOB (SQLSTATE 42884). Los espacios en blanco iniciales y finales se eliminan de la serie y la serie resultante debe ajustarse a las normas para formar una constante de entero, decimal, de coma flotante o de coma flotante decimal (SQLSTATE 22018).

La *expresión-serie* se convierte a la página de códigos de la sección si es necesario para que coincida con la página de códigos del *carácter-decimal* de la constante.

precisión
Una constante de enteros con un valor en el rango de 1 a 31 que especifica la precisión del resultado. Si no se especifica, el valor por omisión es 15.

escala
Una constante de enteros con un valor en el rango de 0 a *precisión* que especifica la escala del resultado. Si no se especifica, el valor por omisión es 0.

carácter-decimal
Especifica la constante de caracteres de un solo byte utilizada para delimitar los dígitos decimales en *expresión-serie* de la parte correspondiente a los enteros del número. El carácter no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco y puede aparecer como máximo una vez en *expresión-caracteres* (SQLSTATE 42815).

El resultado es el mismo número que generaría CAST(*expresión-serie* AS DECIMAL(*precisión*, *escala*)). Los dígitos se truncan al final del número decimal si el número de dígitos a la derecha del carácter separador de decimales es superior al valor de *escala*. Se devuelve un error si el número de dígitos significativos a la izquierda del carácter decimal (la parte entera del número) de *expresión-serie* es superior a *precisión - escala* (SQLSTATE 22003). El carácter decimal por omisión no es válido en la subserie si se especifica un valor diferente para el argumento *carácter-decimal* (SQLSTATE 22018).

De fecha y hora a decimal

expresión-fecha-hora
Una expresión que devuelve un valor del tipo DATE, TIME o TIMESTAMP.

precisión
Una constante de enteros con un valor en el rango de 1 a 31 que especifica la precisión del resultado. Si no se especifica, el valor por omisión para la precisión y la escala depende del tipo de datos de *expresión-fecha-hora* de la manera siguiente:

- La precisión es 8 y la escala es 0 para DATE. El resultado es un valor DECIMAL(8,0) que representa la fecha como *aaaammdd*.
- La precisión es 6 y la escala es 0 para TIME. El resultado es un valor DECIMAL(6,0) que representa la hora como *hhmmss*.
- La precisión es 14+*tp* y la escala es *tp* para TIMESTAMP(*tp*). El resultado es un valor DECIMAL(14+*tp*,*tp*) que representa la indicación de fecha y hora como *aaaammddhhmmss.nnnnnnnnnnnn*.

escala
Una constante de enteros con un valor en el rango de 0 a *precisión* que especifica la escala del resultado. Si no se especifica y se especifica una *precisión*, el valor por omisión es 0.

El resultado es el mismo número que generaría CAST(*fecha_hora - expresión* AS DECIMAL(*precisión*, *escala*)). Los dígitos se truncan al final del número decimal si el número de dígitos a la derecha del carácter separador de decimales es superior al valor de *escala*. Se devuelve un error si el número de dígitos significativos a la izquierda del carácter decimal (la parte entera del número) de *expresión-serie* es superior a *precisión - escala* (SQLSTATE 22003).

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".

Ejemplos

- Utilice la función DECIMAL para forzar a que se devuelva un tipo de datos DECIMAL (con una precisión de 5 y una escala de 2) en una lista-selección para la columna EDLEVEL (tipo de datos = SMALLINT) en la tabla EMPLOYEE. La columna EMPNO debe aparecer también en la lista de selección.

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- Supongamos que la variable del lenguaje principal PERIOD es de tipo INTEGER. En este caso, para utilizar su valor como duración de fecha debe convertirse a decimal(8,0).

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- Supongamos que las actualizaciones en la columna SALARY se entran mediante una ventana como una serie de caracteres que utiliza la coma como carácter decimal (por ejemplo, el usuario entra 21400,50). Cuando se ha validado por la aplicación,se asigna a la variable del lenguaje principal *newsalary* definida como CHAR(10).

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

  El valor de *newsalary* se convierte en 21400.50.

- Añada el carácter decimal por omisión (.) al valor.

```
DECIMAL('21400,50', 9, 2, '.')
```

  Falla porque se especifica un punto (.) como el carácter decimal, pero aparece una coma (,) en el primer argumento como delimitador.

- Supongamos que la columna STARTING (cuyo tipo de datos es TIME) tiene un valor interno equivalente a '12:10:00'.

  ```
  DECIMAL(STARTING)
  ```

  da como resultado el valor `121 000`.
- Supongamos que la columna RECEIVED (cuyo tipo de datos es TIMESTAMP) tiene un valor interno equivalente a '1988-12-22-14.07.21.136421'.

  ```
  DECIMAL(RECEIVED)
  ```

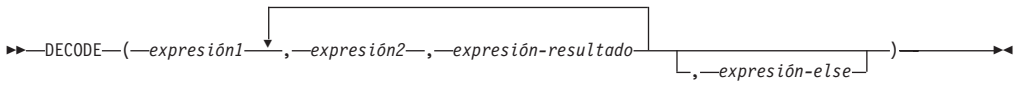
  da como resultado el valor `19 881 222 140 721.136421`.
- En este ejemplo se muestra el resultado decimal y la precisión y la escala resultante para varios valores de entrada de fecha y hora. Supongamos que existen las columnas siguientes con los valores asociados:

Nombre de columna	Tipo de datos	Valor
ENDDT	DATE	2000-03-21
ENDTM	TIME	12:02:21
ENDTS	TIMESTAMP	2000-03-21-12.02.21.123456
ENDTS0	TIMESTAMP(0)	2000-03-21-12.02.21
ENDTS9	TIMESTAMP(9)	2000-03-21-12.02.21.123456789

La tabla siguiente muestra el resultado decimal y la precisión y la escala resultante para varios valores de entrada de fecha y hora.

DECIMAL(argumentos)	Precisión y escala	Resultado
DECIMAL(ENDDT)	(8,0)	20000321.
DECIMAL(ENDDT, 10)	(10,0)	20000321.
DECIMAL(ENDDT, 12, 2)	(12,2)	20000321.00
DECIMAL(ENDTM)	(6,0)	120221.
DECIMAL(ENDTM, 10)	(10,0)	120221.
DECIMAL(ENDTM, 10, 2)	(10,2)	120221.00
DECIMAL(ENDTS)	(20, 6)	20000321120221.123456
DECIMAL(ENDTS, 23)	(23, 0)	20000321120221.
DECIMAL(ENDTS, 23, 4)	(23, 4)	20000321120221.1234
DECIMAL(ENDTS0)	(14,0)	20000321120221.
DECIMAL(ENDTS9)	(23,9)	20000321120221.123456789

DECODE



El esquema es SYSIBM.

La función DECODE compara cada *expresión2* con la *expresión1*. Si *expresión1* es igual a *expresión2*, o *expresión1* y *expresión2* son nulas, se devuelve el valor de la *expresión-resultado* siguiente. Si ninguna *expresión2* coincide con *expresión1*, se devuelve el valor de *expresión-else*; de lo contrario, se devuelve un valor nulo.

La función DECODE es similar a la expresión CASE excepto en el manejo de los valores nulos:

- Un valor nulo de *expresión1* coincidirá con un valor nulo correspondiente de *expresión2*.
- Si se utiliza la palabra clave NULL como argumento de la función DECODE, se deberá convertir en un tipo de datos apropiado.

Las normas para determinar el tipo de resultado de una expresión DECODE se basan en la expresión CASE correspondiente.

Ejemplos:

La expresión DECODE:

```
DECODE (c1, 7, 'a', 6, 'b', 'c')
```

obtiene el mismo resultado que la expresión CASE siguiente:

```
CASE c1
  WHEN 7 THEN 'a'
  WHEN 6 THEN 'b'
  ELSE 'c'
END
```

De forma similar, la expresión DECODE:

```
DECODE (c1, var1, 'a', var2, 'b')
```

donde los valores de c1, var1 y var2 pueden ser valores nulos, obtiene el mismo resultado que la expresión CASE siguiente:

```
CASE
  WHEN c1 = var1 OR (c1 IS NULL AND var1 IS NULL) THEN 'a'
  WHEN c1 = var2 OR (c1 IS NULL AND var2 IS NULL) THEN 'b'
  ELSE NULL
END
```

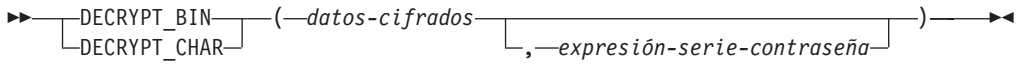
Examine también la consulta siguiente:

```
SELECT ID, DECODE(STATUS, 'A', 'Accepted',
                          'D', 'Denied',
                          CAST(NULL AS VARCHAR(1)), 'Unknown',
                          'Other')
FROM CONTRACTS
```

A continuación se muestra la misma sentencia utilizando una expresión CASE:

```
SELECT ID,
  CASE
    WHEN STATUS = 'A' THEN 'Accepted'
    WHEN STATUS = 'D' THEN 'Denied'
    WHEN STATUS IS NULL THEN 'Unknown'
    ELSE 'Other'
  END
FROM CONTRACTS
```

DECRYPT_BIN y DECRYPT_CHAR



El esquema es SYSIBM.

Tanto la función DECRYPT_BIN como la función DECRYPT_CHAR devuelven un valor obtenido tras descifrar *datos-cifrados*. La contraseña que se utiliza para descifrar es el valor *expresión-serie-contraseña* o el valor de la contraseña de cifrado asignada por la sentencia SET ENCRYPTION PASSWORD. Para mantener el mejor nivel de seguridad en el sistema, se recomienda que no pase la contraseña de cifrado explícitamente con las funciones DECRYPT_BIN y DECRYPT_CHAR en la consulta; en su lugar, utilice la sentencia SET ENCRYPTION PASSWORD para establecer la contraseña, y utilice una variable del lenguaje principal o marcadores de parámetro dinámicos cuando utilice la sentencia SET ENCRYPTION PASSWORD, en lugar de una serie literal.

Las funciones DECRYPT_BIN y DECRYPT_CHAR sólo pueden descifrar valores que se han cifrado mediante la función ENCRYPT (SQLSTATE 428FE).

datos-cifrados
Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA como una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT.

expresión-serie-contraseña
Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). Esta expresión tiene que ser la misma contraseña que se utiliza para cifrar los datos (SQLSTATE 428FD). Si no se ha proporcionado un valor para el argumento de la contraseña o bien dicho valor es nulo, los datos se cifrarán utilizando el valor de la contraseña de cifrado asignado para la sesión mediante la sentencia SET ENCRYPTION PASSWORD (SQLSTATE 51039).

El resultado de la función DECRYPT_BIN es VARCHAR FOR BIT DATA. El resultado de la función DECRYPT_CHAR es VARCHAR. Si *datos-cifrados* incluía una sugerencia, la función no la devolverá. El atributo de longitud del resultado corresponde a la longitud del tipo de datos de *datos-cifrados* menos 8 bytes. La longitud real del valor devuelto por la función coincidirá con la longitud de la serie original que se ha cifrado. Si *datos-cifrados* incluye bytes más allá de la serie cifrada, la función no los devolverá.

Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

Si los datos se descifran en un sistema diferente que utiliza una página de códigos que es diferente de la página de códigos en la que se cifraron los datos, puede producirse una expansión al convertir el valor descifrado a la página de códigos de la base de datos. En dichas situaciones, el valor *datos-cifrados* debe calcularse en una serie VARCHAR con un número mayor de bytes.

Ejemplos

El ejemplo siguiente demuestra el uso de la función DECRYPT_CHAR mostrando los fragmentos de código de una aplicación de SQL incorporado.

```
EXEC SQL BEGIN DECLARE SECTION;
      char hostVarCreateTableStmt[100];
      char hostVarSetEncPassStmt[200];
      char hostVarPassword[128];
      char hostVarInsertStmt1[200];
      char hostVarInsertStmt2[200];
      char hostVarSelectStmt1[200];
      char hostVarSelectStmt2[200];
EXEC SQL END DECLARE SECTION;
/* preparar la sentencia */
strcpy(hostVarCreateTableStmt, "CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA)");
EXEC SQL PREPARE hostVarCreateTableStmt FROM :hostVarCreateTableStmt;

/* ejecutar la sentencia */
EXEC SQL EXECUTE hostVarCreateTableStmt;
```

Utilizar la sentencia SET ENCRYPTION PASSWORD para establecer una contraseña de cifrado para la sesión:

```
/* 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 = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;

/* preparar la sentencia */
strcpy(hostVarInsertStmt1, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832')");
EXEC SQL PREPARE hostVarInsertStmt1 FROM :hostVarInsertStmt1;

/* ejecutar la sentencia */
EXEC SQL EXECUTE hostVarInsertStmt1;

/* preparar la sentencia */
strcpy(hostVarSelectStmt1, "SELECT DECRYPT_CHAR(SSN) FROM EMP");
EXEC SQL PREPARE hostVarSelectStmt1 FROM :hostVarSelectStmt1;

/* ejecutar la sentencia */
EXEC SQL EXECUTE hostVarSelectStmt1;
```

Esta consulta devuelve el valor '289-46-8832'.

Pasar la contraseña de cifrado explícitamente:

```
/* preparar la sentencia */
strcpy(hostVarInsertStmt2, "INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832',?)");
EXEC SQL PREPARE hostVarInsertStmt2 FROM :hostVarInsertStmt2;

/* ejecutar la sentencia correspondiente a hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarInsertStmt2 USING :hostVarPassword;

/* preparar la sentencia */
strcpy(hostVarSelectStmt2, "SELECT DECRYPT_CHAR(SSN,?) FROM EMP");
EXEC SQL PREPARE hostVarSelectStmt2 FROM :hostVarSelectStmt2;

/* ejecutar la sentencia correspondiente a hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSelectStmt2 USING :hostVarPassword;
```

Esta consulta devuelve el valor '289-46-8832'.

DEGREES

El esquema es SYSIBM. (La versión SYSFUN de la función DEGREES continúa estando disponible.)

La función DEGREES devuelve el número de grados del argumento, el cual es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Supongamos que RAD es una variable del lenguaje principal DECIMAL(4,3) con un valor de 3,142.

  ```
  VALUES DEGREES(:RAD)
  ```

  Devuelve el valor aproximado 180.0.

DEREF

La función DEREF devuelve una instancia del tipo de destino del argumento.

El argumento puede ser cualquier valor con un tipo de datos de referencia que tenga un ámbito definido (SQLSTATE 428DT).

El tipo de datos estático del resultado es el tipo de destino del argumento. El tipo de datos dinámico del resultado es un subtipo del tipo de destino del argumento. El resultado puede ser nulo. El resultado es un valor nulo si *expresión* es un valor nulo o si *expresión* es una referencia que no tiene un OID correspondiente en la tabla de destino.

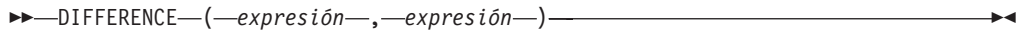
El resultado es una instancia del subtipo del tipo de destino de la referencia. El resultado se determina buscando la fila de la tabla de destino o vista de destino de la referencia que tenga un identificador de objeto que se corresponda con el valor de la referencia. El tipo de esta fila determina el tipo dinámico del resultado. Puesto que el tipo del resultado puede estar basado en una fila de una subtabla o subvista de la tabla de destino o vista de destino, el ID de autorización de la sentencia debe tener un privilegio SELECT sobre la tabla de destino y todas sus subtablas o sobre la vista de destino y todas sus subvistas (SQLSTATE 42501).

Ejemplos:

Supongamos que EMPLOYEE es una tabla de tipo EMP, y que su columna de identificador de objeto se llama EMPID. En este caso, la consulta siguiente devuelve un objeto de tipo EMP (o uno de sus subtipos) para cada fila de la tabla EMPLOYEE (y de sus subtablas). Para ejecutar esta consulta es necesario tener privilegio SELECT sobre EMPLOYEE y todas sus subtablas.

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

DIFFERENCE



El esquema es SYSFUN.

Devuelve un valor de 0 a 4 que representa la diferencia entre los sonidos de dos series basándose en la aplicación de la función SOUNDEX en las series. El valor 4 es la mejor coincidencia de sonido posible.

Los argumentos pueden ser series de caracteres que sean CHAR o VARCHAR y que no superen los 4.000 bytes. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. La función interpreta los datos que se le pasan como si se tratase de caracteres ASCII, aunque la codificación sea UTF-8.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

```
VALUES (DIFFERENCE('CONSTRAINT','CONSTANT'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONSTANT')),
(DIFFERENCE('CONSTRAINT','CONTRITE'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONTRITE'))
```

Este ejemplo devuelve lo siguiente.

```
1           2    3
----------- ---- ----
          4 C523 C523
          2 C523 C536
```

En la primera fila, las palabras tienen el mismo resultado de SOUNDEX, mientras que en la segunda fila las palabras sólo tienen algún parecido.

DIGITS

El esquema es SYSIBM.

La función DIGITS devuelve una representación de serie de caracteres de un número.

La expresión debe devolver un valor que sea un tipo de datos SMALLINT, INTEGER, BIGINT, DECIMAL, CHAR o VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a una serie de caracteres antes de que se ejecute la función. Los valores CHAR o VARCHAR se convierten implícitamente a DECIMAL(31,6) antes de evaluar la función.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

El resultado de la función es una serie de caracteres de longitud fija que representa el valor absoluto del argumento sin tener en cuenta su escala. El resultado no incluye el signo ni el carácter decimal. En su lugar, consta exclusivamente de dígitos, incluyendo, si es necesario, ceros iniciales para rellenar la serie. La longitud de la serie es:

- 5 si el argumento es un entero pequeño
- 10 si el argumento es un entero grande
- 19 si el argumento es un entero superior
- *p* si el argumento es un número decimal con una precisión de *p*.

Ejemplos:

- Supongamos que una tabla llamada TABLEX contiene una columna INTEGER llamada INTCOL que contiene números de 10 dígitos. Liste las cuatro combinaciones de dígitos de los cuatro primeros dígitos de la columna INTCOL.

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
    FROM TABLEX
  ```

- Supongamos que la columna COLUMNX tiene el tipo de datos DECIMAL(6,2) y que uno de sus valores es -6.28. Entonces, para este valor:

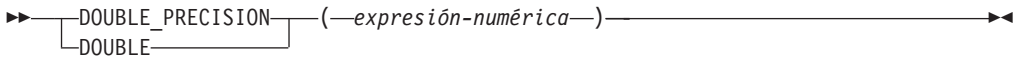
  ```
  DIGITS(COLUMNX)
  ```

  devuelve el valor '000628'.

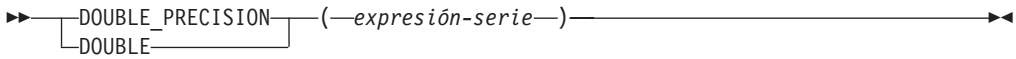
  El resultado es una serie de longitud seis (la precisión de la columna) con ceros iniciales que rellenan la serie hasta esta longitud. No aparecen ni el signo ni la coma decimal en el resultado.

DOUBLE_PRECISION o DOUBLE

De numérico a doble:



De serie de caracteres a doble:



El esquema es SYSIBM.

Las funciones DOUBLE_PRECISION y DOUBLE devuelven una representación de coma flotante de doble precisión de:

- Un número
- Una representación de serie de un número

De numérico a doble

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado es el mismo número que sería si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión doble. Si el valor numérico del argumento no está dentro del rango de la coma flotante de doble precisión, se devuelve un error (SQLSTATE 22003).

De serie de caracteres a doble

expresión-serie
Expresión que devuelve un valor que es una serie de caracteres o la representación de serie gráfica Unicode de un número. El tipo de datos de *expresión-serie* no debe ser un CLOB (SQLSTATE 42884).

El resultado es el mismo número que generaría CAST(*expresión-serie* AS DOUBLE PRECISION). Los espacios en blanco iniciales y finales se eliminan y la serie resultante debe ajustarse a las normas para formar una constante numérica válida (SQLSTATE 22018). Si el valor numérico del argumento no está dentro del rango de la coma flotante de doble precisión, se devuelve un error (SQLSTATE 22003).

El resultado de la función es un número de coma flotante de precisión doble. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas:

- La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".
- FLOAT es un sinónimo de DOUBLE_PRECISION y DOUBLE.
- La versión SYSFUN de DOUBLE (*expresión_serie*) sigue estando disponible.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y COMM) tienen tipos de datos DECIMAL. Para eliminar la posibilidad de resultados fuera de rango, se aplica DOUBLE a SALARY para que la división se lleve a cabo en coma flotante:

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

EMPTY_BLOB, EMPTY_CLOB y EMPTY_DBCLOB

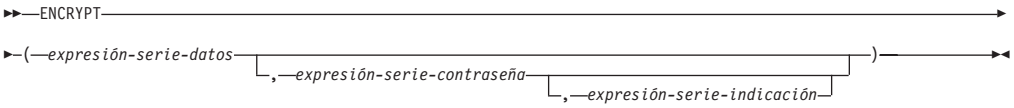
El esquema es SYSIBM.

Las funciones de valor vacío devuelven un valor de longitud cero del tipo de datos asociado. No hay argumentos para estas funciones (se deben especificar los paréntesis vacíos).

- La función EMPTY_BLOB devuelve un valor de longitud cero con un tipo de datos de BLOB(1).
- La función EMPTY_CLOB devuelve un valor de longitud cero con un tipo de datos de CLOB(1).
- La función EMPTY_DBCLOB devuelve un valor de longitud cero con un tipo de datos de DBCLOB(1).

El resultado de estas funciones se puede utilizar en asignaciones para proporcionar valores de longitud cero dónde sean necesarios.

ENCRYPT



El esquema es SYSIBM.

La función ENCRYPT devuelve un valor que es el resultado del cifrado de *expresión-serie-datos*. La contraseña que se utiliza para cifrar es el valor *expresión-serie-contraseña* o el valor de la contraseña de cifrado asignado por la sentencia SET ENCRYPTION PASSWORD. Para mantener el mejor nivel de seguridad en el sistema, se recomienda que no pase la contraseña de cifrado explícitamente con la función ENCRYPT en la consulta; en su lugar, utilice la sentencia SET ENCRYPTION PASSWORD para establecer la contraseña, y utilice una variable del lenguaje principal o marcadores de parámetro dinámicos cuando utilice la sentencia SET ENCRYPTION PASSWORD, en lugar de una serie literal.

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-serie-datos
Una expresión que devuelve el valor CHAR o VARCHAR que debe cifrarse. El atributo de longitud del tipo de datos de *expresión-serie-datos* está limitado a 32663 sin ningún argumento *expresión-serie-sugerencia* y a 32631 cuando se especifica el argumento *expresión-serie-sugerencia* (SQLSTATE 42815).

expresión-serie-contraseña
Una expresión que devuelve un valor CHAR o VARCHAR con un mínimo de 6 bytes y no más de 127 bytes (SQLSTATE 428FC). El valor representa la contraseña utilizada para cifrar *expresión-serie-datos*. Si el valor del argumento de contraseña es nulo o no se ha proporcionado, los datos se cifran con el valor de la contraseña de cifrado que se asignó para la sesión por la sentencia SET ENCRYPTION PASSWORD (SQLSTATE 51039).

expresión-serie-indicación
Una expresión que devuelve un valor CHAR o VARCHAR de un máximo de 32 bytes que ayudará a los propietarios de datos a recordar las contraseñas (por ejemplo, 'Océano' puede ser una sugerencia para recordar 'Pacífico'). Si se proporciona un valor de indicación, la indicación se incorpora en el resultado y puede recuperarse utilizando la función GETHINT. Si este argumento es nulo o no se proporciona, no se incorporará ninguna indicación en el resultado.

El tipo de datos de resultado de la función es VARCHAR FOR BIT DATA.

- Cuando se especifica el parámetro de sugerencia opcional, el atributo de longitud del resultado es igual al atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes + 32 bytes para la longitud de la sugerencia.
- Si no se ha especificado el parámetro de sugerencia opcional, el atributo de longitud del resultado es igual al atributo de longitud de los datos no cifrados + 8 bytes + el número de bytes hasta el siguiente límite de 8 bytes.

Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

Tenga en cuenta que el resultado cifrado tiene una longitud mayor que la del valor *expresión-serie-datos*. Por consiguiente, al asignar valores cifrados, asegúrese de que el destino se declara con un tamaño suficiente para contener el valor cifrado entero.

Notas

- ***Algoritmo de cifrado:*** El algoritmo de cifrado interno es la cifra de bloque RC2 con relleno; la clave secreta de 128 bits se deriva de la contraseña utilizando un resumen de mensaje MD5.
- ***Contraseñas y datos de cifrado:*** la gestión de contraseñas es responsabilidad del usuario. Una vez que se han cifrado los datos, sólo se puede utilizar para descifrarlos la contraseña utilizada para cifrarlos (SQLSTATE 428FD).

  El resultado cifrado puede contener el terminador nulo y otros caracteres no imprimibles. Cualquier asignación o conversión a una longitud inferior a la longitud de datos sugerida puede producir un descifrado anómalo en el futuro y hacer que se pierdan datos. Los espacios en blanco son valores de datos cifrados válidos que se pueden truncar al almacenarse en una columna demasiado pequeña.
- ***Administración de datos cifrados:*** los datos cifrados sólo se pueden descifrar en servidores que soporten las funciones de descifrado que corresponden a la función ENCRYP. Por lo tanto, la duplicación de columnas con datos cifrados sólo se debe realizar en servidores que soporten la función DECRYPT_BIN o DECRYPT_CHAR.

Ejemplos

El ejemplo siguiente demuestra el uso de la función ENCRYPT mostrando los fragmentos de código de una aplicación de SQL incorporado.

```
EXEC SQL BEGIN DECLARE SECTION;
      char hostVarCreateTableStmt[100];
      char hostVarSetEncPassStmt[200];
      char hostVarPassword[128];
      char hostVarInsertStmt1[200];
      char hostVarInsertStmt2[200];
      char hostVarInsertStmt3[200];
EXEC SQL END DECLARE SECTION;
/* preparar la sentencia */
strcpy(hostVarCreateTableStmt, "CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA)");
EXEC SQL PREPARE hostVarCreateTableStmt FROM :hostVarCreateTableStmt;

/* ejecutar la sentencia */
EXEC SQL EXECUTE hostVarCreateTableStmt;
```

Utilizar la sentencia SET ENCRYPTION PASSWORD para establecer una contraseña de cifrado para la sesión:

```
/* 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 = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarSetEncPassStmt USING :hostVarPassword;

/* preparar la sentencia */
strcpy(hostVarInsertStmt1, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832')");
EXEC SQL PREPARE hostVarInsertStmt1 FROM :hostVarInsertStmt1;

/* ejecutar la sentencia */
EXEC SQL EXECUTE hostVarInsertStmt1;
```

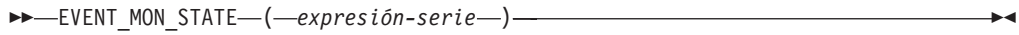
Pasar la contraseña de cifrado explícitamente:

```
/* preparar la sentencia */
strcpy(hostVarInsertStmt2, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832',?)");
EXEC SQL PREPARE hostVarInsertStmt2 FROM :hostVarInsertStmt2;

/* ejecutar la sentencia correspondiente a hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarInsertStmt2 USING :hostVarPassword;
```

Definir una sugerencia de contraseña:

```
/* preparar la sentencia */
strcpy(hostVarInsertStmt3, "INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832',?,'Ocean')");
EXEC SQL PREPARE hostVarInsertStmt3 FROM :hostVarInsertStmt3;

/* ejecutar la sentencia correspondiente a hostVarPassword = 'Pac1f1c' */
strcpy(hostVarPassword, "Pac1f1c");
EXEC SQL EXECUTE hostVarInsertStmt3 USING :hostVarPassword;
```

EVENT_MON_STATE



El esquema es SYSIBM.

La función EVENT_MON_STATE devuelve el estado actual de un supervisor de sucesos.

El argumento es una expresión de serie con un tipo resultante de CHAR o VARCHAR y un valor que es el nombre de un supervisor de sucesos. Si el supervisor de sucesos nombrado no existe en la tabla del catálogo SYSCAT.EVENTMONITORS, se devolverá SQLSTATE 42704. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado es un entero con uno de los valores siguientes:

0 El supervisor de sucesos está inactivo.

1 El supervisor de sucesos está activo.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

El siguiente ejemplo selecciona todos los supervisores de sucesos definidos e indica si cada uno está activo o inactivo:

```
SELECT EVMONNAME,
  CASE
    WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
    WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'
  END
  FROM SYSCAT.EVENTMONITORS
```

EXP

El esquema es SYSIBM. (La versión SYSFUN de la función EXP continúa estando disponible.)

La función EXP devuelve un valor que es la base del logaritmo natural (e) elevada a la potencia especificada por el argumento. Las funciones EXP y LN son operaciones opuestas.

El argumento debe ser una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas

Resultados que implican valores especiales de DECFLOAT: para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:

- EXP(NaN) devuelve NaN.
- EXP(-NaN) devuelve -NaN.
- EXP(Infinity) devuelve Infinity.
- EXP(-Infinity) devuelve 0.
- EXP(sNaN) devuelve NaN y un aviso.
- EXP(-sNaN) devuelve -NaN y un aviso.

Ejemplo

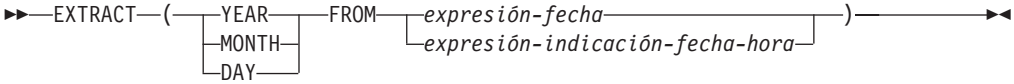
- Supongamos que E es una variable del lenguaje principal DECIMAL(10,9) con un valor de 3,453789832.

  ```
  VALUES EXP(:E)
  ```

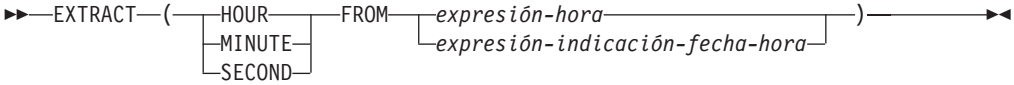
  Devuelve el valor DOUBLE +3.16200000069145E+001.

EXTRACT

Extraer valores de fecha



Extraer valores de hora



El esquema es SYSIBM.

La función EXTRACT devuelve una parte de una fecha o indicación de fecha y hora según sus argumentos.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Extraer valores de fecha

YEAR
Especifica que se devuelve la parte correspondiente al año de *expresión-fecha* o *expresión-indicación-fecha-hora*. El resultado es idéntico a la función escalar YEAR.

MONTH
Especifica que se devuelve la parte correspondiente al mes de *expresión-fecha* o *expresión-indicación-fecha-hora*. El resultado es idéntico a la función escalar MONTH.

DAY
Especifica que se devuelve la parte correspondiente al día de *expresión-fecha* o *expresión-indicación-fecha-hora*. El resultado es idéntico a la función escalar DAY.

expresión-fecha
Una expresión que devuelve el valor de un tipo de datos DATE incorporado o de un tipo de datos de serie de caracteres incorporada.

Si *expresión-fecha* es una serie de caracteres o una serie gráfica, debe ser una representación de serie válida de una fecha que no sea CLOB. En una base de datos Unicode, si una *expresión-fecha* es una serie gráfica, se convertirá en una serie de caracteres antes de que se ejecute la función.

expresión-indicación-fecha-hora
Una expresión que devuelve el valor de un tipo de datos TIMESTAMP incorporado o de un tipo de datos de serie de caracteres incorporada.

Si *expresión-indicación-fecha-hora* es una serie de caracteres o una serie gráfica, debe ser una representación de serie válida de una indicación de fecha y hora que no sea CLOB. En una base de datos Unicode, si una *expresión-indicación-fecha-hora* es una serie gráfica, se convertirá en una serie de caracteres antes de que se ejecute la función.

Extraer valores de hora

HOUR
Especifica que se devuelve la parte correspondiente a la hora de *expresión-hora* o *expresión-indicación-fecha-hora*. El resultado es idéntico a la función escalar HOUR.

MINUTE
Especifica que se devuelve la parte correspondiente al minuto de *expresión-hora* o *expresión-indicación-fecha-hora*. El resultado es idéntico a la función escalar MINUTE.

SECOND
Especifica que se devuelve la parte correspondiente al segundo de *expresión-hora* o *expresión-indicación-fecha-hora*. El resultado es idéntico a:

- SECOND(*expresión*, 6) cuando el tipo de datos de *expresión* es un valor de TIME o una representación en forma de serie de TIME o TIMESTAMP
- SECOND(*expresión*, *s*) cuando el tipo de datos de *expresión* es un valor de TIMESTAMP(*s*)

expresión-hora
Una expresión que devuelve el valor de un tipo de datos TIME incorporado o de un tipo de datos de serie de caracteres incorporada.

Si *expresión-hora* es una serie de caracteres o una serie gráfica, debe ser una representación de serie válida de una hora que no sea CLOB. En una base de datos Unicode, si una *expresión-hora* es una serie gráfica, se convertirá en una serie de caracteres antes de que se ejecute la función.

expresión-indicación-fecha-hora
Una expresión que devuelve el valor de un tipo de datos DATE, TIMESTAMP incorporado o de un tipo de datos de serie de caracteres incorporada.

Si *expresión-indicación-fecha-hora* es un valor DATE, éste se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00).

Si *expresión-indicación-fecha-hora* es una serie de caracteres o una serie gráfica, debe ser una representación de serie válida de una indicación de fecha y hora o de fecha que no sea CLOB. En una base de datos Unicode, si una *expresión-indicación-fecha-hora* es una serie gráfica, se convertirá en una serie de caracteres antes de que se ejecute la función. La serie se convierte en un valor TIMESTAMP(6).

El tipo de datos del resultado de la función depende de la parte del valor de fecha y hora que se especifique:

- Si se especifica YEAR, MONTH, DAY, HOUR o MINUTE, el tipo de datos del resultado es INTEGER.
- Si se especifica SECOND con un valor TIMESTAMP(*p*), el tipo de datos del resultado es DECIMAL(2+*p*, *p*), donde *p* es la precisión en segundos fraccionarios.
- Si se especifica SECOND con un valor de TIME o una representación de serie de TIME o TIMESTAMP, el tipo de datos del resultado es DECIMAL(8,6).

Ejemplo

- Supongamos que la columna PRSTDATE tiene un valor interno equivalente a 1988-12-25:

```
SELECT EXTRACT(MONTH FROM PRSTDATE)
   FROM PROJECT;
```

FLOAT

El esquema es SYSIBM.

La función FLOAT devuelve una representación de coma flotante de un número. FLOAT es sinónimo de DOUBLE.

FLOOR

El esquema es SYSIBM. (La versión SYSFUN de la función FLOOR continúa estando disponible).

Devuelve el valor del entero más grande que es menor o igual que el argumento.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento, con la excepción de que la escala es 0 si el argumento es DECIMAL. Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,5) devuelve DECIMAL(5,0).

El resultado puede ser nulo si el argumento puede ser nulo o si el argumento no es un número de coma flotante decimal y la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

Notas

Resultados que implican valores especiales de DECFLOAT: para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:

- FLOOR(NaN) devuelve NaN.
- FLOOR(-NaN) devuelve -NaN.
- FLOOR(Infinity) devuelve Infinity.
- FLOOR(-Infinity) devuelve -Infinity.
- FLOOR(sNaN) devuelve NaN y un aviso.
- FLOOR(-sNaN) devuelve -NaN y un aviso.

Ejemplos

- Utilice la función FLOOR para truncar los dígitos que haya a la derecha de la coma decimal.

```
SELECT FLOOR(SALARY)
  FROM EMPLOYEE
```

- Utilice la función FLOOR en los números positivos y negativos.

```
   VALUES FLOOR(3.5),
FLOOR(3.1),
     FLOOR(-3.1), FLOOR(-3.5)
```

  Este ejemplo devuelve 3., 3., -4. y -4., respectivamente.

GENERATE_UNIQUE

El esquema es SYSIBM.

La función GENERATE_UNIQUE devuelve una serie de caracteres de datos de bits de 13 bytes de longitud (CHAR(13) FOR BIT DATA) que es exclusiva comparada con cualquier otra ejecución de la misma función. (El reloj del sistema se utiliza para generar la indicación de fecha y hora UTC (Hora universal coordinada) interna junto con el número de la partición de base de datos en la que se ejecuta la función. Los ajustes que retrasan el reloj del sistema real podrían generar valores duplicados). La función está definida como no determinista.

No hay ningún argumento para esta función (se han de especificar los paréntesis vacíos).

El resultado de la función es un valor exclusivo que incluye el formato interno de la hora UTC y el número de la partición de base de datos en la que se ha procesado la función. El resultado no puede ser nulo.

El resultado de esta función se puede utilizar para proporcionar valores exclusivos en una tabla. Cada valor sucesivo será mayor que el valor anterior, proporcionando una secuencia que se puede utilizar en una tabla. El valor incluye el número de la partición de base de datos en la que se ejecuta la función de modo que una tabla particionada en varias particiones de bases de datos también tiene valores exclusivos en alguna secuencia. La secuencia se basa en la hora en que se ha ejecutado la función.

Esta función difiere de la utilización del registro especial CURRENT TIMESTAMP en que se genera un valor exclusivo para cada fila de una sentencia de inserción de múltiples filas o en una sentencia de inserción con una selección completa.

El valor de indicación de fecha y hora que forma parte del resultado de esta función puede determinarse utilizando la función escalar TIMESTAMP con el resultado de GENERATE_UNIQUE como argumento.

Ejemplos:

- Cree una tabla que incluya una columna que sea exclusiva para cada fila. Llene esta columna utilizando la función GENERATE_UNIQUE. Tenga en cuenta que la columna UNIQUE_ID tiene especificado ″FOR BIT DATA″ para identificar la columna como una serie de caracteres de datos de bits.

  ```
  CREATE TABLE EMP_UPDATE
    (UNIQUE_ID CHAR(13) FOR BIT DATA,
    EMPNO CHAR(6),
    TEXT VARCHAR(1000))
    INSERT INTO EMP_UPDATE
      VALUES (GENERATE_UNIQUE(), '000020', 'Update entry...'),
      (GENERATE_UNIQUE(), '000050', 'Update entry...')
  ```

  Esta tabla tendrá un identificador exclusivo para cada fila siempre que la columna UNIQUE_ID se establezca siempre utilizando GENERATE_UNIQUE. Esto se puede realizar introduciendo un activador en la tabla.

```
CREATE TRIGGER EMP_UPDATE_UNIQUE
  NO CASCADE BEFORE INSERT ON EMP_UPDATE
  REFERENCING NEW AS NEW_UPD
  FOR EACH ROW
  SNEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

Con este activador definido, la sentencia INSERT anterior se emitiría sin la primera columna, tal como se indica a continuación.

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
  VALUES ('000020', 'Update entry 1...'),
  ('000050', 'Update entry 2...')
```

Puede devolverse la indicación de fecha y hora (en UTC) para el momento en que se ha añadido una fila a EMP_UPDATE utilizando:

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT
  FROM EMP_UPDATE
```

Por lo tanto, no hay necesidad de tener una columna de indicación de fecha y hora en la tabla para registrar el momento en que se ha insertado una fila.

GETHINT

El esquema es SYSIBM.

La función GETHINT devolverá la indicación de contraseña si se encuentra alguna en *datos-cifrados*. Una indicación de contraseña es una expresión que ayuda a los propietarios de datos a recordar las contraseñas; por ejemplo, 'Océano' como indicación para recordar 'Pacífico'. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

datos-cifrados
Una expresión que devuelve un valor CHAR FOR BIT DATA o VARCHAR FOR BIT DATA que es una serie de datos cifrada completa. La serie de datos se tiene que haber cifrado utilizando la función ENCRYPT (SQLSTATE 428FE).

El resultado de la función es VARCHAR(32). El resultado puede ser nulo; si la función ENCRYPT no ha añadido el parámetro de indicación a los *datos-cifrados* o el primer argumento es nulo, el resultado será el valor nulo.

Ejemplo:

En este ejemplo se almacena la indicación 'Océano' para ayudar al usuario a recordar la contraseña de cifrado 'Pacífico'.

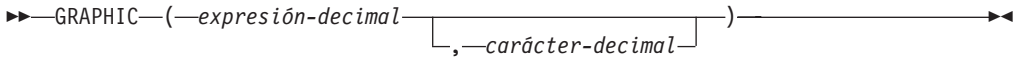
```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Pacífico','Océano');
SELECT GETHINT(SSN)
  FROM EMP;
```

El valor devuelto es 'Océano'.

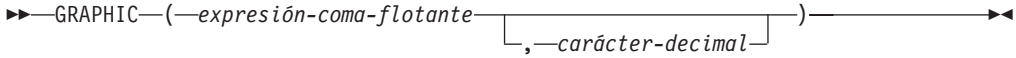
GRAPHIC

De entero a gráfico:

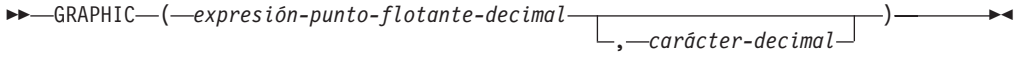
De decimal a gráfico:



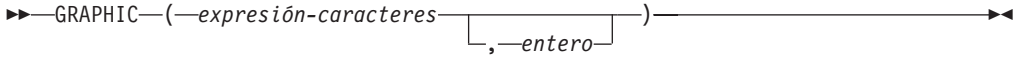
De coma flotante a gráfico:



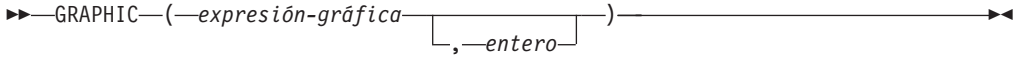
De coma flotante decimal a gráfico:



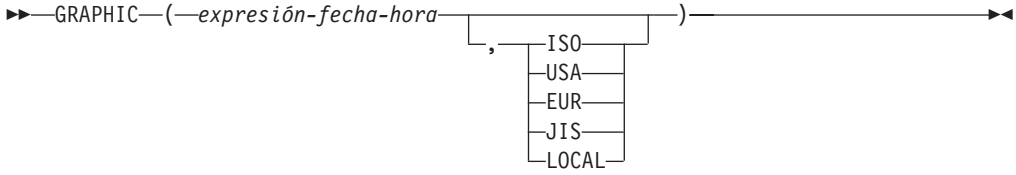
De carácter a gráfico:



De gráfico a gráfico:



De fecha y hora a gráfico:



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función.

La función GRAPHIC devuelve una representación de serie gráfica de longitud fija de:

- Un número entero (sólo base de datos Unicode), si el primer argumento es SMALLINT, INTEGER o BIGINT
- Un número decimal (sólo base de datos Unicode), si el primer argumento es un número decimal

- Un número de coma flotante de precisión doble (sólo bases de datos Unicode), si el primer argumento es DOUBLE o REAL
- Un número de coma flotante decimal (sólo base de datos Unicode), si el argumento es un número de coma flotante decimal (DECFLOAT)
- Una serie de caracteres, convirtiendo los caracteres de un solo byte en caracteres de doble byte, si el primer argumento es cualquier tipo de serie de caracteres
- Una serie gráfica, si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora (sólo para bases de datos Unicode), si el primer argumento es un valor DATE, TIME o TIMESTAMP

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter se convierte en un carácter en blanco (X'0020'). No confíe en este comportamiento, porque podría cambiar en los releases futuros.

El resultado de la función es una serie gráfica de longitud fija. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De entero a gráfico

expresión-entero
Una expresión que devuelve un valor que es un tipo de datos de entero (SMALLINT, INTEGER o BIGINT).

El resultado es una representación de serie gráfica de longitud fija de la *expresión-entero* con el formato de una constante de entero SQL. El resultado consta de *n* caracteres de doble byte, que representan los dígitos significativos del argumento, precedido por un signo menos si el argumento es negativo. El resultado está justificado por la izquierda.

- Si el primer argumento es un entero pequeño, la longitud del resultado es de 6.
- Si el primer argumento es un entero grande, la longitud del resultado es de 11.
- Si el primer argumento es un entero superior, la longitud del resultado es de 20.

Si el número de caracteres de doble byte del resultado es inferior a la longitud definida del resultado, el resultado se rellena a la derecha con espacios en blanco.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De decimal a gráfico

expresión-decimal
Una expresión que devuelve un valor que es de un tipo de datos decimal. La función escalar DECIMAL puede utilizarse para cambiar la precisión y la escala.

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo

más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud fija de la *expresión-decimal* con el formato de una constante decimal SQL. La longitud del resultado es 2+*p*, donde *p* es la precisión de la *expresión-decimal*. Los ceros iniciales no se incluyen. Los ceros finales se incluyen. Si *expresión-decimal* es negativa, el primer carácter de doble byte del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si la escala de *expresión-decimal* es cero, el carácter decimal no se devuelve. Si el número de caracteres de doble byte del resultado es inferior a la longitud definida del resultado, el resultado se rellena a la derecha con espacios en blanco.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De coma flotante a gráfico

expresión-coma-flotante
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud fija de la *expresión-coma-flotante* con el formato de una constante de coma flotante SQL. La longitud del resultado es 24. El resultado es el número menor de caracteres de doble byte que puedan representar el valor de la *expresión-coma-flotante*, de manera que la mantisa conste de un solo dígito que no sea cero seguido de un punto y una secuencia de dígitos. Si *expresión-coma-flotante* es negativa, el primer carácter de doble byte del resultado es un signo menos; si no, el primer carácter de doble byte es un dígito. Si *expresión-coma-flotante* es cero, el resultado es 0E0. Si el número de caracteres de doble byte del resultado es inferior a 24, el resultado se rellena a la derecha con espacios en blanco.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De coma flotante decimal a gráfico

expresión-coma-flotante-decimal
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante decimal (DECFLOAT).

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud fija de la *expresión-coma-flotante-decimal* con el formato de una constante de coma

flotante decimal SQL. El atributo de longitud del resultado es 42. El resultado es el número menor de caracteres que pueda representar el valor de *expresión-coma-flotante-decimal*. Si *expresión-coma-flotante-decimal* es negativa, el primer carácter de doble byte del resultado es un signo menos; si no, el primer carácter de doble byte es un dígito. Si *expresión-coma-flotante-decimal* es cero, el resultado es 0.

Si el valor de *expresión-coma-flotante-decimal* es el valor especial Infinity, sNaN o NaN, se devuelven las series G'INFINITY', G'SNAN' y G'NAN', respectivamente. Si el valor especial es negativo, el primer carácter de doble byte del resultado es un signo menos. El valor especial de coma flotante decimal sNaN no genera un aviso cuando se convierte en una serie. Si el número de caracteres de doble byte del resultado es inferior a 42, el resultado se rellena a la derecha con espacios en blanco.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De carácter a gráfico

expresión-caracteres
Una expresión que devuelve un valor que es una serie de caracteres incorporados (CHAR, VARCHAR o CLOB).

entero
El atributo de longitud de la serie gráfica de longitud fija resultante. El valor debe estar entre 0 y 127. Si el segundo argumento no se especifica:

- Si la *expresión-caracteres* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento. Si la longitud real del primer argumento (incluidos los espacios en blanco finales) es superior a 127, se devuelve un error (SQLSTATE 22001).

La longitud real del resultado es la misma que el atributo de longitud del resultado. Si la longitud de la *expresión-caracteres* es menor que el atributo de longitud del resultado, el resultado se rellena con espacios en blanco hasta llenar toda la longitud del resultado. Si la longitud de la *expresión-caracteres* es superior al atributo de longitud del resultado, éste se trunca sin que se devuelva ningún aviso.

De gráfico a gráfico

expresión-gráfica
Una expresión que devuelve un valor incorporado que es de un tipo de datos de serie gráfica (GRAPHIC, VARGRAPHIC o DBCLOB).

entero
El atributo de longitud de la serie gráfica de longitud fija resultante. El valor debe estar entre 0 y 127.

Si el segundo argumento no se especifica:

- Si la expresión-gráfica es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento. Si la longitud real del primer argumento (sin contar los espacios en blanco finales) es superior a 127, se devuelve un error (SQLSTATE 22001).

La longitud real del resultado es la misma que el atributo de longitud del resultado. Si la longitud de la *expresión-gráfica* es menor que el atributo de longitud del resultado, el resultado se rellena con espacios en blanco hasta llenar toda la longitud del resultado. Si la longitud de la *expresión-gráfica* es superior al atributo de longitud del resultado, éste se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y la *expresión-gráfica* no sea DBCLOB.

De fecha y hora a gráfico

expresión-fecha-hora
Una expresión que sea uno de los tipos de datos siguientes:

DATE El resultado es la representación de serie gráfica de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIME El resultado es la representación de serie gráfica de la hora en el formato especificado por el segundo argumento. La longitud del resultado es 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIMESTAMP
El resultado es la representación de serie gráfica de la indicación de fecha y hora. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(0), la longitud del resultado es 19. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(*n*), donde *n* es un número entre 1 y 12, la longitud del resultado es 20+*n*. De lo contrario, la longitud del resultado es 26.

La página de códigos del resultado es la página de códigos de la sección.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones cuando el primer argumento es numérico o si el primer argumento es una serie y se especifica el argumento de longitud. Para obtener más información, consulte la "especificación CAST".

Ejemplos

- La columna EDLEVEL está definida como SMALLINT. Lo siguiente devuelve el valor como serie gráfica de longitud fija.

```
SELECT GRAPHIC(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

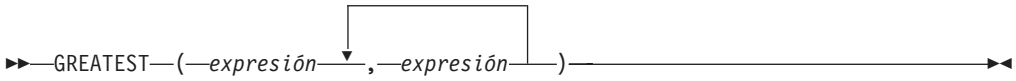
  Genera el valor G'18 '.

- Las columnas SALARY y COMM se definen como DECIMAL con una precisión de 9 y una escala de 2. Se devuelven los ingresos totales del empleado Haas con el carácter de coma decimal.

```
SELECT GRAPHIC(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  Genera el valor G'56970,00 '.

GREATEST



El esquema es SYSIBM.

La función GREATEST devuelve el valor máximo de un conjunto de valores.

Los argumentos deben ser compatibles, y cada argumento debe ser una expresión que devuelva un valor de cualquier tipo de datos distinto de ARRAY, LOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42815). Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Dado que esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para soportar tipos diferenciados definidos por el usuario.

El argumento seleccionado se convierte, si es necesario, a los atributos del resultado. Los atributos del resultado los determinan todos los operandos basándose en las normas de los tipos de datos de resultado.

El resultado de la función es el valor de argumento más grande. Si como mínimo un argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

La función escalar GREATEST es sinónimo de la función escalar MAX.

Ejemplos:

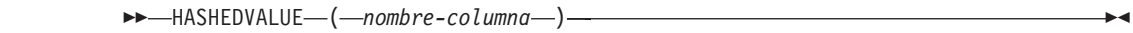
Suponga que la tabla T1 contiene tres columnas C1, C2 y C3 con los valores 1, 7 y 4, respectivamente. La consulta:

```
SELECT GREATEST (C1, C2, C3) FROM T1
```

devuelve 7.

Si la columna C3 tiene un valor de NULL en lugar de 4, la misma consulta devuelve NULL.

HASHEDVALUE



El esquema es SYSIBM.

La función HASHEDVALUE devuelve el índice de la correlación de distribución de la fila que se obtiene al aplicar la función de particionamiento en el valor de la clave de distribución de la fila. Por ejemplo, si se utiliza en una cláusula SELECT, devuelve el índice de correlación de distribución de cada fila de la tabla que se ha utilizado para formar el resultado de la sentencia SELECT.

El índice de la correlación de distribución devuelta en las variables y las tablas de transición se deriva de los valores de transición actuales de las columnas de claves de distribución. Por ejemplo, en un activador BEFORE INSERT, la función devolverá el índice de la correlación de distribución proyectada que corresponda a los valores actuales de las variables de transición nuevas. No obstante, es posible que los valores de las columnas de claves de distribución se modifiquen mediante un activador BEFORE INSERT subsiguiente. Por lo tanto, el índice de la correlación de distribución de la fila cuando se inserta en la base de datos puede ser distinto del valor proyectado.

El argumento debe ser el nombre calificado o no calificado de una columna de una tabla. La columna puede tener cualquier tipo de datos. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos diferenciados definidos por el usuario). Si *nombre-columna* hace referencia a una columna de una vista, la expresión de la vista para la columna debe hacer referencia a una columna de la tabla base principal y la vista debe ser suprimible. Una expresión de tabla anidada o común sigue las mismas normas que una vista.

La fila (y la tabla) específica para la que la función HASHEDVALUE devuelve el índice de la correlación de distribución se determina a partir del contexto de la sentencia de SQL que utiliza la función.

El tipo de datos del resultado es INTEGER en el rango de 0 a 32767. Para una tabla que no tiene clave de distribución, el resultado siempre es 0. Nunca se devuelve un valor nulo. Puesto que se devuelve información a nivel de fila, los resultados son los mismos, sin tener en cuenta las columnas que se especifican para la tabla.

La función HASHEDVALUE no puede utilizarse en tablas duplicadas, dentro de restricciones de comprobación ni en la definición de columnas generadas (SQLSTATE 42881).

Para compatibilidad con versiones anteriores a la Versión 8, el nombre de función PARTITION puede sustituirse por HASHEDVALUE.

Ejemplo:

- Lista de números de empleados (EMPNO) de la tabla EMPLOYEE para todas las filas cuyo índice de correlación de distribución es 100.

```
SELECT EMPNO FROM EMPLOYEE
  WHERE HASHEDVALUE(PHONENO) = 100
```

- Registrar el número de empleado y el índice de la correlación de distribución proyectada de la nueva fila en una tabla denominada EMPINSERTLOG2 para cualquier inserción de empleados creando un activador BEFORE en la tabla EMPLOYEE.

```
CREATE TRIGGER EMPINSLOGTRIG2
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AW NEWTABLE
  FOR EACH ROW
  INSERT INTO EMPINSERTLOG2
    VALUES(NEWTABLE.EMPNO, HASHEDVALUE(NEWTABLE.EMPNO))
```

HEX

El esquema es SYSIBM.

La función HEX devuelve una representación hexadecimal de un valor como una serie de caracteres.

El argumento puede ser una expresión que sea un valor de cualquier tipo de datos incorporados con una longitud máxima de 16.336 bytes.

El resultado de la función es una serie de caracteres. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La página de códigos es la página de códigos de la sección.

El resultado es una serie de dígitos hexadecimales. Los dos primeros bytes representan el primer byte del argumento, los dos siguientes el segundo byte del argumento, etcétera. Si el argumento es un valor de indicación de fecha y hora o un valor numérico el resultado es la representación hexadecimal del formato interno del argumento. La representación hexadecimal que se devuelve puede ser diferente según el servidor de aplicaciones donde se ejecuta la función. Los casos en que las diferencias pueden ser evidentes son:

- Los argumentos de serie de caracteres cuando se ejecuta la función HEX en un cliente ASCII con un servidor EBCDIC o en un cliente EBCDIC con un servidor ASCII.
- Los argumentos numéricos (en algunos casos) cuando se ejecuta la función HEX donde los sistemas cliente y servidor tienen distintas clasificaciones de bytes para los valores numéricos.

El tipo y la longitud del resultado varían basándose en el tipo y la longitud de los argumentos de serie de caracteres.

- Serie de caracteres
  - Longitud fija no mayor que 127
    - El resultado es una serie de caracteres de longitud fija el doble de la longitud definida del argumento.
  - Longitud fija mayor que 127
    - El resultado es una serie de caracteres de longitud variable el doble de la longitud definida del argumento.
  - Longitud variable
    - El resultado es una serie de caracteres de longitud variable con una longitud máxima el doble de la longitud máxima definida del argumento.
- Serie gráfica
  - Longitud fija no mayor que 63
    - El resultado es una serie de caracteres de longitud fija cuatro veces la longitud definida del argumento.
- Longitud fija mayor que 63
  - El resultado es una serie de caracteres de longitud variable cuatro veces la longitud definida del argumento.

- Longitud variable
  - El resultado es una serie de caracteres de longitud variable con una longitud máxima cuatro veces la longitud máxima definida del argumento.

Ejemplos:

Supongamos que utiliza un servidor de aplicaciones DB2 para AIX para los ejemplos siguientes.

- Utilizando la tabla DEPARTMENT establezca la variable del lenguaje principal HEX_MGRNO (char(12)) en la representación hexadecimal del número del director (MGRNO) para el departamento 'PLANNING' (DEPTNAME).

```
SELECT HEX(MGRNO)
  INTO :HEX_MGRNO
  FROM DEPARTMENT
  WHERE DEPTNAME = 'PLANNING'
```

  HEX_MGRNO se establecerá en '303030303230' cuando se utilice la tabla de ejemplo (el valor de caracteres es '000020').
- Supongamos que COL_1 es una columna con un tipo de datos de char(1) y un valor de 'B'. La representación hexadecimal de la letra 'B' es X'42'. HEX(COL_1) devuelve una serie de dos bytes '42'.
- Supongamos que COL_3 es una columna con un tipo de datos de decimal(6,2) y un valor de 40,1. Una serie de ocho bytes '0004010C' es el resultado de aplicar la función HEX a la representación interna del valor decimal 40,1.

HOUR

El esquema es SYSIBM.

La función HOUR devuelve la parte correspondiente a la hora de un valor.

Una expresión que devuelve un valor de uno de los siguientes tipos de datos incorporados: un valor DATE, TIME, TIMESTAMP, una serie de caracteres o un tipo de datos numérico exacto.

- Si *expresión* es una serie de caracteres, no debe ser CLOB ni DBCLOB y su valor debe ser una representación de serie válida de un valor de fecha y hora. Para conocer los formatos válidos de las representaciones de serie de los valores de fecha y hora, consulte "Representación mediante series de los valores de fecha y hora" en "Valores de fecha y hora".
- Si *expresión* es un valor numérico exacto, debe ser una duración de tiempo o una duración de indicación de fecha y hora. Para obtener información sobre duraciones válidas de tiempo y de indicación de fecha y hora, consulte "Operandos y duración de fecha y hora".
- Sólo las bases de datos Unicode dan soporte a una expresión que es una representación de serie gráfica válida de un valor de fecha y hora que no sea DBCLOB. La serie gráfica se convierte en una serie de caracteres antes de ejecutar la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

- Si el argumento es un valor TIME, TIMESTAMP o una representación de serie válida de una hora o indicación de fecha y hora:
  - El resultado es la parte correspondiente a la hora del valor, que es un entero entre 0 y 24.
- Si el argumento es un valor DATE o una representación de serie válida de una fecha:
  - El resultado es 0.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente a la hora del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo

Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases que empiezan por la tarde.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

El esquema es SYSIBM.

La función IDENTITY_VAL_LOCAL es una función no determinista que devuelve el valor asignado más recientemente para una columna de identidad, donde la asignación se ha producido como resultado de una sentencia INSERT individual utilizando una cláusula VALUES. La función no tiene parámetros de entrada.

El resultado es un DECIMAL(31,0), independientemente del tipo de datos real de la columna de identidad correspondiente.

El valor devuelto por la función es el valor asignado a la columna de identidad de la tabla identificada en la operación de inserción de fila individual más reciente. La sentencia INSERT debe contener una cláusula VALUES en una tabla que contenga una columna de identidad. La sentencia INSERT también debe ejecutarse al mismo nivel; es decir, el valor debe estar disponible localmente en el nivel al que se ha asignado, hasta que se sustituya por el siguiente valor asignado. (Se inicia un nivel nuevo cada vez que se invoca un activador o una rutina).

El valor asignado es un valor proporcionado por el usuario (si la columna de identidad está definida como GENERATED BY DEFAULT) o un valor de identidad generado por el gestor de bases de datos.

Se recomienda utilizar una sentencia SELECT FROM referencia-tabla-cambio-datos para obtener el valor asignado para una columna de identidad. Consulte ″referencia-tabla″ en el apartado sobre ″subselección″ para obtener más información.

La función devuelve un valor nulo si no se ha emitido una sentencia INSERT de fila individual con una cláusula VALUES en el nivel de proceso actual para una tabla que contiene una columna de identidad.

El resultado de la función no queda afectado por lo siguiente:

- Una sentencia INSERT de fila individual con una cláusula VALUES para una tabla sin columna de identidad
- Una sentencia INSERT de múltiples filas con una cláusula VALUES
- Una sentencia INSERT con una selección completa
- Una sentencia ROLLBACK TO SAVEPOINT

Notas

- Las expresiones de la cláusula VALUES de una sentencia INSERT se evalúan antes que las asignaciones para las columnas de destino de la operación de inserción. Por consiguiente, una invocación de una función IDENTITY_VAL_LOCAL en la cláusula VALUES de una sentencia INSERT utilizará el valor asignado más recientemente de una columna de identidad de una operación de inserción anterior. La función devuelve el valor nulo si no se ha ejecutado ninguna sentencia INSERT de fila individual anterior con una cláusula VALUES para una tabla que contiene una columna de identidad dentro del mismo nivel que la función IDENTITY_VAL_LOCAL.

- El valor de la columna de identidad de la tabla para la que se define el activador puede determinarse dentro de un activador, haciendo referencia a la variable de transición activador para la columna de identidad.
- El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la condición activador de un activador de inserción es un valor nulo.
- Es posible que existan múltiples activadores de inserción anteriores o posteriores para una tabla. En este caso, cada activador se procesa por separado y los valores de identidad asignados por una acción activada no están disponibles para las demás acciones activadas utilizando la función IDENTITY_VAL_LOCAL. Esto es válido incluso aunque las múltiples acciones activadas estén definidas conceptualmente al mismo nivel.
- Generalmente no es recomendable utilizar la función IDENTITY_VAL_LOCAL en el cuerpo de un activador anterior (before) de inserción. El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde dentro de la acción activada de un activador de inserción anterior es el valor nulo. El valor de la columna de identidad de la tabla para la que se ha definido el activador no se puede obtener invocando la función IDENTITY_VAL_LOCAL en la acción activada de un activador de inserción anterior. Sin embargo, el valor de la columna de identidad puede obtenerse en la acción activada, haciendo referencia a la variable de transición activador para la columna de identidad.
- El resultado de la invocación de la función IDENTITY_VAL_LOCAL desde la acción activada de un activador de inserción posterior (after) es el valor asignado a una columna de identidad de la tabla identificada en la operación de inserción de fila individual más reciente invocada en la misma acción activada que tenía una cláusula VALUES para una tabla que contenía una columna de identidad. (Esto se aplica a los activadores posteriores (after) de inserción FOR EACH ROW y FOR EACH STATEMENT). Si una sentencia INSERT de una sola fila con una cláusula VALUES para la tabla que contiene una columna de identidad no se ha ejecutado dentro de la misma acción activada antes de la invocación de la función IDENTITY_VAL_LOCAL, la función devolverá un valor nulo.
- Dado que la función IDENTITY_VAL_LOCAL no es determinista, el resultado de invocar esta función dentro de la sentencia SELECT de un cursor puede variar para cada sentencia FETCH.
- El valor asignado es el valor realmente asignado a la columna de identidad (es decir, el valor que se devolverá en una sentencia SELECT subsiguiente). Este valor no es necesariamente el valor proporcionado en la cláusula VALUES de la sentencia INSERT o un valor generado por el gestor de bases de datos. El valor asignado puede ser un valor especificado en una sentencia de variable de transición SET, dentro del cuerpo de un activador de inserción anterior, para una variable de transición activador asociada con la columna de identidad.
- ***Ámbito de IDENTITY_VAL_LOCAL***: El valor de IDENTITY_VAL_LOCAL se mantiene hasta que se produce durante la sesión actual la siguiente inserción en una tabla con una columna de identidad definida allí, o hasta que finaliza la sesión de la aplicación. El valor no se ve afectado por las sentencias COMMIT o ROLLBACK. El valor de IDENTITY_VAL_LOCAL no se puede definir directamente y es el resultado de insertar una fila 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 direccione transacciones a una conexión arbitraria. En estas situaciones, la disponibilidad del valor de IDENTITY_VAL_LOCAL solamente es válida hasta el final de la transacción. Algunos ejemplos de dónde puede producirse este tipo de situación incluyen aplicaciones que utilicen protocolos XA, usen la

agrupación de conexiones, empleen el concentrador de conexiones y utilicen HADR para lograr la migración tras error.

- El valor devuelto por la función es imprevisible después de una sentencia INSERT de fila individual anómala con una cláusula VALUES en una tabla con una columna de identidad. Puede ser el valor que la función habría devuelto si ésta se hubiera invocado antes de la operación de inserción anómala o bien el valor que se hubiese asignado si la operación de inserción hubiera sido satisfactoria. El valor real devuelto depende del punto de anomalía y, por consiguiente, es imprevisible.

Ejemplos:

Ejemplo 1: Crear dos tablas, T1 y T2, cada una con una columna de identidad llamada C1. Iniciar la secuencia de identificación para la tabla T2 en 10. Insertar algunos valores para C2 en T1.

```
CREATE TABLE T1
  (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
   C2 INTEGER)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY (START WITH 10),
   C2 INTEGER)

INSERT INTO T1 (C2) VALUES (5)

INSERT INTO T1 (C2) VALUES (6)

SELECT * FROM T1
```

Esta consulta devuelve:

```
C1                   C2
-------------------- --------------------
                   1                    5
                   2                    6
```

Insertar una única fila en la tabla T2, donde la columna C2 obtiene un valor de la función IDENTITY_VAL_LOCAL.

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL())

SELECT * FROM T2
```

Esta consulta devuelve:

```
C1                   C2
-------------------- --------------------
                  10.                    2
```

Ejemplo 2: En un entorno anidado que incluya un activador, utilice la función IDENTITY_VAL_LOCAL para recuperar el valor de identidad asignado en un nivel determinado, incluso aunque puedan haber valores de identidad asignados en niveles inferiores. Supongamos que existen tres tablas, EMPLOYEE, EMP_ACT y ACCT_LOG. Hay un activador de inserción posterior definido en EMPLOYEE que produce inserciones adicionales en las tablas EMP_ACT y ACCT_LOG.

```
CREATE TABLE EMPLOYEE
  (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
   NAME    CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT)

CREATE TABLE EMP_ACT
```

```
    (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
     EMPNO SMALLINT)

  CREATE TABLE ACCT_LOG
    (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
     ACNT_NUM SMALLINT,
     EMPNO SMALLINT)

  CREATE TRIGGER NEW_HIRE
    AFTER INSERT ON EMPLOYEE
    REFERENCING NEW AS NEW_EMP
    FOR EACH ROW
    BEGIN ATOMIC
      INSERT INTO EMP_ACT (EMPNO) VALUES (NEW_EMP.EMPNO);
      INSERT INTO ACCT_LOG (ACNT_NUM, EMPNO)
        VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
    END
```

La primera operación de inserción activada inserta una fila en la tabla EMP_ACT. La sentencia utiliza una variable de transición activadora para la columna EMPNO de la tabla EMPLOYEE para indicar que el valor de identidad de la columna EMPNO de la tabla EMPLOYEE debe copiarse en la columna EMPNO de la tabla EMP_ACT. La función IDENTITY_VAL_LOCAL no se ha podido utilizar para obtener el valor asignado a la columna EMPNO de la tabla EMPLOYEE, ya que no se ha emitido una sentencia INSERT en este nivel de anidamiento. Si se hubiera invocado la función IDENTITY_VAL_LOCAL en la cláusula VALUES de la sentencia INSERT para la tabla EMP_ACT, se hubiera devuelto un valor nulo. La operación de inserción en la tabla EMP_ACT también hace que se genere un nuevo valor de identidad para la columna ACNT_NUM.

La segunda operación de inserción activada inserta una fila en la tabla ACCT_LOG. La sentencia invoca la función IDENTITY_VAL_LOCAL para indicar que el valor de identidad asignado a la columna ACNT_NUM de la tabla EMP_ACT en la operación de inserción anterior de la acción activada debe copiarse en la columna ACNT_NUM de la tabla ACCT_LOG. A la columna EMPNO se le asigna el mismo valor que a la columna EMPNO de la tabla EMPLOYEE.

Después de que se hayan procesado la sentencia INSERT siguiente y todas las acciones activadas:

```
  INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
    VALUES ('Rupert', 989.99, 50)
```

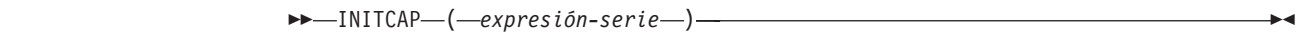
el contenido de las tres tablas será el siguiente:

```
  SELECT EMPNO, SUBSTR(NAME,1,10) AS NAME, SALARY, DEPTNO
    FROM EMPLOYEE

EMPNO  NAME       SALARY  DEPTNO
------ ---------- ------- ------
  1000 Rupert      989.99     50

  SELECT ACNT_NUM, EMPNO
    FROM EMP_ACT

ACNT_NUM EMPNO
-------- ------
       1   1000

  SELECT * FROM ACCT_LOG
```

```
ID     ACNT_NUM EMPNO
------ -------- ------
   100        1   1000
```

El resultado de la función IDENTITY_VAL_LOCAL es el valor asignado más recientemente para una columna de identidad en el mismo nivel de anidamiento. Después de procesar la sentencia INSERT original y todas las acciones activadas, la función IDENTITY_VAL_LOCAL devuelve un valor de 1000, porque éste es el valor asignado a la columna EMPNO de la tabla EMPLOYEE.

INITCAP



El esquema es SYSIBM.

La función INITCAP devuelve una serie con el primer carácter de cada *palabra* convertido a mayúscula (con la semántica de la función UPPER) y el resto de caracteres convertido a minúsculas (con la semántica de la función LOWER). Una *palabra* está delimitada por cualquiera de los caracteres siguientes:

Tabla 46. Caracteres delimitadores de palabras

Carácter o rango de caracteres	**Elementos de código Unicode o rango de elementos de código Unicode**
(blanco)	U+0020
! " # $ % & ' ( ) * + , - . /	Desde U+0021 hasta U+002F
: ; < = > ? @	Desde U+003A hasta U+0040
[\ ] ^ _ `	Desde U+005B hasta U+0060
{ | } ~	Desde U+007B hasta U+007E
Caracteres de control, incluidos los caracteres de control de SQL siguientes: • tabulador • línea nueva • salto de página • retorno de carro • salto de línea	U+0009, U+000A, U+000B, U+000C, U+000D, U+0085

Nota: Los caracteres enumerados en la tabla anterior no pueden tener un elemento de código asignado en una página de códigos de la base de datos en particular.

expresión-serie
Expresión que devuelve un valor que es un tipo de datos CHAR o VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función.

El tipo de datos del resultado depende del tipo de datos de la *expresión-serie*, como se describe en la tabla siguiente:

Tabla 47. Tipo de datos de la expresión-serie comparado con el tipo de datos del resultado

Tipo de datos de *expresión-serie*	**Tipo de datos del resultado**
CHAR o VARCHAR	VARCHAR
GRAPHIC o VARGRAPHIC	VARGRAPHIC

El atributo de longitud del resultado es el mismo que el atributo de longitud de la *expresión-serie*.

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Especificar la serie “título libro consulta” para devolver la serie “Título Libro Consulta”.

```
VALUES INITCAP ('título libro consulta')
1
------------------------
Título Libro Consulta
```

- Especificar la serie “SU NOMBRE” para devolver la serie “Su Nombre”.

```
VALUES INITCAP ('SU NOMBRE')
1
---------
Su Nombre
```

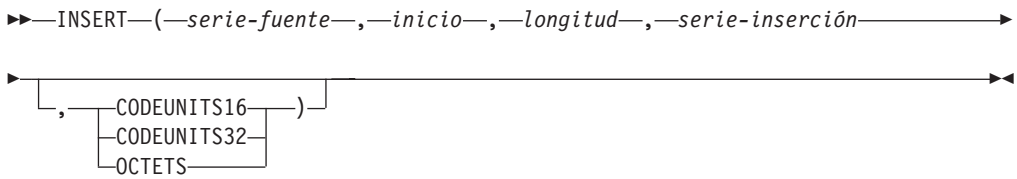
- Especificar la serie “mi_curriculum” para devolver la serie “Mi_Curriculum”.

```
VALUES INITCAP ('mi_curriculum')
1
---------
Mi_Curriculum
```

- Especificar la serie “élégant” para devolver la serie “Élégant”.

```
VALUES INITCAP ('FORMAT:élégant')
1
--------------
Format:Élégant
```

INSERT



El esquema es SYSIBM. La versión SYSFUN de la función INSERT continúa estando disponible.

La función INSERT devuelve una serie donde, a partir de *inicio* en *serie-fuente*, se han seleccionado *longitud* bytes y se ha insertado *serie-inserción*.

La función INSERT es idéntica a la función OVERLAY, excepto en que el argumento de longitud es obligatorio.

serie-fuente
Expresión que especifica la serie fuente. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función.

inicio
Expresión que devuelve un valor entero. El valor entero especifica el punto de partida en la serie fuente donde debe empezar la supresión de bytes y la inserción de otra serie. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe estar comprendido entre 1 y la longitud de la *serie-fuente* más uno (SQLSTATE 42815). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número impar entre 1 y el doble del atributo de longitud de *serie-fuente* más uno (SQLSTATE 428GC).

longitud
Expresión que especifica el número de unidades de código (en las unidades de serie especificadas) que se deben suprimir de la serie fuente, a partir de la posición identificada por *inicio*. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe estar en el rango comprendido entre 0 y la longitud de la *serie-fuente*, expresado en unidades que se especifican implícita o explícitamente (SQLSTATE 22011). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número par entre 0 y el doble del atributo de longitud de *serie-fuente* (SQLSTATE 428GC).

serie-inserción
Expresión que especifica la serie que se debe insertar en *serie-fuente*, a partir de la posición identificada por *inicio*. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función.

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie de *inicio* y *longitud*.

CODEUNITS16 especifica que *inicio* y *longitud* se expresan en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *inicio* y *longitud* se expresan en unidades de código UTF-32 de 32 bits. OCTETS especifica que *inicio* y *longitud* se expresan en bytes.

Si la unidad de serie se especifica como CODEUNITS16 o CODEUNITS32 y el resultado es una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si la unidad de serie se especifica como OCTETS y *serie-inserción* y *serie-fuente* son series binarias, se devuelve un error (SQLSTATE 42815). Si la unidad de serie se especifica como OCTETS, la operación se realiza en la página de códigos de la *serie-fuente*. Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *inicio* y *longitud* se expresan en unidades de dos bytes; de lo contrario, se expresan en bytes. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El tipo de datos del resultado depende de los tipos de datos de *serie-fuente* y *serie-inserción*, como se muestra en la tabla siguiente de combinaciones de tipos soportadas.

Tabla 48. Tipo de datos del resultado como función de los tipos de datos de serie-fuente y serie-inserción

serie-fuente	*serie-inserción*	**Resultado**
CHAR o VARCHAR	CHAR o VARCHAR	VARCHAR
GRAPHIC o VARGRAPHIC	GRAPHIC o VARGRAPHIC	VARGRAPHIC
CLOB	CHAR, VARCHAR o CLOB	CLOB
DBCLOB	GRAPHIC, VARGRAPHIC o DBCLOB	DBCLOB
CHAR o VARCHAR	CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	CHAR, VARCHAR, CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
BLOB	BLOB	BLOB
Sólo para bases de datos Unicode:		
CHAR o VARCHAR	GRAPHIC o VARGRAPHIC	VARCHAR
GRAPHIC o VARGRAPHIC	CHAR o VARCHAR	VARGRAPHIC
CLOB	GRAPHIC, VARGRAPHIC o DBCLOB	CLOB
DBCLOB	CHAR, VARCHAR o CLOB	DBCLOB

Una *serie-fuente* puede tener una longitud de 0; en este caso, *inicio* debe ser 1 y *longitud* debe ser 0 (como lo implican los vínculos para *inicio* y *longitud* descritos más arriba) y el resultado de la función es una copia de la *serie-inserción*.

Una *serie-inserción* también puede tener una longitud de 0. Esto tiene el efecto de suprimir las unidades de código de las posiciones *inicio* a *inicio* + *longitud* - 1 de la *serie-fuente*.

El atributo de longitud del resultado es el atributo de longitud de *serie-fuente* más el atributo de longitud de *serie-inserción*. La longitud real del resultado es A1 - MIN((A1 - V2 + 1), V3) + A4, donde:

- A1 es la longitud real de *serie-fuente*
- V2 es el valor de *inicio*
- V3 es el valor de *longitud*
- A4 es la longitud real de *serie-inserción*

Si la longitud real de la serie de resultado excede el máximo del tipo de datos de retorno, se devuelve un error (SQLSTATE 54006).

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Cree las series 'INSISTING', 'INSISERTING' e 'INSTING' a partir de la serie 'INSERTING' insertando texto en medio del texto existente.

```
SELECT INSERT('INSERTING',4,2,'IS'),
  INSERT('INSERTING',4,0,'IS'),
  INSERT('INSERTING',4,2,'')
  FROM SYSIBM.SYSDUMMY1
```

- Cree las series 'XXINSERTING', 'XXNSERTING', 'XXSERTING' y 'XXERTING' a partir de la serie 'INSERTING' insertando texto antes del texto existente, utilizando 1 como punto de partida.

```
SELECT INSERT('INSERTING',1,0,'XX'),
  INSERT('INSERTING',1,1,'XX'),
  INSERT('INSERTING',1,2,'XX'),
  INSERT('INSERTING',1,3,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- Cree la serie 'ABCABCXX' a partir de la serie 'ABCABC' insertando texto después del texto existente. Dado que la serie fuente tiene una longitud de 6 caracteres, establezca la posición inicial en 7 (uno más la longitud de la serie fuente).

```
SELECT INSERT('ABCABC',7,0,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- Cambie la serie 'Hegelstraße' por 'Hegelstrasse'.

```
SELECT INSERT('Hegelstraße',10,1,'ss',CODEUNITS16)
  FROM SYSIBM.SYSDUMMY1
```

- El ejemplo siguiente funciona con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	**'N'**	**'~'**	**'A'**	**'B'**
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  Supongamos que las variables UTF8_VAR y UTF16_VAR contienen las representaciones de la serie en UTF-8 y UTF-16BE respectivamente. Utilice la función INSERT para insertar una 'C' en la serie Unicode '&N~AB'.

```
SELECT INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'CAB', 'CB' y 'CN~AB', respectivamente.

```
SELECT INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 5, 1, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '&N~CB', '&N~AC' y '&C~AB', respectivamente.

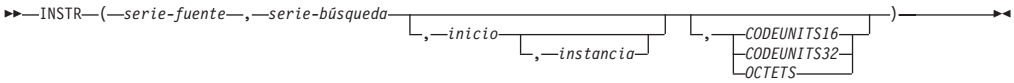
```
SELECT INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'CAB', 'CB' y 'CN~AB', respectivamente.

```
SELECT INSERT(UTF16_VAR, 5, 2, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 5, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '&N~C', '&N~AC' y '&CAB', respectivamente.

INSTR



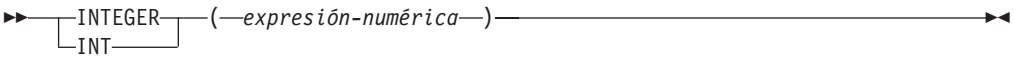
El esquema es SYSIBM

La función INSTR devuelve la posición inicial de una serie (denominada *serie-búsqueda*) dentro de otra serie (denominada *serie-fuente*).

La función escalar INSTR es sinónimo de la función escalar LOCATE_IN_STRING.

INTEGER o INT

De numérico a entero:



De serie a entero:

De fecha a entero:

De hora a entero:

El esquema es SYSIBM.

La función INTEGER devuelve una representación de entero de:

- Un número
- Una representación de serie de un número
- Un valor de fecha
- Un valor de hora

De numérico a entero:

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado es el mismo número que el que se generaría si se asignara el argumento a una variable o columna de enteros grandes. La parte fraccional del argumento se trunca. Si la parte completa del argumento no está dentro del rango de los enteros grandes, se devuelve un error (SQLSTATE 22003).

De serie a entero:

expresión-serie
Una expresión que devuelve un valor de serie de caracteres o la representación de serie gráfica Unicode de un número con una longitud no superior a la longitud máxima de una constante de caracteres.

El resultado es el mismo número que generaría CAST (*expresión-serie*AS INTEGER). Los espacios en blanco iniciales y finales se eliminan y la serie resultante debe ajustarse a las normas para formar una constante de entero, decimal, de coma flotante o coma flotante decimal (SQLSTATE 22018). Si la parte completa del argumento no está dentro del rango de los enteros grandes,

se devuelve un error (SQLSTATE 22003). El tipo de datos de *expresión-serie* no debe ser CLOB o DBCLOB (SQLSTATE 42884).

De fecha a entero:

expresión-fecha
Una expresión que devuelve un valor del tipo de datos DATE. El resultado es un valor INTEGER que representa la fecha como *aaaammdd*.

De hora a entero:

expresión-hora
Una expresión que devuelve un valor del tipo de datos TIME. El resultado es un valor INTEGER que representa la hora como *hhmmss*.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".

Ejemplos

- Utilizando la tabla EMPLOYEE, seleccione una lista que contenga el salario (SALARY) dividido por el nivel de formación (EDLEVEL). Trunque cualquier decimal en el cálculo. La lista también debe contener los valores utilizados en el cálculo y el número de empleado (EMPNO). La lista debe estar en orden descendente del valor calculado.

```
SELECT INTEGER (SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
  ORDER BY 1 DESC
```

- Utilizando la tabla EMPLOYEE, seleccione la columna EMPNO en el formato de enteros para procesarla más en la aplicación.

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

- Supongamos que la columna BIRTHDATE (cuyo tipo de datos es DATE) tiene un valor interno equivalente a '1964-07-20'.

```
INTEGER(BIRTHDATE)
```

  da como resultado el valor `19 640 720`.

- Supongamos que la columna STARTTIME (cuyo tipo de datos es TIME) tiene un valor interno equivalente a '12:03:04'.

```
INTEGER(STARTTIME)
```

  da como resultado el valor `120 304`.

JULIAN_DAY

El esquema es SYSFUN.

Devuelve un valor entero que representa el número de días desde el 1 de enero de 4713 AC (la fecha de inicio del calendario Juliano) hasta el valor de fecha especificado en el argumento.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

LAST_DAY

El esquema es SYSIBM.

La función escalar LAST_DAY devuelve un valor de fecha y hora que representa el último día del mes del argumento.

expresión
Expresión que especifica la fecha inicial. La expresión debe devolver un valor de uno de los siguientes tipos de datos incorporados: valor DATE o valor TIMESTAMP.

El resultado de la función tiene el mismo tipo de datos que *expresión*, a menos que *expresión* sea una serie, en cuyo caso el tipo de datos del resultado es DATE. El resultado puede ser nulo; si el valor de *expresión-fecha* es nulo, el resultado es el valor nulo.

La función no modifica la información de horas, minutos, segundos o segundos fraccionados incluida en *expresión*.

Ejemplos

- Establecer la variable del lenguaje principal *END_OF_MONTH* con el último día del mes en curso.

  ```
  SET :END_OF_MONTH = LAST_DAY(CURRENT_DATE);
  ```

  La variable del lenguaje principal *END_OF_MONTH* se establece con el valor que representa el final del mes en curso. Si el día en curso es 2000-02-10, *END_OF_MONTH* se establece en 2000-02-29.
- Establecer la variable del lenguaje principal *END_OF_MONTH* con el último día del mes en formato EUR correspondiente a la fecha dada.

  ```
  SET :END_OF_MONTH = CHAR(LAST_DAY('1965-07-07'), EUR);
  ```

  La variable del lenguaje principal *END_OF_MONTH* se establece con el valor '31.07.1965'.

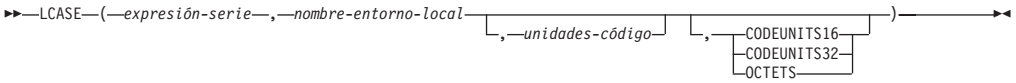
LCASE

El esquema es SYSIBM.

La función LCASE devuelve una serie en la que todos los caracteres SBCS se han convertido a minúsculas.

LCASE es sinónimo de LOWER.

LCASE (sensible al entorno local)

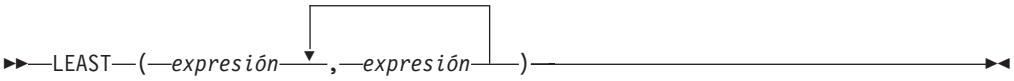


El esquema es SYSIBM.

La función LCASE devuelve una serie en la que todos los caracteres se han convertido a minúsculas utilizando las normas asociadas con el entorno local especificado.

LCASE es sinónimo de LOWER.

LEAST



El esquema es SYSIBM.

La función LEAST devuelve el valor mínimo de un conjunto de valores.

Los argumentos deben ser compatibles, y cada argumento debe ser una expresión que devuelva un valor de cualquier tipo de datos distinto de ARRAY, LOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42815). Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Dado que esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para soportar tipos diferenciados definidos por el usuario.

El argumento seleccionado se convierte, si es necesario, a los atributos del resultado. Los atributos del resultado los determinan todos los operandos basándose en las normas de los tipos de datos de resultado.

El resultado de la función es el valor de argumento más pequeño. Si como mínimo un argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

La función escalar LEAST es sinónimo de la función escalar MIN.

Ejemplos:

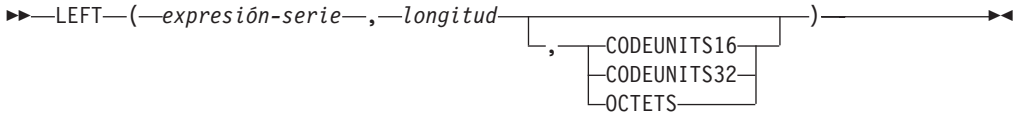
Suponga que la tabla T1 contiene tres columnas C1, C2 y C3 con los valores 1, 7 y 4, respectivamente. La consulta:

```
SELECT LEAST (C1, C2, C3) FROM T1
```

devuelve 1.

Si la columna C3 tiene un valor de NULL en lugar de 4, la misma consulta devuelve NULL.

LEFT



El esquema es SYSIBM. La versión SYSFUN de la función LEFT continúa estando disponible.

La función LEFT devuelve la serie situada más a la izquierda de *expresión-serie* de la longitud *longitud*, expresada en la unidad de serie especificada. Si *expresión-serie* es una serie de caracteres, el resultado es una serie de caracteres. Si *expresión-serie* es una serie gráfica, el resultado es una serie gráfica.

expresión-serie
Una expresión que especifica la serie de la que se deriva el resultado. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente en VARCHAR antes de evaluar la función. Una subserie de *expresión-serie* es cero o más elementos de código contiguos de *expresión-serie*.

longitud
Una expresión que especifica la longitud del resultado. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor debe estar entre 0 y la longitud de *expresión-serie*, expresado en unidades que se especifiquen implícita o explícitamente (SQLSTATE 22011). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número par entre 0 y el doble del atributo de longitud de *expresión-serie* (SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de serie de *longitud*.

CODEUNITS16 especifica que *longitud* se expresa en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *longitud* se expresa en unidades de código UTF-32 de 32 bits. OCTETS especifica que *longitud* se expresa en bytes.

Si la unidad de serie se especifica como CODEUNITS16 o CODEUNITS32 y *expresión-serie* es una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si la unidad de serie se especifica como OCTETS y *expresión-serie* es una serie gráfica, *longitud* debe ser un número par; de lo contrario, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *longitud* se expresa en unidades de dos bytes; de lo contrario, se expresa en bytes. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado “Unidades de serie en funciones incorporadas” en “Series de caracteres”.

La *expresión-serie* se rellena a la derecha con el número necesario de caracteres de relleno para que exista siempre la subserie especificada de *expresión-serie*. El carácter utilizado para el relleno es el mismo carácter que se utiliza para rellenar la

serie en contextos donde se debe producir relleno. Para obtener más información sobre el relleno, consulte “Asignaciones de serie” en “Asignaciones y comparaciones”.

El resultado de la función es una serie de longitud variable con un atributo de longitud que es el mismo atributo de longitud que el de *expresión-serie* y un tipo de datos que depende del tipo de datos de *expresión-serie*:

- VARCHAR si *expresión-serie* es CHAR o VARCHAR
- CLOB si *expresión-serie* es CLOB
- VARGRAPHIC si *expresión-serie* es GRAPHIC o VARGRAPHIC
- DBCLOB si *expresión-serie* es DBCLOB
- BLOB si *serie-expresión* es BLOB

La longitud real del resultado (en unidades de serie) es *longitud*.

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Suponga que la variable ALPHA tiene un valor de ’ABCDEF’. La siguiente sentencia:

  ```
  SELECT LEFT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  devuelve ’ABC’, que son los tres caracteres situados más a la izquierda en ALPHA.
- Suponga que la variable NAME, que se define como VARCHAR(50), tiene un valor de ’KATIE AUSTIN’ y que la variable de entero FIRSTNAME_LEN tiene un valor de 5. La siguiente sentencia:

  ```
  SELECT LEFT(NAME,FIRSTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  devuelve el valor ’KATIE’.
- La siguiente sentencia devuelve una serie de longitud cero.

  ```
  SELECT LEFT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```
- La columna FIRSTNME de la tabla EMPLOYEE se define como VARCHAR(12). Se busca el nombre de un empleado cuyo apellido es ’BROWN’ y se debe devolver el nombre en una serie de 10 bytes.

  ```
  SELECT LEFT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  devuelve una serie VARCHAR(12) que tiene el valor ’DAVID’ seguido de cinco caracteres en blanco.
- En una base de datos Unicode, FIRSTNAME es una columna VARCHAR(12). Uno de sus valores es la serie de 6 caracteres ’Jürgen’. Cuanto FIRSTNAME tiene este valor:

  ```
  Función...                      Devuelve...

  LEFT(FIRSTNAME,2,CODEUNITS32)   'Jü' -- x'4AC3BC'
  LEFT(FIRSTNAME,2,CODEUNITS16)   'Jü' -- x'4AC3BC'
  LEFT(FIRSTNAME,2,OCTETS)        'J'  -- x'4A20', una serie truncada
  ```

- El ejemplo siguiente funciona con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  Suponga que la variable UTF8_VAR, con un atributo de longitud de 20 bytes, contiene la representación UTF-8 de la serie.

```
SELECT LEFT(UTF8_VAR, 2, CODEUNITS16),
  LEFT(UTF8_VAR, 2, CODEUNITS32),
  LEFT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&', '&N' y 'bb', respectivamente, donde 'b' representa el carácter en blanco.

```
SELECT LEFT(UTF8_VAR, 5, CODEUNITS16),
  LEFT(UTF8_VAR, 5, CODEUNITS32),
  LEFT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&N~A', '&N~AB' y '&N', respectivamente.

```
SELECT LEFT(UTF8_VAR, 10, CODEUNITS16),
  LEFT(UTF8_VAR, 10, CODEUNITS32),
  LEFT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&N~ABbbbb', '&N~ABbbbbb' y '&N~ABb', respectivamente, donde 'b' representa el carácter en blanco.

  Suponga que la variable UTF16_VAR, con un atributo de longitud de 20 unidades de código, contiene la representación UTF-16BE de la serie.

```
SELECT LEFT(UTF16_VAR, 2, CODEUNITS16),
  LEFT(UTF16_VAR, 2, CODEUNITS32),
  HEX (LEFT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&', '&N' y X'D834', respectivamente, donde X'D834' es un carácter de sustitución no coincidente.

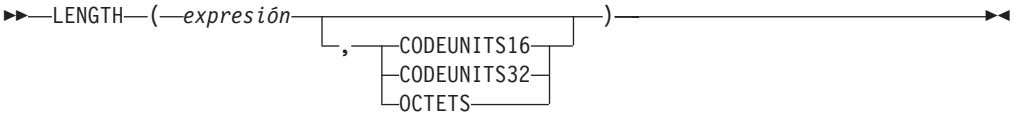
```
SELECT LEFT(UTF16_VAR, 5, CODEUNITS16),
  LEFT(UTF16_VAR, 5, CODEUNITS32),
  LEFT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&N~A', '&N~AB' y '&N', respectivamente.

```
SELECT LEFT(UTF16_VAR, 10, CODEUNITS16),
  LEFT(UTF16_VAR, 10, CODEUNITS32),
  LEFT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores '&N~ABbbbb', '&N~ABbbbbb' y '&N~A', respectivamente, donde 'b' representa el carácter en blanco.

LENGTH



El esquema es SYSIBM.

La función LENGTH devuelve la longitud de la *expresión* en la unidad de la serie implícita o explícita.

expresión
Expresión que devuelve un valor que es un tipo de datos incorporado. Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo.

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie del resultado. CODEUNITS16 especifica que el resultado debe expresarse en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que el resultado debe expresarse en unidades de código UTF-32 de 32 bits. OCTETS especifica que el resultado debe expresarse en bytes.

Si una unidad de la serie se especifica de forma explícita, y si la *expresión* no son datos de serie, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32 y la *expresión* es una serie binaria o datos de bits, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie se especifica como OCTETS y la *expresión* es una serie binaria, se devuelve un error (SQLSTATE 42815). Para obtener más información acerca de CODEUNITS16, CODEUNITS32 y OCTETS, consulte "Unidades de serie en las funciones incorporadas" en "Series de caracteres".

Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos de gráfico, el valor devuelto especifica la longitud en unidades de 2 bytes. En caso contrario, el valor devuelto especifica la longitud en bytes.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La longitud de las series de caracteres y de gráficos incluye espacios en blanco finales. La longitud de las series binarias incluye ceros binarios. La longitud de las series de longitud variable es la longitud real y no la longitud máxima. La longitud de todos los demás valores es el número de bytes utilizados para representar el valor:

- 2 para entero pequeño
- 4 para entero grande
- (p/2)+1 para números decimales con una precisión p
- 8 para DECFLOAT(16)
- 16 para DECFLOAT(34)
- La longitud de la serie para series binarias
- La longitud de la serie para series de caracteres
- 4 para coma flotante de precisión simple

- 8 para coma flotante de precisión doble
- 4 para DATE
- 3 para TIME
- 7+(*p*+1)/2 para TIMESTAMP(*p*)

Ejemplos

- Supongamos que la variable del lenguaje principal es una serie de caracteres de longitud variable cuyo valor es '895 Don Mills Road'.

```
LENGTH(:ADDRESS)
```

  devuelve el valor 18.
- Supongamos que START_DATE es una columna de tipo DATE.

```
LENGTH(START_DATE)
```

  devuelve el valor 4.
-

```
LENGTH(CHAR(START_DATE, EUR))
```

  devuelve el valor 10.
- Los ejemplos siguientes funcionan con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

  Supongamos que la variable UTF8_VAR contiene la representación UTF-8 de la serie.

```
SELECT LENGTH(UTF8_VAR, CODEUNITS16),
  LENGTH(UTF8_VAR, CODEUNITS32),
  LENGTH(UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores 6, 5 y 9, respectivamente.

  Supongamos que la variable UTF16_VAR contiene la representación UTF-16BE de la serie.

```
SELECT LENGTH(UTF16_VAR, CODEUNITS16),
   LENGTH(UTF16_VAR, CODEUNITS32),
   LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  devuelve los valores 6, 5 y 12, respectivamente.

LN

El esquema es SYSIBM. (La versión SYSFUN de la función LN continúa estando disponible.)

La función LN devuelve el logaritmo natural de un número. Las funciones EXP y LN son operaciones opuestas.

El argumento debe ser una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función. El valor del argumento debe ser superior a cero (SQLSTATE 22003).

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas

- ***Resultados que implican valores especiales de DECFLOAT:*** para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:
  - LN(NaN) devuelve NaN.
  - LN(-NaN) devuelve -NaN.
  - LN(Infinity) devuelve Infinity.
  - LN(-Infinity) devuelve NaN y un aviso.
  - LN(sNaN) devuelve NaN y un aviso.
  - LN(-sNaN) devuelve -NaN y un aviso.
  - LN(DECFLOAT('0') devuelve -Infinity.
- ***Alternativas de sintaxis:*** Se puede especificar LOG en lugar de LN. Sólo se soporta por compatibilidad con las versiones anteriores de productos DB2. LN debe utilizarse en vez de LOG, ya que algunas aplicaciones y gestores de base de datos implementan LOG como logaritmo común de un número en vez del logaritmo natural de un número.

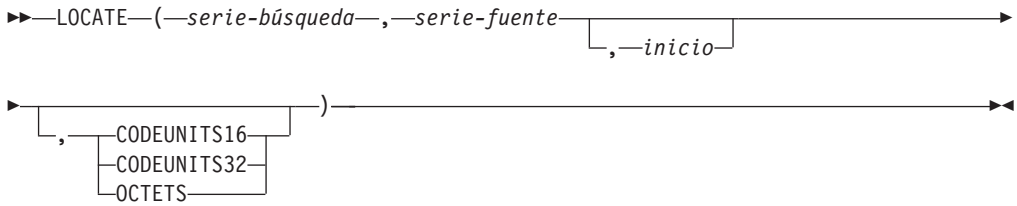
Ejemplo:

- Supongamos que NATLOG es una variable del lenguaje principal DECIMAL(4,2) con un valor de 31,62.

  ```
  VALUES LN(:NATLOG)
  ```

  Devuelve el valor aproximado 3,45.

LOCATE



El esquema es SYSIBM. La versión SYSFUN de la función LOCATE continúa estando disponible, pero no es sensible a la clasificación de base de datos.

La función LOCATE devuelve la posición inicial de la primera aparición de una serie (denominada *serie-búsqueda*) dentro de otra serie (denominada *serie-fuente*). Si el argumento *serie-búsqueda* no se encuentra y ningún argumento es nulo, el resultado es cero. Si el argumento *serie-búsqueda* se encuentra, el resultado es un número de 1 la longitud real del argumento *serie-fuente*. La búsqueda se realiza utilizando la clasificación de la base de datos, a menos que se defina *serie-búsqueda* o *serie-fuente* como FOR BIT DATA, en cuyo caso la búsqueda se realiza utilizando una comparación binaria.

Si se especifica el argumento opcional *inicio*, éste indica la posición del carácter en el argumento *serie-fuente* en el que debe iniciarse la búsqueda. Puede especificarse una unidad de la serie opcional para indicar en qué unidades se expresan el argumento *inicio* y el resultado de la función.

Si *serie-búsqueda* tiene una longitud de cero, el resultado que devuelve la función es 1. En cambio, si *serie-fuente* tiene una longitud de cero, el resultado que devuelve la función es 0. En caso contrario:

- Si el valor del argumento *serie-búsqueda* es igual a una longitud idéntica de subserie de posiciones contiguas dentro del valor de *serie-búsqueda*, el resultado que devuelve la función es la posición inicial de la subserie de este tipo dentro del valor *serie-fuente*.
- De lo contrario, el resultado que devuelve la función es 0.

serie-búsqueda

Expresión que especifica la serie que es objeto de la búsqueda. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, BLOB, VARGRAPHIC, GRAPHIC o CHAR incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC, VARGRAPHIC o BLOB, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión no puede ser una variable de referencia a archivo BLOB. La expresión puede especificarse mediante lo siguiente:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores
- Un parámetro de procedimiento de SQL

Estas normas son parecidas a las que se describen para *expresión-patrón* para el predicado LIKE.

serie-fuente
Expresión que especifica la serie en la que debe realizarse la búsqueda. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión puede especificarse mediante lo siguiente:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal (incluida una variable localizadora o una variable de referencia de archivo)
- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores

inicio
Expresión que especifica la posición entro de *serie-fuente* en la que debe iniciarse la búsqueda. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe ser superior o igual a cero. Si se especifica *inicio*, la función LOCATE es parecida a la siguiente:

```
POSITION(serie-búsqueda,
  SUBSTRING(serie-fuente, inicio, unidad-serie),
  unidad-serie) + inicio - 1
```

donde *unidad-serie* es CODEUNITS16, CODEUNITS32 u OCTETS.

Si no se especifica *inicio*, la búsqueda empieza en la primera posición de la serie fuente, y la función LOCATE es parecida a la siguiente:

```
POSITION(serie-búsqueda, serie-fuente, unidad-serie)
```

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie de *inicio* y del resultado. CODEUNITS16 especifica que *inicio* y el resultado deben expresarse en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *inicio* y el resultado deben expresarse en unidades de código UTF-32 de 32 bits. OCTETS especifica que *inicio* y el resultado deben expresarse en bytes.

Si una unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32, y *serie-búsqueda* o *serie-fuente* son una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si se especifica una unidad de la serie como OCTETS y *serie-búsqueda* y *serie-fuente* son series binarias, de devuelve un error (SQLSTATE 42815).

Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *inicio* y la posición devuelta se expresan en unidades de dos bytes; en caso contrario, se expresan en bytes.

Si se utiliza una clasificación basada en la UCA sensible a la configuración local para esta función, la opción CODEUNITS16 ofrece las mejores características de rendimiento.

Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El primer y segundo argumento deben tener tipos de serie compatibles. Para obtener más información sobre la compatibilidad, consulte el apartado "Normas para conversiones de series". En una base de datos Unicode, si un argumento de serie es de tipo carácter (no FOR BIT DATA) y el otro argumento de serie es gráfico, la *serie-búsqueda* se convierte al tipo de datos de la *serie-fuente* para el proceso. Si un argumento es de tipo carácter FOR BIT DATA, el otro argumento no debe ser gráfico (SQLSTATE 42846).

El resultado de la función es un entero grande. Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Buscar la ubicación de la primera aparición del carácter 'N' en la serie 'DINING'.

  ```
  SELECT LOCATE('N', 'DINING')
  FROM SYSIBM.SYSDUMMY1
  ```

  El resultado es el valor 3.

- Para todas las filas de la tabla denominada IN_TRAY, seleccionar la columna RECEIVED, la columna SUBJECT y la posición de inicio de la serie 'GOOD' dentro de la columna NOTE_TEXT.

  ```
  SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
  FROM IN_TRAY
  WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
  ```

- Localizar el carácter 'ß' en la serie 'Jürgen lives on Hegelstraße', y definir la variable del lenguaje principal LOCATION con la posición, según se mide en unidades CODEUNITS32, dentro de la serie.

  ```
  SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS32)
  ```

  El valor de la variable del lenguaje principal LOCATION se establece en 26.

- Localizar el carácter 'ß' en la serie 'Jürgen lives on Hegelstraße', y establecer la variable del lenguaje principal LOCATION con la posición, según se mide en unidades CODEUNITS16, dentro de la serie.

  ```
  SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS16)
  ```

  El valor de la variable del lenguaje principal LOCATION se establece en 26.

- Localizar el carácter 'ß' en la serie 'Jürgen lives on Hegelstraße', y definir la variable del lenguaje principal LOCATION con la posición, según se mide en OCTETS, dentro de la serie.

  ```
  SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, OCTETS)
  ```

  El valor de la variable del lenguaje principal LOCATION se establece en 27.

- Los ejemplos siguientes funcionan con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación sin avance de espacio. A continuación se muestra esta serie en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

Supongamos que la variable UTF8_VAR contiene la representación UTF-8 de la serie.

```
SELECT LOCATE('~', UTF8_VAR, CODEUNITS16),
  LOCATE('~', UTF8_VAR, CODEUNITS32),
  LOCATE('~', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 4, 3 y 6, respectivamente.

Supongamos que la variable UTF16_VAR contiene la representación UTF-16BE de la serie.

```
SELECT LOCATE('~', UTF16_VAR, CODEUNITS16),
   LOCATE('~', UTF16_VAR, CODEUNITS32),
   LOCATE('~', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

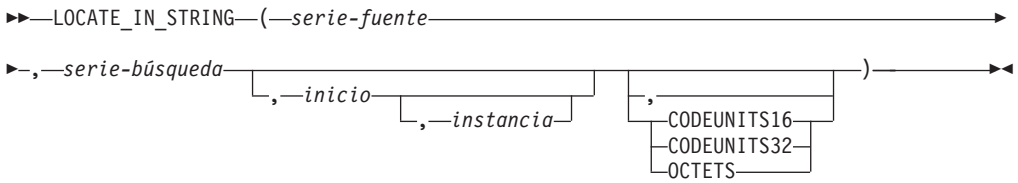
devuelve los valores 4, 3 y 7, respectivamente.

- En una base de datos Unicode creada con la clasificación no sensible a las mayúsculas y UCA500R1_LEN_S1, busque la posición de la palabra 'Brown' en la frase 'The quick brown fox'.

  ```
  SET :LOCATION = LOCATE('Brown', 'The quick brown fox', CODEUNITS16)
  ```

  El valor de la variable del lenguaje principal LOCATION se establece en 11.

LOCATE_IN_STRING



El esquema es SYSIBM

La función LOCATE_IN_STRING devuelve la posición inicial de una serie (denominada *serie-búsqueda*) dentro de otra serie (denominada *serie-fuente*). Si el argumento *serie-búsqueda* no se encuentra y ningún argumento es nulo, el resultado es cero. Si el argumento *serie-búsqueda* se encuentra, el resultado es un número de 1 la longitud real del argumento *serie-fuente*. La búsqueda se realiza utilizando la clasificación de la base de datos, a menos que se defina *serie-búsqueda* o *serie-fuente* como FOR BIT DATA, en cuyo caso la búsqueda se realiza utilizando una comparación binaria.

Si se especifica el argumento opcional *inicio*, éste indica la posición del carácter en el argumento *serie-fuente* en el que debe iniciarse la búsqueda. Si se especifica el *inicio*, también se puede especificar un número de instancia. El argumento *instancia* se utiliza para determinar la posición de una ocurrencia concreta de *serie-búsqueda* dentro de *serie-fuente*. Puede especificarse una unidad de la serie opcional para indicar en qué unidades se expresan el argumento *inicio* y el resultado de la función.

Si la *serie-búsqueda* tiene una longitud cero, el resultado que devuelve la función es 1. Si la *serie-fuente* tiene una longitud cero, el resultado que devuelve la función es 0. Si no se da ninguna de las condiciones y si el valor de *serie-búsqueda* es igual a una longitud idéntica de una subserie de posiciones contiguas dentro del valor de *serie-fuente*, el resultado que devuelve la función es la posición inicial de esa subserie dentro del valor *serie-fuente*; si no, el resultado que devuelve la función es 0.

serie-fuente
Expresión que especifica la serie en la que debe realizarse la búsqueda. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión puede especificarse de cualquiera de las maneras siguientes:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal (incluida una variable localizadora de LOB o una variable de referencia de archivo)
- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores

serie-búsqueda
Expresión que especifica la serie que es objeto de la búsqueda. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, BLOB, VARGRAPHIC, GRAPHIC o CHAR incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC, VARGRAPHIC o BLOB, se convierte implícitamente a VARCHAR antes de evaluar la función. La longitud real no debe superar la longitud máxima de una VARCHAR. La *serie-búsqueda* no puede ser una variable de referencia a archivo BLOB. La expresión puede especificarse de cualquiera de las maneras siguientes:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal
- Una función escalar cuyos argumentos sean cualquiera de los elementos anteriores
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores

Estas normas son parecidas a las que se describen para una *expresión-patrón* para el predicado LIKE.

inicio
Expresión que especifica la posición dentro de *serie-fuente* en la que debe iniciarse la búsqueda de una coincidencia. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si el valor del entero es superior a cero, la búsqueda empieza en *inicio* y continúa en cada posición hasta el final de la serie. Si el valor del entero es inferior a cero, la búsqueda empieza en LENGTH(*serie-fuente*) + *inicio* + 1 y continúa en cada posición hasta el principio de la serie.

Si no se especifica *inicio*, el valor por omisión es 1. Si el valor del entero es cero, se devuelve un error (SQLSTATE 42815).

instancia
Expresión que especifica la instancia de *serie-búsqueda* que se debe buscar en *serie-fuente*. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. Si no se especifica la *instancia*, el valor por omisión es 1. El valor del entero debe ser superior o igual a 1 (SQLSTATE 42815).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie de *inicio* y del resultado. CODEUNITS16 especifica que *inicio* y el resultado deben expresarse en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *inicio* y el resultado deben expresarse en unidades de código UTF-32 de 32 bits. OCTETS especifica que *inicio* y el resultado deben expresarse en bytes.

Si una unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32, y *serie-búsqueda* o *serie-fuente* son una serie binaria o datos de bits, se devuelve un error (SQLSTATE 428GC). Si se especifica una unidad de la serie como OCTETS y *serie-búsqueda* y *serie-fuente* son series binarias, se devuelve un error (SQLSTATE 42815).

Si la unidad de la serie no se especifica de forma explícita, el tipo de datos de la *serie-fuente* determina la unidad que se utiliza. Si la *serie-fuente* son datos gráficos, *inicio* y la posición devuelta se expresan en unidades de dos bytes; en caso contrario, se expresan en bytes.

Si se utiliza una clasificación basada en la UCA sensible a la configuración local para esta función, la opción CODEUNITS16 ofrece las mejores características de rendimiento.

Para obtener más información acerca de CODEUNITS16, CODEUNITS32 y OCTETS, consulte ″Unidades de serie en las funciones incorporadas″ en ″Series de caracteres″.

El primer y segundo argumento deben tener tipos de serie compatibles. Para obtener más información sobre la compatibilidad, consulte el apartado ″Normas para conversiones de series″. En una base de datos Unicode, si un argumento de serie es de tipo carácter (no FOR BIT DATA) y el otro argumento de serie es gráfico, la *serie-búsqueda* se convierta al tipo de datos de la *serie-fuente* para el proceso. Si un argumento es de tipo carácter FOR BIT DATA, el otro argumento no debe ser gráfico (SQLSTATE 42846).

En cada posición de búsqueda, se encuentra una coincidencia cuando los valores de la subserie en esa posición y LENGTH(*serie-búsqueda*) - 1 a la derecha de la posición de búsqueda de la *serie-fuente* equivalen a *serie-búsqueda*.

El resultado de la función es un entero grande. El resultado es la posición inicial de la instancia de *serie-búsqueda* dentro de *serie-fuente*. El valor es relativo al principio de la serie (independientemente de la especificación de *inicio*). Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

INSTR puede utilizarse como sinónimo de LOCATE.

Ejemplos

- Localizar el carácter ’ß’ en la serie ’Jürgen lives on Hegelstraße’ realizando una búsqueda desde el final de la serie y definir la variable del lenguaje principal POSITION con la posición, según se mide en unidades CODEUNITS32, dentro de la serie.

  ```
  SET :POSITION = LOCATE_IN_STRING('Jürgen lives on Hegelstraße',
                                   'ß',-1,CODEUNITS32);
  ```

  El valor de la variable del lenguaje principal POSITION se establece en 26.

- Buscar la ubicación de la tercera aparición del carácter ’N’ en la serie ’WINNING’ realizando una búsqueda desde el inicio de la serie y, a continuación, establecer la variable del lenguaje principal POSITION con la posición del carácter, según se mide en bytes, dentro de la serie.

  ```
  SET :POSITION =
  LOCATE_IN_STRING('WINNING','N',1,3,OCTETS);
  ```

  El valor de la variable del lenguaje principal POSITION se establece en 6.

LOG10

El esquema es SYSIBM. (La versión SYSFUN de la función LOG10 continúa estando disponible.)

La función LOG10 devuelve el logaritmo común (en base 10) de un número.

El argumento debe ser una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función. El valor del argumento debe ser superior a cero (SQLSTATE 22003).

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas

- ***Resultados que implican valores especiales de DECFLOAT:*** para valores de coma flotante decimal, los valores especiales se tratan como se indica a continuación:
  - LOG10(NaN) devuelve NaN.
  - LOG10(-NaN) devuelve -NaN.
  - LOG10(Infinity) devuelve Infinity.
  - LOG10(-Infinity) devuelve NaN y un aviso.
  - LOG10(sNaN) devuelve NaN y un aviso.
  - LOG10(-sNaN) devuelve -NaN y un aviso.
  - LOG10(DECFLOAT('0') devuelve -Infinity.

Ejemplo

- Supongamos que L es una variable del lenguaje principal DECIMAL(4,2) con un valor de 31,62.

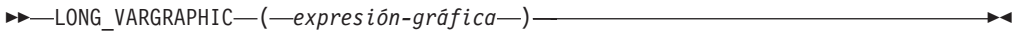
  ```
  VALUES LOG10(:L)
  ```

  Devuelve el valor de DOUBLE +1,49996186559619E+000.

LONG_VARCHAR

La función LONG_VARCHAR ha quedado obsoleta y se puede eliminar en un futuro release. La función es compatible con versiones de DB2 anteriores.

LONG_VARGRAPHIC



La función LONG_VARGRAPHIC ha quedado obsoleta y se puede eliminar en un futuro release. La función es compatible con versiones de DB2 anteriores.

LOWER

El esquema es SYSIBM. (La versión SYSFUN de esta función sigue estando disponible con el soporte para los argumentos CLOB).

La función LOWER devuelve una serie en la que todos los caracteres SBCS se han convertido a minúsculas. Es decir, los caracteres de la A a la Z se convertirán en los caracteres de la a a la z y el resto de caracteres se convertirán en sus minúsculas equivalentes, si existen. Por ejemplo, en la página de códigos 850, É se correlación con é. Si la longitud de elemento de código del carácter de resultado no es la misma que la del elemento de código del carácter fuente, el carácter fuente no se convierte. Dado que no todos los caracteres se convierten, LOWER(UPPER(*expresión-serie*)) no devuelve necesariamente el mismo resultado que LOWER(*expresión-serie*).

El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR.

El resultado de la función tiene el mismo tipo de datos y el mismo atributo de longitud que el argumento. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

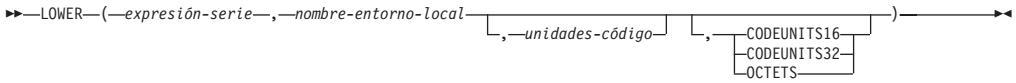
Ejemplo:

Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelvan en minúsculas.

```
SELECT LOWER(JOB)
  FROM EMPLOYEE
  WHERE EMPNO = '000020';
```

El resultado es el valor 'director'.

LOWER (sensible al entorno local)



El esquema es SYSIBM.

La función LOWER devuelve una serie en la que todos los caracteres se han convertido a minúsculas utilizando las normas asociadas con el entorno local especificado.

expresión-serie
Expresión que devuelve una serie CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si *expresión-serie* es CHAR o VARCHAR, la expresión no debe ser FOR BIT DATA (SQLSTATE 42815).

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local que define las normas de conversión a minúsculas. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery".

unidades-código
Constante entera que especifica el número de unidades de código en el resultado. Si se especifica, *unidades-código* debe ser un entero entre 1 y 32.672 si el resultado son datos de tipo carácter o entre 1 y 16.336 si el resultado son datos gráficos (SQLSTATE 42815). Si *unidades-código* no se especifica de forma explícita, es implícitamente el atributo de longitud de *expresión-serie*. Si se especifica OCTETS y el resultado son datos gráficos, el valor de *unidades-código* debe ser par (SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de serie de *unidades-código*.

CODEUNITS16 especifica que *unidades-código* se expresa en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *unidades-código* se expresa en unidades de código UTF-32 de 32 bits. OCTETS especifica que *unidades-código* se expresa en bytes.

Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *unidades-código* se expresa en unidades de dos bytes; de lo contrario, se expresa en bytes. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El resultado de la función es VARCHAR si *expresión-serie* es CHAR o VARCHAR y VARGRAPHIC si *expresión-serie* es GRAPHIC o VARGRAPHIC.

El atributo de longitud del resultado lo determina el valor implícito o explícito de *unidades-código*, la unidad de serie implícita o explícita y el tipo de datos de resultado, como se muestra en la tabla siguiente:

Tabla 49. Atributo de longitud del resultado de LOWER como función de unidad de serie y tipo de resultado

Unidad de serie	Tipo de resultado de caracteres	Tipo de resultado de gráfico
CODEUNITS16	MIN(*unidades-código* * 3, 32672)	*unidades-código*
CODEUNITS32	MIN(*unidades-código* * 4, 32672)	MIN(*unidades-código* * 2, 16336)
OCTETS	*unidades-código*	MIN(*unidades-código* / 2, 16336)

La longitud real del resultado puede ser mayor que la longitud de *expresión-serie*. Si la longitud real del resultado es mayor que el atributo de longitud del resultado, se devuelve un error (SQLSTATE 42815). Si el número de unidades de código del resultado excede el valor de *unidades-código*, se devuelve un error (SQLSTATE 42815).

Si *expresión-serie* no está en UTF-16, esta función realiza la conversión de página de códigos de *expresión-serie* a UTF-16 y del resultado de UTF-16 a la página de códigos de *expresión-serie*. Si cualquiera de las conversiones de página de códigos produce como mínimo un carácter de sustitución, el resultado incluye el carácter de sustitución, se devuelve un aviso (SQLSTATE 01517) y el distintivo de aviso SQLWARN8 de la SQLCA se establece en 'W'.

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelvan en minúsculas.

```
SELECT LOWER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

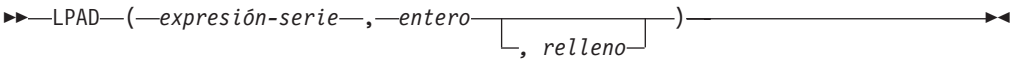
  El resultado es el valor 'director'.

- Busque las minúsculas de todos los caracteres 'I' de una serie en idioma turco.

```
VALUES LOWER('IIii', 'tr_TR', CODEUNITS16)
```

  El resultado es la serie 'iiii'.

LPAD



El esquema es SYSIBM.

La función LPAD devuelve una serie compuesta de *expresión-serie* que se rellena por la izquierda con *relleno* o con espacios en blanco. La función LPAD trata los blancos iniciales o finales de *expresión-serie* como significativos. El relleno solo se utiliza si la longitud real de *expresión-serie* es menor que *entero* y si *relleno* no es una serie vacía.

expresión-serie
Expresión que especifica la serie fuente. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

entero
Expresión entera que especifica la longitud del resultado. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor debe ser cero o un entero positivo que sea menor o igual que *n*, siendo *n* 32.672 si *expresión-serie* es una serie de caracteres o 16.336 si *expresión-serie* es una serie gráfica.

relleno
Expresión que especifica la serie con la que se realizará el relleno. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

Si *relleno* no se especifica, el carácter de relleno se determina del modo siguiente:

- Carácter en blanco SBCS si *expresión-serie* es una serie de caracteres.
- Carácter en blanco ideográfico si *expresión-serie* es una serie gráfica. Para las series gráficas de una base de datos EUC, se utiliza X'3000'. Para las series gráficas de una base de datos Unicode, se utiliza X'0020'.

El resultado de la función es una serie de longitud variable que tiene la misma página de códigos que *expresión-serie*. El valor de *expresión-serie* y el valor de *relleno* deben tener tipos de datos compatibles. Si *expresión-serie* y *relleno* tienen páginas de códigos diferentes, *relleno* se convertirá a la página de códigos de *expresión-serie*. Si *expresión-serie* o *relleno* es FOR BIT DATA, no se lleva a cabo ninguna conversión de caracteres.

El atributo de longitud del resultado depende de si el valor de *entero* está disponible cuando se compila la sentencia de SQL que contiene la invocación de la función (por ejemplo, si se especifica como constante o como expresión de constante) o de si está disponible únicamente cuando se ejecuta la función (por ejemplo, si se especifica como resultado de la invocación de una función). En el caso de que el valor esté disponible cuando se compila la sentencia de SQL que

contiene la invocación de la función, si *entero* es mayor que cero, el atributo de longitud del resultado es *entero*. En el caso de que *entero* sea 0, el atributo de longitud del resultado es 1. Si el valor está disponible únicamente cuando se ejecuta la función, el atributo de longitud del resultado se determina según lo indicado en la tabla siguiente:

Tabla 50. Cómo se determina la longitud del resultado si entero está disponible únicamente cuando se ejecuta la función

Tipo de datos de *expresión-serie*	**Longitud del tipo de datos del resultado**
CHAR(*n*) o VARCHAR(*n*)	Mínimo de *n*+100 y 32.672
GRAPHIC(*n*) o VARGRAPHIC(*n*)	Mínimo de *n*+100 y 16.336

La longitud real del resultado se determina a partir de *entero*. Si *entero* es 0, la longitud real es 0 y el resultado es una serie vacía. Si *entero* es menor que la longitud real de *expresión-serie*, la longitud real es *entero* y el resultado se trunca.

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La consulta siguiente rellenaría completamente un valor con puntos por la izquierda:

  ```
  SELECT LPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME ---------------
  ..........Chris
  ............Meg
  ...........Jeff
  ```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La consulta siguiente solo rellenaría cada valor hasta una longitud de 5:

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME -----
  Chris
  ..Meg
  .Jeff
  ```

- Supongamos que NAME es una columna CHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La función LPAD no incluye ningún relleno porque NAME es un campo de caracteres de longitud fija y ya está rellenado con espacios en blanco. Sin embargo, puesto que la longitud del resultado es 5, las columnas de truncan:

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME -----
  Chris
  Meg
  Jeff
  ```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. En algunos casos, se devuelve una instancia parcial de la especificación de relleno:

  ```
  SELECT LPAD(NAME,15,'123' ) AS NAME FROM T1;
  ```

devuelve:

```
NAME ---------------
1231231231Chris
123123123123Meg
12312312312Jeff
```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. Observe que “Chris” se trunca, “Meg” se rellena y “Jeff” no cambia:

```
SELECT LPAD(NAME,4,'.' ) AS NAME FROM T1;
```

devuelve:

```
NAME ----
Chri
.Meg
Jeff
```

LTRIM

El esquema es SYSIBM. (La versión SYSFUN de esta función sigue estando disponible con el soporte para los argumentos CLOB).

La función LTRIM elimina los blancos del principio de la *expresión-serie*.

La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte iniciales.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 iniciales.
- De lo contrario, se eliminan los blancos de un solo byte iniciales.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si elimina todos los caracteres se obtiene una serie vacía de longitud variable (longitud de cero).

Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

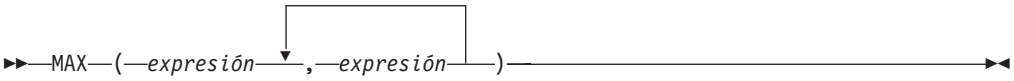
Ejemplo:

Supongamos que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor de ' Hola'.

```
VALUES LTRIM(:HELLO)
```

El resultado es 'Hola'.

MAX



El esquema es SYSIBM.

La función MAX devuelve el valor máximo de un conjunto de valores.

Los argumentos deben ser compatibles, y cada argumento debe ser una expresión que devuelva un valor de cualquier tipo de datos distinto de ARRAY, LOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42815). Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Dado que esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para soportar tipos diferenciados definidos por el usuario.

El argumento seleccionado se convierte, si es necesario, a los atributos del resultado. Los atributos del resultado los determinan todos los operandos basándose en las normas de los tipos de datos de resultado.

El resultado de la función es el valor de argumento más grande. Si como mínimo un argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

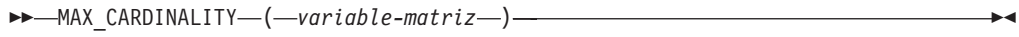
La función escalar MAX es sinónimo de la función escalar GREATEST.

Ejemplo:

Devolver la bonificación de un empleado, el valor que sea mayor entre 500 y el 5% del salario del empleado.

```
SELECT EMPNO, MAX(SALARY * 0.05, 500)
  FROM EMPLOYEE
```

MAX_CARDINALITY



El esquema es SYSIBM.

La función MAX_CARDINALITY devuelve un valor del tipo BIGINT que representa el número máximo de elementos que puede contener una matriz. Esta es la cardinalidad que se especificó en la sentencia CREATE TYPE para el tipo de matriz común.

variable-matriz
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz, o especificación CAST de un marcador de parámetro a un tipo de matriz.

El resultado puede ser nulo; si el argumento es nulo o una matriz asociativa, el resultado es el valor nulo.

Ejemplo

- Devolver la máxima cardinalidad de la variable de matriz RECENT_CALLS de tipo de matriz PHONENUMBERS:

```
SET LIST_SIZE = MAX_CARDINALITY(RECENT_CALLS)
```

  La variable de SQL LIST_SIZE está establecida en 50, que es la máxima cardinalidad con la que se definió el tipo de matriz PHONENUMBERS.

MICROSECOND

El esquema es SYSIBM.

La función MICROSECOND devuelve la parte correspondiente a los microsegundos de un valor.

El argumento debe ser un valor DATE, TIMESTAMP, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que sea de un tipo de datos CHAR o VARCHAR. Si un argumento proporcionado es un valor DATE, éste primero se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00). En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

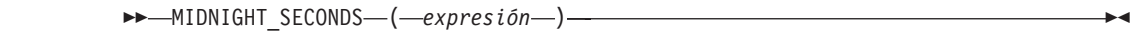
- Si el argumento es un valor DATE, TIMESTAMP o una representación de serie válida de una fecha o indicación de fecha y hora:
  - El entero está en el rango de 0 a 999.999.
  - Si la precisión de la indicación de fecha y hora supera 6, el valor se truncará.
- Si el argumento es una duración:
  - El resultado refleja la parte de microsegundos del valor que es un entero entre -999.999 y 999.999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo

- Supongamos que una tabla TABLEA contiene dos columnas, TS1 y TS2, del tipo TIMESTAMP. Seleccione todas las filas cuya parte correspondiente a los microsegundos de TS1 no sea cero y las partes correspondientes a los segundos de TS1 y TS2 sean idénticas.

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0
    AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



El esquema es SYSFUN.

Devuelve un valor entero en el rango de 0 a 86.400, que representa el número de segundos entre medianoche y el valor de hora especificado en el argumento.

Expresión que devuelve un valor que debe ser de tipo DATE, TIME o TIMESTAMP o una representación de serie válida de una fecha, hora o indicación de fecha y hora que no sea un CLOB ni un DBCLOB.

Si *expresión* es un valor DATE o una representación de serie válida de una fecha, ésta primero se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00).

Sólo las bases de datos Unicode dan soporte a un argumento que sea una representación de serie gráfica de una fecha, una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Si *expresión* es una serie de caracteres, los espacios en blanco iniciales se incluyen y los espacios en blanco finales se eliminan antes de convertir el valor en un valor de fecha y hora. Para conocer los formatos válidos de las representaciones de serie de los valores de fecha y hora, consulte "Representación mediante series de los valores de fecha y hora" en "Valores de fecha y hora".

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Encuentre el número de segundos entre la medianoche y 00:10:10 y la medianoche y 13:10:10.

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  Este ejemplo devuelve lo siguiente:

  ```
  1           2
  ----------- -----------
          610       47410
  ```

  Puesto que un minuto es 60 segundos, hay 610 segundos entre la medianoche y la hora especificada. Es lo mismo para el segundo ejemplo. Hay 3600 segundos en una hora y 60 segundos en un minuto, lo que da como resultado 47.410 segundos entre la hora especificada y la medianoche.
- Encuentre el número de segundos entre la medianoche y 24:00:00, y la medianoche y 00:00:00.

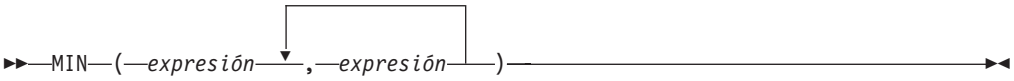
  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  Este ejemplo devuelve lo siguiente:

  ```
  1           2
  ----------- -----------
        86400           0
  ```

  Observe que estos dos valores representan el mismo momento en el tiempo, pero devuelven distintos valores MIDNIGHT_SECONDS.

MIN



El esquema es SYSIBM.

La función MIN devuelve el valor mínimo de un conjunto de valores.

Los argumentos deben ser compatibles, y cada argumento debe ser una expresión que devuelva un valor de cualquier tipo de datos distinto de ARRAY, LOB, XML, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42815). Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Dado que esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para soportar tipos diferenciados definidos por el usuario.

El argumento seleccionado se convierte, si es necesario, a los atributos del resultado. Los atributos del resultado los determinan todos los operandos basándose en las normas de los tipos de datos de resultado.

El resultado de la función es el valor de argumento más pequeño. Si como mínimo un argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

La función escalar MIN es sinónimo de la función escalar LEAST.

Ejemplo:

Devolver la bonificación de un empleado, el valor que sea menor entre 5000 y el 5% del salario del empleado.

```
SELECT EMPNO, MIN(SALARY * 0.05, 5000)
  FROM EMPLOYEE
```

MINUTE

El esquema es SYSIBM.

La función MINUTE devuelve la parte correspondiente a los minutos de un valor.

El argumento debe ser un valor DATE, TIME o TIMESTAMP, una duración de hora, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha, hora o indicación de fecha y hora que no sea un CLOB. Si un argumento proporcionado es un valor DATE, éste primero se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00). En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

- Si el argumento es un valor DATE, TIME, TIMESTAMP o una representación de serie válida de una fecha, hora o indicación de fecha y hora:
  - El resultado es la parte correspondiente a los minutos de un valor, que es un entero entre 0 y 59.
- Si el argumento es una duración de hora o una duración de indicación de fecha y hora:
  - El resultado es la parte correspondiente a los minutos del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos

- Utilizando la tabla de ejemplo CL_SCHED, seleccione todas las clases con una duración inferior a 50 minutos.

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0
    AND
  MINUTE(ENDING - STARTING) < 50
```

MOD

El esquema es SYSFUN.

Devuelve el resto del primer argumento dividido por el segundo argumento. El resultado sólo es negativo si el primer argumento es negativo.

El resultado de la función es:

- SMALLINT si ambos argumentos son SMALLINT
- INTEGER si un argumento es INTEGER y el otro es INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

MONTH

El esquema es SYSIBM.

La función MONTH devuelve la parte correspondiente al mes de un valor.

El argumento debe ser un valor DATE, TIMESTAMP, una duración de fecha, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

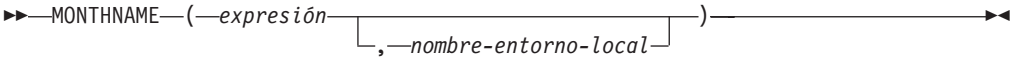
- Si el argumento es un valor DATE, TIMESTAMP o una representación de serie válida de una fecha o una indicación de fecha y hora:
  - El resultado es la parte correspondiente al mes del valor, que es un entero entre 1 y 12.
- Si el argumento es una duración de fecha o duración de la indicación de fecha y hora:
  - El resultado es la parte correspondiente al mes del valor, que es un entero entre -99 y 99. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplo

- Seleccione todas las filas de la tabla EMPLOYEE de las personas que han nacido (BIRTHDATE) en diciembre (DECEMBER).

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME



El esquema es SYSIBM. La versión SYSFUN de la función **MONTHNAME** sigue estando disponible.

La función **MONTHNAME** devuelve una serie de caracteres que contiene el nombre del mes (por ejemplo, enero) para la parte correspondiente al día de la *expresión*, basada en el *nombre-entorno-local* o el valor del registro especial CURRENT LOCALE LC_TIME.

expresión
Expresión que devuelve un valor de uno de los tipos de datos incorporados siguientes: DATE, TIMESTAMP o una representación de serie de caracteres de una fecha o una indicación de fecha y hora que no sea CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local utilizado para determinar el idioma del resultado. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte ″Nombres de entorno local para SQL y XQuery″. Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado es una serie de caracteres de longitud variable. El atributo de longitud es 100. Si la serie resultante supera el atributo de longitud del resultado, el resultado se truncará. Si el argumento *expresión* puede ser nulo, el resultado puede ser nulo; si el argumento *expresión* es nulo, el resultado es el valor nulo. La página de códigos del resultado es la página de códigos de la sección.

Notas

- **Calendario juliano y gregoriano**: esta función tiene en cuenta la transición del calendario juliano al calendario gregoriano el 15 de octubre de 1582. Sin embargo, la versión SYSFUN de la función MONTHNAME adopta el calendario gregoriano para todos los cálculos.
- **Determinismo:** MONTHNAME es una función determinista. No obstante, cuando no se especifica de forma explícita *nombre-entorno-local*, la invocación de la función depende del valor del registro especial CURRENT LOCALE LC_TIME. Esta invocación que depende del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales (SQLSTATE 42621 o 428EC).

Ejemplo

- Suponga que la variable TMSTAMP se ha definido como TIMESTAMP y tiene el valor siguiente: 2007-03-09-14.07.38.123456. Los ejemplos siguientes muestran varias invocaciones de la función y los valores de serie resultantes. En cada caso el tipo de resultado es VARCHAR(100).

```
Invocación de función                          Resultado
-------------------------                     ----------
MONTHNAME (TMSTAMP, 'CLDR 1.5:en_US')            March
MONTHNAME (TSMTAMP, 'CLDR 1.5:de_DE')            Marz
MONTHNAME (TMSTAMP, 'CLDR 1.5:fr_FR')            mars
```

MONTHS_BETWEEN

El esquema es SYSIBM.

La función MONTHS_BETWEEN devuelve una estimación del número de meses entre *expresión1* y *expresión2*.

expresión1 **o** *expresión2*
Expresiones que devuelvan un valor de tipo de datos DATE o bien TIMESTAMP.

Si *expresión1* representa una fecha posterior a la de *expresión2*, el resultado es positivo. Si *expresión1* representa una fecha anterior a la de expresión2, el resultado es negativo.

- Si *expresión1* y *expresión2* representan fechas o indicaciones de fecha y hora con el mismo día del mes, o si ambos argumentos representan el último día de su mes respectivo, el resultado es la diferencia en forma de número entero en función de los valores de año y mes pasando por alto las partes de hora de los argumentos de indicación de fecha y hora.
- En caso contrario, el número entero que forma parte del resultado es la diferencia basada en los valores de año y mes. La parte fraccionaria del resultado se calcula a partir del resto basándose en que todos los meses tienen 31 días. Si alguno de los argumentos representa una indicación de fecha y hora, los argumentos se procesan correctamente como indicaciones de fecha y hora con precisión máxima, y las partes de hora de estos valores también se tienen en cuenta al calcular el resultado.

El resultado de la función es un DECIMAL(31,15). Si algún argumento puede ser nulo, el resultado puede ser nulo. Si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

Ejemplos

- Calcule el número de meses que durará el proyecto AD3100. Supongamos que la fecha de inicio es 1982-01-01 y la fecha de finalización es 1983-02-01.

```
SELECT MONTHS_BETWEEN (PRENDATE, PRSDATE)
   FROM PROJECT
   WHERE PROJNO='AD3100'
```

  El resultado e 13.000000000000000.

Aquí se ofrecen algunos ejemplos más:

Tabla 51. Ejemplos adicionales en los que se utiliza MONTHS_BETWEEN

Valor para el argumento *e1*	**Valor para el argumento** *e2*	**Valor devuelto por MONTHS_BETWEEN (***e1***,***e2***)**	**Valor devuelto por ROUND (MONTHS_BETWEEN (***e1***,***e2***)*31,2)**	**Comentario**
2005-02-02	2005-01-01	1.032258064516129	32.00	
2007-11-01-09.00.00.00000	2007-12-07-14.30.12.12345	-1.200945386592741	-37.23	
2007-12-13-09.40.30.00000	2007-11-13-08.40.30.00000	1.000000000000000	31.00	Véase la nota 1

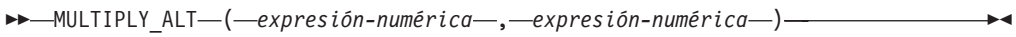
Tabla 51. Ejemplos adicionales en los que se utiliza MONTHS_BETWEEN (continuación)

Valor para el argumento *e1*	**Valor para el argumento** *e2*	**Valor devuelto por MONTHS_BETWEEN (***e1***,***e2***)**	**Valor devuelto por ROUND (MONTHS_BETWEEN (***e1***,***e2***)*31,2)**	**Comentario**
2007–03–15	2007–02–20	0.838709677419354	26.00	Véase la nota 2
2008-02-29	2008-02-28-12.00.00	0.016129032258064	0.50	
2008-03-29	2008-02-29	1.000000000000000	31.00	
2008-03-30	2008-02-29	1.032258064516129	32.00	
2008-03-31	2008-02-29	1.000000000000000	31.00	Véase la nota 3

Nota:

1. La diferencia de hora se pasa por alto porque el día del mes es el mismo en ambos valores.
2. El resultado no es 23 ya que, aunque el mes de febrero tiene 28 días, se supone que todos los meses tienen 31 días.
3. El resultado no es 33 porque ambas fechas son el último día de su mes respectivo, de modo que el resultado solamente se basa en las partes del año y del mes.

MULTIPLY_ALT



El esquema es SYSIBM.

La función escalar MULTIPLY_ALT devuelve el producto de los dos argumentos. Se proporciona como alternativa al operador de multiplicación, especialmente cuando la suma de las precisiones decimales de los argumentos es mayor que 31.

Los argumentos pueden ser de cualquier tipo de datos numérico incorporado.

El resultado de la función es DECIMAL cuando los dos argumentos son tipos de datos numéricos exactos (DECIMAL, BIGINT, INTEGER o SMALLINT); en caso contrario, la operación se realiza utilizando la aritmética de coma flotante decimal y el resultado de la función es una coma flotante decimal cuya precisión viene determinada por el tipo de datos de los argumentos, de la misma forma en la que se determina la precisión para la aritmética de coma flotante decimal. Un argumento de serie o coma decimal se convierte a DECFLOAT(34) antes de evaluar la función.

Cuando el resultado de la función es DECIMAL, la precisión y la escala del resultado se determinan del modo siguiente, utilizando los símbolos *p* y *s* para indicar la precisión y la escala del primer argumento y los símbolos *p'* y *s'* para indicar la precisión y la escala del segundo argumento.

- La precisión es MIN(31, *p* + *p'*)
- La escala es:
  - 0 si la escala de ambos argumentos es 0
  - MIN(31, *s* + *s'*) si *p* + *p'* es inferior o igual a 31
  - MAX(MIN(3, *s* + *s'*), 31 - (*p* - *s* + *p'* - *s'*)) si *p* + *p'* es superior a 31.

El resultado puede ser un valor nulo si, como mínimo, un argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado será el valor nulo si uno de los argumentos es nulo.

Es mejor elegir la función MULTIPLY_ALT que el operador de multiplicación cuando se realizan operaciones aritméticas decimales donde se necesita como mínimo una escala de 3 y la suma de las precisiones excede de 31. En estos casos, se realiza el cálculo interno a fin de evitar desbordamientos. Entonces se asigna el resultado final al tipo de datos de resultado, utilizando el truncamiento donde sea necesario para coincidir con la escala. Tenga en cuenta que el desbordamiento del resultado final sigue siendo posible cuando la escala es 3.

A continuación se proporciona un ejemplo en el que se comparan los tipos de resultado utilizando MULTIPLY_ALT y el operador de multiplicación.

Tipo de argumento 1	Tipo de argumento 2	Resultado utilizando MULTIPLY_ALT	Resultado utilizando el operador de multiplicación
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)

Tipo de argumento 1	Tipo de argumento 2	Resultado utilizando MULTIPLY_ALT	Resultado utilizando el operador de multiplicación
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

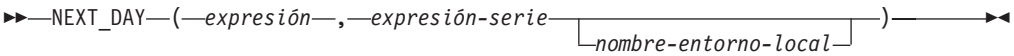
Ejemplo:

Multiplique dos valores donde el tipo de datos del primer argumento es DECIMAL(26,3) y el tipo de datos del segundo argumento es DECIMAL(9,8). El tipo de datos del resultado es DECIMAL(31,7).

```
values multiply_alt(98765432109876543210987.654,5.43210987)
1
--------------------------------
536504678578875294857887.5277415
```

Observe que el producto completo de estos dos números es 536504678578875294857887.52774154498, pero los 4 últimos dígitos están truncados para coincidir con la escala del tipo de datos de resultado. Si se utiliza el operador de multiplicación con los mismos valores se producirá un desbordamiento aritmético, dado que el tipo de datos de resultado es DECIMAL(31,11) y el valor de resultado tiene 24 dígitos a la izquierda del decimal, pero el tipo de datos de resultado sólo soporta 20 dígitos.

NEXT_DAY



El esquema es SYSIBM.

La función escalar NEXT_DAY devuelve un valor de fecha y hora que representa el primer día de la semana, indicado en *expresión-serie*, que sigue a la fecha de *expresión*.

expresión
Una expresión que devuelve un valor de uno de los siguientes tipos de datos incorporados: valor DATE o valor TIMESTAMP.

expresión-serie
Expresión que devuelve un tipo de datos de carácter integrado. El valor debe ser un día válido de la semana para *nombre-entorno-local*. El valor se puede especificar como el nombre completo del día o como la abreviatura correspondiente. Por ejemplo, si el entorno local es 'en_US', los valores válidos son los siguientes:

Tabla 52. Días de la semana y abreviaturas válidos para el entorno local 'en_US'

Día de la semana	Abreviatura
MONDAY	MON
TUESDAY	TUE
WEDNESDAY	WED
THURSDAY	THU
FRIDAY	FRI
SATURDAY	SAT
SUNDAY	SUN

La longitud mínima del valor de entrada es la longitud de la abreviatura. Los caracteres se pueden especificar en mayúsculas o en minúsculas; todos los caracteres que pueda haber inmediatamente después de una abreviatura válida se pasan por alto.

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local utilizado para determinar el idioma del valor de *expresión-serie*. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función tiene el mismo tipo de datos que *expresión*, a menos que *expresión* sea una serie, en cuyo caso el tipo de datos del resultado es TIMESTAMP(6). El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

La función no modifica la información de horas, minutos, segundos o segundos fraccionados incluida en *expresión*. Si *expresión* es una serie que representa una fecha, la información de hora del valor TIMESTAMP resultante se establece en cero.

Notas

- **Determinismo:** NEXT_DAY es una función determinista. No obstante, cuando no se especifica de forma explícita nombre-entorno-local, la invocación de la función depende del valor del registro especial CURRENT LOCALE LC_TIME. Las invocaciones de la función que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales.

Ejemplos

- Establecer la variable NEXTDAY con la fecha del martes que sigue al 24 de abril de 2007.

  ```
  NEXTDAY = NEXT_DAY(DATE '2007-04-24', 'TUESDAY')
  ```

  La variable NEXTDAY se establece con el valor '2007–05–01–00.00.00.000000', ya que el 24 de abril de 2007 es martes.

- Establecer la variable vNEXTDAY con la fecha del primer lunes del mes de mayo de 2007. Supongamos que la variable vDAYOFWEEK = 'MON'.

  ```
  vNEXTDAY = NEXT_DAY(LAST_DAY(CURRENT_TIMESTAMP),vDAYOFWEEK)
  ```

  La variable vNEXTDAY se establece con el valor '2007-05-07-12.01.01.123456', suponiendo que el valor del registro especial CURRENT_TIMESTAMP es '2007-04-24-12.01.01.123456'.

NORMALIZE_DECFLOAT

El esquema es SYSIBM.

La función NORMALIZE_DECFLOAT devuelve un valor de coma flotante decimal igual al argumento de entrada establecido en su forma más sencilla; es decir, se eliminan los ceros finales en el coeficiente de un número distinto de cero que los tenga. Para ello es posible que sea necesario representar el número en formato normalizado dividiendo el coeficiente por la potencia de diez apropiada y ajustando el exponente como corresponde. El exponente de un valor cero se establece en 0.

expresión
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Los argumentos del tipo SMALLINT, INTEGER, REAL, DOUBLE o DECIMAL(*p*,*s*), donde *p* <= 16, se convierten a DECFLOAT(16) para su proceso. Los argumentos del tipo BIGINT o DECIMAL(*p*,*s*), donde *p* > 16, se convierten a DECFLOAT(34) para su proceso.

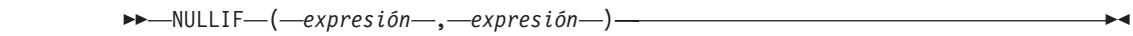
El resultado de la función es un valor DECFLOAT(16) si el tipo de datos de la expresión después de la conversión a coma flotante decimal es DECFLOAT(16). De lo contrario, el resultado de la función es un valor DECFLOAT(34). Si el argumento es un valor de coma flotante decimal especial, el resultado es el mismo valor de coma flotante decimal especial. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Los ejemplos siguientes muestran los valores que devuelve la función NORMALIZE_DECFLOAT, si se da una variedad de valores de coma flotante decimal de entrada:

```
NORMALIZE_DECFLOAT(DECFLOAT(2.1)) = 2.1
NORMALIZE_DECFLOAT(DECFLOAT(-2.0)) = -2
NORMALIZE_DECFLOAT(DECFLOAT(1.200)) = 1.2
NORMALIZE_DECFLOAT(DECFLOAT(-120)) = -1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(120.00)) = 1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(0.00)) = 0
NORMALIZE_DECFLOAT(-NAN) = -NaN
NORMALIZE_DECFLOAT(-INFINITY) = -Infinity
```

NULLIF



El esquema es SYSIBM.

La función NULLIF devuelve un valor nulo si los argumentos son iguales, de lo contrario, devuelve el valor del primer argumento.

Los argumentos deben poderse comparar. Pueden ser de un tipo de datos interno (que no sea una serie LOB) o de un tipo de datos diferenciado (que no esté basado en una serie LOB). (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como esta función acepta cualquier tipo de datos compatible como argumento, no es necesario crear signaturas adicionales para dar soporte a los tipos de datos diferenciados definidos por el usuario). Los atributos del resultado son los atributos del primer argumento.

El resultado de utilizar NULLIF(e1,e2) es igual a utilizar la expresión

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

Observe que cuando e1=e2 se evalúa como desconocido (porque uno o los dos argumentos son NULL), las expresiones CASE lo consideran como no verdadero. Por lo tanto, en esta situación, NULLIF devuelve el valor del primer argumento.

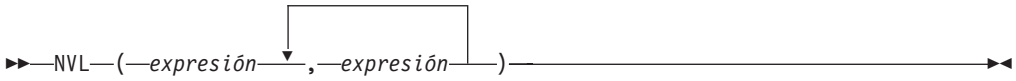
Ejemplo:

- Supongamos que las variables del lenguaje principal PROFIT, CASH y LOSSES tienen tipos de datos DECIMAL con los valores 4500.00, 500.00 y 5000.00 respectivamente:

  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  Devuelve un valor nulo.

NVL

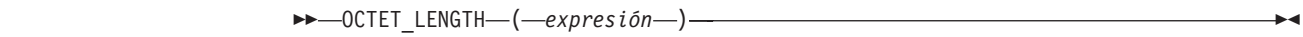


El esquema es SYSIBM.

La función NVL devuelve el primer argumento que no es nulo.

NVL es sinónimo de COALESCE.

OCTET_LENGTH



El esquema es SYSIBM.

La función OCTET_LENGTH devuelve la longitud de *expresión* en octetos (bytes).

expresión
Expresión que devuelve un valor que es un tipo de datos de serie incorporado.

El resultado de la función es INTEGER. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La longitud de las series de caracteres o de gráficos incluye espacios en blanco finales. La longitud de las series binarias incluye ceros binarios. La longitud de las series de longitud variable es la longitud real y no la longitud máxima.

Para una portabilidad mayor, codifique la aplicación para que sea capaz de aceptar un resultado de tipo de datos DECIMAL(31).

Ejemplos:

- Supongamos que la tabla T1 tiene una columna GRAPHIC(10) denominada C1.

  ```
  SELECT OCTET_LENGTH(C1) FROM T1
  ```

  devuelve el valor 20.

- El ejemplo siguiente funciona con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

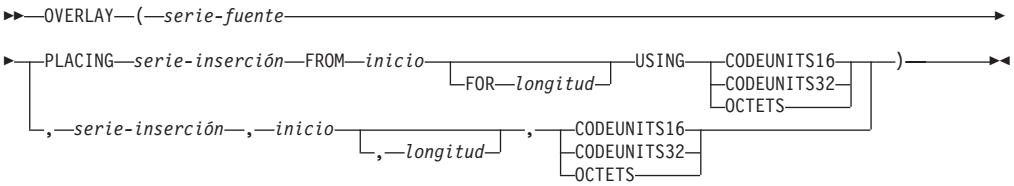
	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  Supongamos que las variables UTF8_VAR y UTF16_VAR contienen las representaciones de la serie en UTF-8 y UTF-16BE respectivamente.

  ```
  SELECT OCTET_LENGTH(UTF8_VAR),
    OCTET_LENGTH(UTF16_VAR)
  FROM SYSIBM.SYSDUMMY1
  ```

  devuelve los valores 9 y 12 respectivamente.

OVERLAY



El esquema es SYSIBM.

La función OVERLAY devuelve una serie en la que, a partir de *inicio* en la *serie-fuente*, se ha suprimido la *longitud* de las unidades de código especificadas y se ha insertado una *serie-inserción*.

serie-fuente
Expresión que especifica la serie fuente. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función.

serie-inserción
Expresión que especifica la serie que se debe insertar en *serie-fuente*, a partir de la posición identificada por *inicio*. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. Si la página de códigos de la *serie-inserción* difiere de la página de códigos de la *serie-fuente*, *serie-inserción* se convierte a la página de códigos de la *serie-fuente*.

inicio
Expresión que devuelve un valor entero. El valor entero especifica el punto de partida en la serie fuente donde debe empezar la supresión de bytes y la inserción de otra serie. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe estar comprendido entre 1 y la longitud de la *serie-fuente* más uno (SQLSTATE 42815). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número impar entre 1 y el doble del atributo de longitud de *serie-fuente* más uno (SQLSTATE 428GC).

longitud
Expresión que especifica el número de unidades de código (en las unidades de serie especificadas) que se deben suprimir de la serie fuente, a partir de la posición identificada por *inicio*. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe ser entre 0 y la longitud de la *serie-fuente*, expresada en unidades que se hayan especificado explícita o implícitamente (SQLSTATE 22011). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número par entre 0 y el doble del atributo de longitud de *serie-fuente* (SQLSTATE 428GC).

El hecho de no especificar *longitud* equivale a especificar un valor de 1, excepto cuando se especifica OCTETS y el resultado son datos gráficos, en cuyo caso el hecho de no especificar *longitud* equivale a especificar un valor de 2.

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie de *inicio* y *longitud*.

CODEUNITS16 especifica que *inicio* y *longitud* se expresan en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *inicio* y *longitud* se expresan en unidades de código UTF-32 de 32 bits. OCTETS especifica que *inicio* y *longitud* se expresan en bytes.

Si la unidad de serie se especifica como CODEUNITS16 o CODEUNITS32 y el resultado es una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si la unidad de serie se especifica como OCTETS y *serie-inserción* y *serie-fuente* son series binarias, se devuelve un error (SQLSTATE 42815). Si la unidad de serie se especifica como OCTETS, la operación se realiza en la página de códigos de la *serie-fuente*. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El tipo de datos del resultado depende de los tipos de datos de *serie-fuente* y *serie-inserción*, como se muestra en la tabla siguiente de combinaciones de tipos soportadas.

Tabla 53. Tipo de datos del resultado como función de los tipos de datos de serie-fuente y serie-inserción

serie-fuente	*serie-inserción*	**Resultado**
CHAR o VARCHAR	CHAR o VARCHAR	VARCHAR
GRAPHIC o VARGRAPHIC	GRAPHIC o VARGRAPHIC	VARGRAPHIC
CLOB	CHAR, VARCHAR o CLOB	CLOB
DBCLOB	GRAPHIC, VARGRAPHIC o DBCLOB	DBCLOB
CHAR o VARCHAR	CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	CHAR, VARCHAR, CHAR FOR BIT DATA o VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
BLOB	BLOB	BLOB
Sólo para bases de datos Unicode:		
CHAR o VARCHAR	GRAPHIC o VARGRAPHIC	VARCHAR
GRAPHIC o VARGRAPHIC	CHAR o VARCHAR	VARGRAPHIC
CLOB	GRAPHIC, VARGRAPHIC o DBCLOB	CLOB
DBCLOB	CHAR, VARCHAR o CLOB	DBCLOB

Una *serie-fuente* puede tener una longitud de 0; en este caso, *inicio* debe ser 1 y *longitud* debe ser 0 (como lo implican los vínculos para *inicio* y *longitud* descritos más arriba) y el resultado de la función es una copia de la *serie-inserción*. Si en este caso no se especifica explícitamente la longitud, se devuelve un error porque la longitud que se presupone no es cero (SQLSTATE 22011).

Una *serie-inserción* también puede tener una longitud de 0. Esto tiene el efecto de suprimir las unidades de código de las posiciones *inicio* a *inicio* + *longitud* - 1 de la *serie-fuente*.

El atributo de longitud del resultado es el atributo de longitud de *serie-fuente* más el atributo de longitud de *serie-inserción*. La longitud real del resultado es A1 - MIN((A1 - V2 + 1), V3) + A4, donde:

- A1 es la longitud real de *serie-fuente*
- V2 es el valor de *inicio*
- V3 es el valor de *longitud*
- A4 es la longitud real de *serie-inserción*

Si la longitud real de la serie de resultado excede el máximo del tipo de datos de retorno, se devuelve un error (SQLSTATE 54006).

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Cree las series 'INSISTING', 'INSISERTING' e 'INSTING' a partir de la serie 'INSERTING' insertando texto en medio del texto existente.

```
SELECT OVERLAY('INSERTING','IS',4,2,OCTETS),
  OVERLAY('INSERTING','IS',4,0,OCTETS),
  OVERLAY('INSERTING','',4,2,OCTETS)
  FROM SYSIBM.SYSDUMMY1
```

- Cree las series 'XXINSERTING', 'XXNSERTING', 'XXSERTING' y 'XXERTING' a partir de la serie 'INSERTING' insertando texto antes del texto existente, utilizando 1 como punto de partida.

```
SELECT OVERLAY('INSERTING','XX',1,0,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,1,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,2,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,3,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- Cree la serie 'ABCABCXX' a partir de la serie 'ABCABC' insertando texto después del texto existente. Dado que la serie fuente tiene una longitud de 6 caracteres, establezca la posición inicial en 7 (uno más la longitud de la serie fuente).

```
SELECT OVERLAY('ABCABC','XX',7,0,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- Cambie la serie 'Hegelstraße' por 'Hegelstrasse'.

```
SELECT OVERLAY('Hegelstraße','ss',10,1,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- El ejemplo siguiente funciona con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	**'N'**	**'~'**	**'A'**	**'B'**
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  Supongamos que las variables UTF8_VAR y UTF16_VAR contienen las representaciones de la serie en UTF-8 y UTF-16BE respectivamente. Utilice la función OVERLAY para insertar una 'C' en la serie Unicode '&N~AB'.

```
SELECT OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 1, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'C?N~AB', 'CN~AB' y 'CbbbN~AB', respectivamente, donde '?' representa X'EDB49E', que corresponde a X'DD1E' en el formato UTF-16 intermedio, y 'bbb' sustituye los caracteres UTF-8 incompletos X'9D849E'.

```
SELECT OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '&N~CB', '&N~AC' y '&N~AB', respectivamente.

```
SELECT OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'C?N~AB' y 'CN~AB', respectivamente, donde '?' representa el carácter de sustitución bajo no coincidente U+DD1E.

```
SELECT OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '&N~CB' y '&N~AC', respectivamente.

PARAMETER

La función PARAMETER representa una posición en una sentencia de SQL en la que XQuery proporciona dinámicamente el valor como parte de la invocación de la función db2-fn:sqlquery.

El esquema es SYSIBM.

El *constante-entero* es un índice de posición para un valor de los argumentos de db2-fn:sqlquery. El valor debe estar entre 1 y el número total de argumentos especificados en la sentencia de SQL db2-fn:sqlquery SQL (SQLSTATE 42815).

La función PARAMETER representa una posición en una sentencia de SQL en la que XQuery proporciona dinámicamente el valor como parte de la invocación de la función db2-fn:sqlquery. El argumento de la función PARAMETER determina el valor que se sustituye por la función PARAMETER cuando se ejecuta la función db2-fn:sqlquery. Se puede hacer referencia al valor proporcionado por la función PARAMETER varias veces dentro de la misma sentencia de SQL.

Esta función sólo puede utilizarse en una selección completa contenida en el argumento literal de serie de la función db2-fn:sqlquery en una expresión XQuery (SQLSTATE 42887).

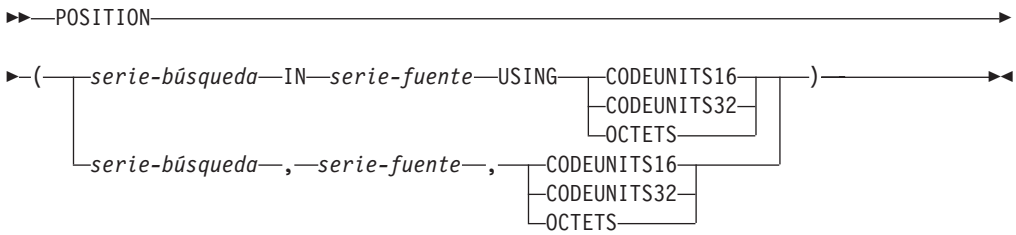
Ejemplo:

En el ejemplo siguiente, la llamada a la función db2-fn:sqlquery utiliza una llamada a la función PARAMETER y la expresión XQuery `$po/@OrderDate`, el atributo de fecha de pedido. La llamada a la función PARAMETER devuelve el valor del atributo de fecha de pedido:

```
xquery
declare default element namespace "http://posample.org";
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/PurchaseOrder,
  $item in $po/item/partid
for $p in db2-fn:sqlquery(
  "select description from product where promostart < PARAMETER(1)",
    $po/@OrderDate )
where $p//@pid = $item
return
<RESULT>
  <PoNum>{data($po/@PoNum)}</PoNum>
  <PartID>{data($item)} </PartID>
  <PoDate>{data($po/@OrderDate)}</PoDate>
</RESULT>
```

El ejemplo devuelve el ID de compra, el ID de pieza y la fecha de compra de todas las piezas vendidas a partir de la fecha de inicio de la promoción.

POSITION



El esquema es SYSIBM.

La función POSITION devuelve la posición inicial de la primera aparición de una serie (denominada *serie-búsqueda*) dentro de otra serie (denominada *serie-fuente*). Si *serie-búsqueda* no se encuentra y ninguno de los argumentos es nulo, el resultado es cero. Si se encuentra el argumento *serie-búsqueda*, el resultado es un número de 1 a la longitud real del argumento *serie-fuente*, expresado en la unidad de serie que se especifique explícitamente. La búsqueda se realiza utilizando la clasificación de la base de datos, a menos que se defina *serie-búsqueda* o *serie-fuente* como FOR BIT DATA, en cuyo caso la búsqueda se realiza utilizando una comparación binaria.

Si *serie-fuente* tiene una longitud real de 0, el resultado de la función es 0. Si *serie-búsqueda* tiene una longitud real de 0 y *serie-fuente* no es nulo, el resultado de la función es 1.

serie-búsqueda
Expresión que especifica la serie que es objeto de la búsqueda. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, BLOB, VARGRAPHIC, GRAPHIC o CHAR incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC, VARGRAPHIC o BLOB, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión no puede ser una variable de referencia a archivo BLOB. La expresión puede especificarse mediante lo siguiente:

- Una constante
- Un registro especial
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene (mediante CONCAT o `||`) cualquiera de los elementos anteriores
- Un parámetro de procedimiento de SQL

Estas normas son parecidas a las que se describen para *expresión-patrón* para el predicado LIKE.

serie-fuente
La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión puede especificarse mediante lo siguiente:

- Una constante
- Un registro especial
- Una variable del lenguaje principal (incluida una variable localizadora o una variable de referencia de archivo)

- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie del resultado. CODEUNITS16 especifica que el resultado debe expresarse en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que el resultado debe expresarse en unidades de código UTF-32 de 32 bits. OCTETS especifica que el resultado debe expresarse en bytes.

Si una unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32, y *serie-búsqueda* o *serie-fuente* son una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si se especifica una unidad de la serie como OCTETS y *serie-búsqueda* y *serie-fuente* son series binarias, de devuelve un error (SQLSTATE 42815).

Si se utiliza una clasificación basada en la UCA sensible a la configuración local para esta función, la opción CODEUNITS16 ofrece las mejores características de rendimiento.

Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El primer y segundo argumento deben tener tipos de serie compatibles. Para obtener más información sobre la compatibilidad, consulte el apartado "Normas para conversiones de series". En una base de datos Unicode, si un argumento de serie es de tipo carácter (no FOR BIT DATA) y el otro argumento de serie es gráfico, la *serie-búsqueda* se convierte al tipo de datos de la *serie-fuente* para el proceso. Si un argumento es de tipo carácter FOR BIT DATA, el otro argumento no debe ser gráfico (SQLSTATE 42846).

El resultado de la función es un entero grande. Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Seleccione la columna RECEIVED, la columna SUBJECT y la posición de inicio de la serie 'GOOD BEER' en la columna NOTE_TEXT para todas las filas de la tabla IN_TRAY que contengan esa serie.

```
SELECT RECEIVED, SUBJECT, POSITION('GOOD BEER', NOTE_TEXT, OCTETS)
FROM IN_TRAY
WHERE POSITION('GOOD BEER', NOTE_TEXT, OCTETS) <> 0
```

- Localizar el carácter 'ß' en la serie 'Jürgen lives on Hegelstraße' y definir la variable del lenguaje principal LOCATION con la posición, según se mide en unidades CODEUNITS32, dentro de la serie.

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', CODEUNITS32
)
```

  El valor de la variable del lenguaje principal LOCATION se establece en 26.

- Localizar el carácter 'ß' en la serie 'Jürgen lives on Hegelstraße' y definir la variable del lenguaje principal LOCATION con la posición, según se mide en unidades OCTETS, dentro de la serie.

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', OCTETS
)
```

El valor de la variable del lenguaje principal LOCATION se establece en 27.

- Los ejemplos siguientes funcionan con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación sin avance de espacio. A continuación se muestra esta serie en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

Supongamos que la variable UTF8_VAR contiene la representación UTF-8 de la serie.

```
SELECT POSITION('N', UTF8_VAR, CODEUNITS16),
  POSITION('N', UTF8_VAR, CODEUNITS32),
  POSITION('N', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 3, 2 y 5, respectivamente.

Supongamos que la variable UTF16_VAR contiene la representación UTF-16BE de la serie.

```
SELECT POSITION('B', UTF16_VAR, CODEUNITS16),
   POSITION('B', UTF16_VAR, CODEUNITS32),
   POSITION('B', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 6, 5 y 11, respectivamente.

- En una base de datos Unicode creada con la clasificación no sensible a las mayúsculas y UCA500R1_LEN_S1, busque la posición de la palabra 'Brown' en la frase 'The quick brown fox'.

```
SET
:LOCATION = POSITION('Brown', 'The quick brown fox', CODEUNITS16)
```

El valor de la variable del lenguaje principal LOCATION se establece en 11.

POSSTR

El esquema es SYSIBM.

La función POSSTR devuelve la posición inicial de la primera aparición de una serie (llamada la *serie-búsqueda*) en otra serie (llamada la *serie-fuente*). Los números para la posición de *serie-búsqueda* empiezan en 1 (no en 0).

El resultado de la función es un entero grande. Si alguno de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor nulo.

serie-fuente
Una expresión que especifica la serie fuente en la que se ha de llevar a cabo la búsqueda.

La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal (incluida una variable localizadora o una variable de referencia de archivo)
- Una función escalar
- Un localizador de objeto grande
- Un nombre de columna
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores

serie-búsqueda
Una expresión que especifica la serie que se ha de buscar.

La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, BLOB, VARGRAPHIC, GRAPHIC o CHAR incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC, VARGRAPHIC o BLOB, se convierte implícitamente a VARCHAR antes de evaluar la función. La longitud real no debe superar la longitud máxima de un tipo de datos VARCHAR. La expresión no puede ser una variable de referencia a archivo BLOB. La expresión se puede especificar mediante:

- Una constante
- Un registro especial
- Una variable global
- Una variable del lenguaje principal
- Una función escalar cuyos operandos sean cualquiera de los anteriores
- Una expresión que concatene (mediante CONCAT o ||) cualquiera de los elementos anteriores
- Un parámetro de procedimiento de SQL

A continuación se muestran ejemplos de expresiones de serie o series válidas:

- Parámetros de función SQL definidos por el usuario
- Variables de transición activadores
- Variables locales en sentencias compuestas dinámicas

En una base de datos Unicode, si un argumento es de tipo carácter (no FOR BIT DATA) y el otro argumento es gráfico, la *serie-búsqueda* se convierta al tipo de datos de la *serie-fuente* para el proceso. Si un argumento es de tipo carácter FOR BIT DATA, el otro argumento no debe ser gráfico (SQLSTATE 42846).

Tanto la *serie-búsqueda* como la *serie-fuente* tienen cero o más posiciones continuas. Si las series son series de caracteres o binarias, una posición es un byte. Si las series son series gráficas, una posición es un carácter gráfico (DBCS).

La función POSSTR acepta las series de datos mixtos. No obstante, POSSTR funciona según un recuento de bytes estricto, ajeno a la clasificación de base de datos y a los cambios entre caracteres de un byte y de varios bytes.

Se aplican las siguientes normas:

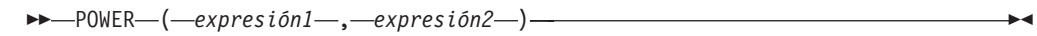
- Los tipos de datos de la *serie-fuente* y la *serie-búsqueda* deben ser compatibles, de lo contrario se genera un error (SQLSTATE 42884).
  - Si *serie-fuente* es una serie de caracteres, *serie-búsqueda* debe ser una serie de caracteres, pero no un CLOB, con una longitud real de 32.672 bytes como máximo.
  - Si *serie-fuente* es una serie gráfica, *serie-búsqueda* debe ser una serie gráfica, pero no un DBCLOB, con una longitud real de 16.336 caracteres de doble byte como máximo.
  - Si *serie-fuente* es una serie binaria, *serie-búsqueda* debe ser una serie binaria con una longitud real de 32.672 bytes o menos.
- Si la *serie-búsqueda* tiene una longitud de cero, el resultado que devuelve la función es 1.
- En otro caso:
  - Si la *serie-fuente* tiene una longitud de cero, el resultado que devuelve la función es cero.
  - En otro caso:
    - Si el valor de la *serie-búsqueda* es igual a una subserie de longitud idéntica de posiciones continuas del valor de la *serie-fuente*, el resultado devuelto por la función es la posición inicial de la primera de dicha subserie en el valor *serie-fuente*.
    - De lo contrario, el resultado que devuelve la función es 0.

Ejemplo

- Seleccione las columnas RECEIVED y SUBJECT así como la posición inicial de las palabras 'GOOD BEER' de la columna NOTE_TEXT para todas las entradas de la tabla IN_TRAY que contienen estas palabras.

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER



El esquema es SYSIBM. (La versión SYSFUN de la función POWER continúa estando disponible.)

La función POWER devuelve el resultado de elevar el primer argumento a la potencia del segundo argumento.

Los argumentos pueden ser de cualquier tipo de datos numérico interno. Los argumentos DECIMAL y REAL se convierten a un número de coma flotante de precisión doble. Si uno de los dos argumentos es una coma flotante decimal, los argumentos se convierten a DECFLOAT(34) para que los procese la función.

El resultado de la función es:

- INTEGER si ambos argumentos son INTEGER o SMALLINT
- BIGINT si un argumento es BIGINT y el otro argumento es BIGINT, INTEGER o SMALLINT
- DECFLOAT(34) si uno de los argumentos es una coma flotante decimal. Si uno de los dos argumentos es DECFLOAT y se cumple como verdadero una de las siguientes sentencias, el resultado será NAN y una condición de operación no válida:
  - Ambos argumentos son cero
  - El segundo argumento tiene una parte fraccionaria distinta de cero
  - El segundo argumento tiene más de 9 dígitos
  - El segundo argumento es INFINITY
- DOUBLE en caso contrario

Si el argumento es un valor de coma flotante decimal especial, se aplicarán las normas para las operaciones aritméticas generales para la coma flotante decimal. Consulte el apartado "General arithmetic operation rules for decimal floating-point" (Operaciones aritméticas generales para coma flotante decimal) en "Normas generales de operaciones aritméticas para coma flotante decimal" en la página 236.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Suponga que la variable del lenguaje principal HPOWER es un entero con un valor de 3.

  ```
  VALUES POWER(2,:HPOWER)
  ```

  Devuelve el valor 8.

QUANTIZE

El esquema es SYSIBM.

La función QUANTIZE devuelve un valor de coma flotante decimal que equivale en valor (excepto para el redondeo) y signo con *expresión-numérica* y que tiene un exponente igual al exponente de *expresión-exp*. El número de dígitos (16 ó 34) es igual al número de dígitos de *expresión-numérica*.

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es un valor de coma flotante decimal, se convertirá a DECFLOAT(34) para su proceso.

expresión-exp
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es un valor de coma flotante decimal, se convertirá a DECFLOAT(34) para su proceso. La *expresión-exp* se usa como patrón de ejemplo para volver a dar escala a *expresión-numérica*. Se ignora el signo y el coeficiente de *expresión-exp*.

El coeficiente del resultado deriva del de *expresión-numérica*. Se redondeo, si es necesario (si el exponente se está aumentando), multiplicado por una potencia de diez (si el exponente se está disminuyendo), o permanece inalterado (si el exponente es ya igual al de *expresión-exp*).

El registro especial CURRENT DECFLOAT ROUNDING MODE determina la modalidad de redondeo.

Al contrario que otras operaciones aritméticas del tipo de datos de coma flotante decimal, en el caso de que la longitud del coeficiente posterior a la operación de quantize sea superior a la precisión especificada por *expresión-exp*, el resultado será NaN y se devolverá un aviso (SQLSTATE 0168D). Esta acción asegurará que, a menos que haya una condición de aviso, el exponente del resultado de QUANTIZE siempre será igual al de *expresión-exp*.

- Si cualquiera de los argumentos es NaN, se devolverá NaN
- Si cualquiera de los argumentos es sNaN, se devolverá NaN y se devolverá un aviso (SQLSTATE 01565)
- Si ambos argumentos son infinito (positivo o negativo), se devolverá infinito con el mismo signo que el primer argumento
- Si un argumento es infinito (positivo o negativo) y el otro argumento no es infinito se devolverá NaN y se devolverá un aviso (SQLSTATE 0168D)

El resultado de la función es un valor de DECFLOAT(16) si ambos argumentos son DECFLOAT(16). En caso contrario, el resultado de la función es un valor DECFLOAT(34). El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Los ejemplos siguientes muestran los valores devueltos por la función QUANTIZE, si se dan varios valores coma flotante decimal de entrada y se supone una modalidad de redondeo de ROUND_HALF_UP:

```
QUANTIZE(2.17, DECFLOAT(0.001)) = 2.170
QUANTIZE(2.17, DECFLOAT(0.01)) = 2.17
QUANTIZE(2.17, DECFLOAT(0.1)) = 2.2
QUANTIZE(2.17, DECFLOAT('1E+0')) = 2
QUANTIZE(2.17, DECFLOAT('1E+1')) = 0E+1
QUANTIZE(2, DECFLOAT(INFINITY)) = NaN    -- aviso
QUANTIZE(0, DECFLOAT('1E+5')) = 0E+5
QUANTIZE(217, DECFLOAT('1E-1')) = 217.0
QUANTIZE(217, DECFLOAT('1E+0')) = 217
QUANTIZE(217, DECFLOAT('1E+1')) = 2.2E+2
QUANTIZE(217, DECFLOAT('1E+2')) = 2E+2
```

- En el ejemplo siguiente, se devuelve el valor -0 para la función QUANTIZE. Se utiliza la función CHAR para evitar la potencial eliminación del signo menos por parte de un programa cliente:

```
CHAR(QUANTIZE(-0.1, DECFLOAT(1))) = -0
```

QUARTER

El esquema es SYSFUN.

Devuelve un valor entero comprendido entre 1 y 4 que representa el trimestre del año para la fecha especificada en el argumento.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica (excepto DBCLOB), se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

RADIANS

El esquema es SYSIBM. (La versión SYSFUN de la función RADIANS continúa estando disponible.)

La función RADIANS devuelve el número de radianes de un argumento que se expresa en grados.

El argumento puede ser de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

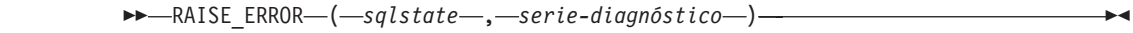
Ejemplo:

- Suponga que la variable del lenguaje principal HDEG es INTEGER con un valor de 180. La siguiente sentencia:

  ```
  VALUES RADIANS(:HDEG)
  ```

  Devuelve el valor +3.14159265358979E+000.

RAISE_ERROR



El esquema es SYSIBM.

La función RAISE_ERROR hace que la sentencia que incluye la función devuelva un error especificando SQLSTATE, SQLCODE -438 y la *serie-diagnóstico*. La función RAISE_ERROR siempre devuelve NULL con un tipo de datos no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

sqlstate
Una serie de caracteres que contiene 5 bytes exactamente. Debe ser de tipo CHAR definido con una longitud de 5 o un tipo VARCHAR definido con una longitud de 5 o mayor. El valor de *sqlstate* debe seguir las normas de los SQLSTATE definidos por la aplicación, de la siguiente manera:

- Cada carácter debe pertenecer al conjunto de dígitos (del '0' al '9') o de letras mayúsculas no acentuadas (de la 'A' a la 'Z')
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', '01' ni '02', pues no son clases de error.
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 0-6 o A-H, la subclase (tres últimos caracteres) debe empezar por una letra del rango I-Z
- Si la clase SQLSTATE (primeros dos caracteres) empieza por un carácter de los rangos 7-9 o I-Z, la subclase (tres últimos caracteres) puede ser un valor cualquiera de 0-9 o A-Z.

Si SQLSTATE no se ajusta a estas normas se produce un error (SQLSTATE 428B3).

serie-diagnóstico
Una expresión de tipo CHAR o VARCHAR que devuelve una serie de caracteres de un máximo de 70 bytes que describe la condición de error. Si la serie tiene más de 70 bytes de longitud, se truncará.

Para utilizar esta función en un contexto en el que no se puede determinar el tipo de datos, debe utilizarse una especificación cast para otorgar un tipo de datos al valor nulo devuelto. La función RAISE_ERROR resultará más útil en una expresión CASE.

Ejemplo:

Liste los números de empleados y los niveles de formación como, por ejemplo, Postgraduado, Graduado y Diplomado. Si el nivel de formación es mayor que 20, genere un error.

```
SELECT EMPNO,
  CASE WHEN EDUCLVL < 16 THEN 'Diplomado'
       WHEN EDUCLVL < 18 THEN 'Graduado'
       WHEN EDUCLVL < 21 THEN 'Postgraduado'
       ELSE RAISE_ERROR('70001',
         'EDUCLVL tiene un valor mayor que 20')
  END
FROM EMPLOYEE
```

RAND

El esquema es SYSFUN.

La función RAND devuelve un valor de coma flotante comprendido entre 0 y 1.

Si se especifica una expresión, ésta se utiliza como el valor raíz. La expresión debe ser un tipo de datos SMALLINT o INTEGER incorporado con un valor entre 0 y 2147483647.

El tipo de datos del resultado es de coma flotante de precisión doble. Si el argumento es nulo, el resultado es el valor nulo.

Un valor raíz concreto generará la misma secuencia de números aleatorios para una instancia determinada de una función RAND de una consulta cada vez que se ejecute la consulta. Si no se especifica ningún valor raíz, se genera una secuencia de números aleatorios distinta cada vez que se ejecuta la consulta en la misma sesión. Para generar un conjunto de números aleatorios que varíe de una sesión a otra, debe especificarse un valor raíz aleatorio como, por ejemplo, uno que se base en la hora actual.

RAND es una función no determinante.

REAL

De numérico a real

De serie a real

El esquema es SYSIBM.

La función REAL devuelve una representación de coma flotante de precisión única de:

- Un número
- Una representación de serie de un número

De numérico a real

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado es el mismo número que se generaría si el argumento se hubiese asignado a una columna o variable de coma flotante de precisión simple. Si el valor numérico del argumento no está dentro del rango de la coma flotante de precisión única, se devuelve un error (SQLSTATE 22003).

De serie a real

expresión-serie
Expresión que devuelve un valor que es una serie de caracteres o la representación de serie gráfica Unicode de un número. El tipo de datos de expresión-serie no debe ser CLOB ni DBCLOB (SQLSTATE 42884).

El resultado es el mismo número que el que generaría CAST(expresión-serie AS REAL). Los espacios en blanco iniciales y finales se eliminan y la serie resultante debe ajustarse a las normas para formar una constante numérica válida (SQLSTATE 22018). Si el valor numérico del argumento no está dentro del rango de la coma flotante de precisión única, se devuelve un error (SQLSTATE 22003).

El resultado de la función es un número de coma flotante de precisión simple. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".

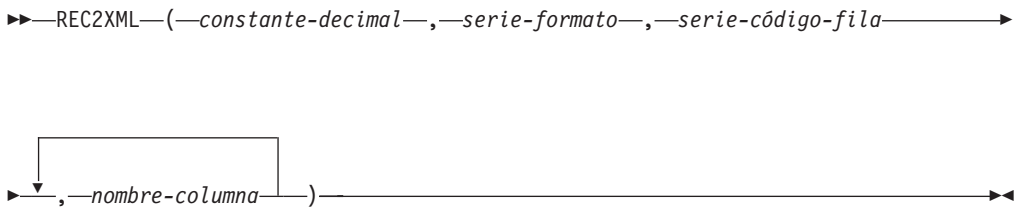
Ejemplo:

Utilizando la tabla EMPLOYEE, busque la proporción de salario y comisiones para los empleados cuya comisión no sea cero. Las columnas implicadas (SALARY y COMM) tienen tipos de datos DECIMAL. El resultado se desea en coma flotante de precisión simple. Por consiguiente, se aplica REAL a SALARY para que la

división se lleve a cabo en coma flotante (en realidad en precisión doble) y después se aplica REAL a la expresión completa para devolver el resultado en coma flotante de precisión simple.

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
  FROM EMPLOYEE
  WHERE COMM > 0
```

REC2XML



El esquema es SYSIBM.

La función REC2XML devuelve una serie formateada codificada en XML que contiene nombres de columna y datos de columna. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

constante-decimal
Factor de expansión para sustituir caracteres de datos de columna. El valor decimal debe ser mayor que 0.0 y menor que o igual a 6.0. (SQLSTATE 42820).

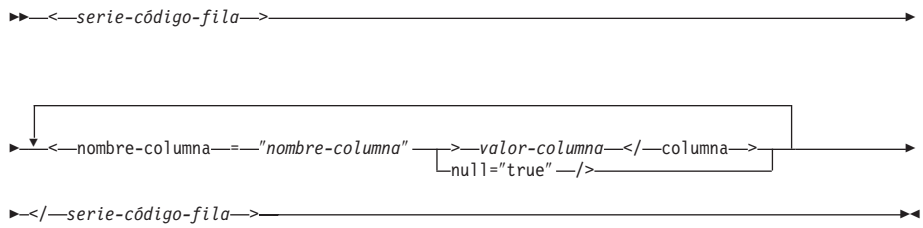
El valor *constante-decimal* se utiliza para calcular la longitud de resultado de la función. Para cada columna con un tipo de datos de tipo carácter, el atributo de longitud de la columna se multiplica por este factor de expansión antes de que se sume a la longitud de resultado.

Para especificar que no haya expansión, utilice un valor de 1.0. Si se especifica un valor menor que 1.0, se reducirá la longitud de resultado calculada. Si la longitud real de la serie de resultado es mayor que la longitud de resultado calculada de la función, se producirá un error (SQLSTATE 22001).

serie-formato
Constante de serie que especifica qué formato debe utilizar la función durante la ejecución.

La *serie-formato* es sensible a las mayúsculas y minúsculas, de modo que los valores siguientes deben especificarse en mayúsculas para que se reconozcan.

COLATTVAL o COLATTVAL_XML
Estos formatos devuelven una serie con columnas como valores de atributo.



Los nombres de columna pueden ser valores de atributo XML válidos o pueden no serlo. Para los nombres de columna que no son valores de atributo XML válidos, se realiza la sustitución de caracteres en el nombre de columna antes de incluirlo en la serie de resultado.

Los valores de columna pueden ser nombres de elemento XML válidos o pueden no serlo. Si se especifica la *serie-formato* COLATTVAL, para los nombres de columna que no son valores de elemento XML válidos, se realiza la sustitución de caracteres en el valor de columna antes de incluirlo en la serie de resultado. Si se especifica la *serie-formato* COLATTVAL_XML, no se realiza la sustitución de caracteres en los valores de columna (aunque la sustitución de caracteres se realiza de todas formas en los nombres de columna).

serie-código-fila
Constante de serie que especifica el código utilizado para cada fila. Si se especifica una serie vacía, se supone un valor de 'row'.

Si se especifica una serie de uno o más caracteres en blanco, no aparecerá ninguna *serie-código-fila* inicial o *serie-código-fila* final (incluidos los delimitadores de signo mayor que y menor que) en la serie de resultado.

nombre-columna
Nombre calificado o no calificado de una columna de tabla. La columna debe tener uno de los tipos de datos siguientes (SQLSTATE 42815):

- numérico (SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE)
- serie de caracteres (CHAR, VARCHAR; no se permite una serie de caracteres con un subtipo de BIT DATA)
- fecha y hora (DATE, TIME, TIMESTAMP)
- un tipo definido por el usuario basado en uno de los tipos anteriores

No se puede especificar más de una vez el mismo nombre de columna (SQLSTATE 42734).

El resultado de la función es VARCHAR. La longitud máxima es de 32.672 bytes (SQLSTATE 54006).

Examine la invocación siguiente:

```
REC2XML (dc, fs, rt, c1, c2, ..., cn)
```

Si el valor de ″fs″ es ″COLATTVAL″ o ″COLATTVAL_XML″, el resultado será igual que esta expresión:

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

donde y_n es equivalente a:

```
'<column name="' CONCAT xvcn CONCAT vn
```

y vn es equivalente a:

```
'">' CONCAT rn CONCAT '</column>'
```

si la columna no es nula y

```
'" null="true"/>'
```

si el valor de columna es nulo.

xvc_n es equivalente a una representación de serie del nombre de columna de c_n, donde cualquier carácter que aparezca en la Tabla 55 en la página 541 se sustituye por la representación correspondiente. Esto asegura que la serie resultante sea un símbolo de valor de elemento o atributo XML válido.

r_n es equivalente a una representación de serie como se indican en la Tabla 54

Tabla 54. Resultado de serie de valores de columna

Tipo de datos de c_n	**r_n**
CHAR, VARCHAR	El valor es una serie. Si la *serie-formato* no termina con los caracteres "_XML", cada carácter de c_n se sustituye por la representación de sustitución correspondiente de la Tabla 55, como se indica. El atributo de longitud es: dc * el atributo de longitud de c_n.
SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE	El valor es LTRIM(RTRIM(CHAR(c_n))). El atributo de longitud es la longitud de resultado de CHAR(c_n). El carácter decimal es siempre el carácter de punto ('.').
DATE	El valor es CHAR(c_n,ISO). El atributo de longitud es la longitud de resultado de CHAR(c_n,ISO).
TIME	El valor es CHAR(c_n,JIS). El atributo de longitud es la longitud de resultado de CHAR(c_n,JIS)
TIMESTAMP	El valor es CHAR(c_n). El atributo de longitud es la longitud de resultado de CHAR(c_n).

Sustitución de caracteres:

En función del valor especificado para la *serie-formato*, se sustituirán determinados caracteres en los nombres de columna y en los valores de columna para asegurar que los nombres de columna formen valores de atributo XML válidos y que los valores de columna formen valores de elemento XML válidos.

Tabla 55. Sustituciones de caracteres para valores de atributo y valores de elemento XML

Carácter	**Sustitución**
<	&lt;
>	&gt;
"	&quot;
&	&amp;
'	&apos;

Ejemplos:

Nota: REC2XML no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Utilizando la tabla DEPARTMENT de la base de datos de ejemplo, formatee la fila de tabla del departamento, excepto las columnas DEPTNAME y LOCATION, para el departamento 'D01' en una serie XML. Dado que los datos no contienen ninguno de los caracteres que necesitan sustituirse, el factor de expansión será 1.0 (sin expansión). Observe también que el valor MGRNO es nulo para esta fila.

```
SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
  FROM DEPARTMENT
  WHERE DEPTNO = 'D01'
```

Este ejemplo devuelve la serie VARCHAR(117) siguiente:

```
<row>
<column name="DEPTNO">D01</column>
<column name="MGRNO" null="true"/>
<column name="ADMRDEPT">A00</column>
</fila>
```

- Una planificación universitaria de 5 días introduce una clase llamada '&43<FIE' en una tabla llamada CL_SCHED, con un formato nuevo para la columna CLASS_CODE. Utilizando la función REC2XML, este ejemplo formatea una serie XML con estos datos de clase nuevos, exceptuando la hora final de clase.

  El atributo de longitud para la llamada REC2XML (vea más abajo) con un factor de expansión de 1.0 será 128 (11 para la actividad general de '<row>' y '</row>', 21 para los nombres de columna, 75 para '<column name=', '>', '</column>' y las comillas dobles, 7 para los datos CLASS_CODE, 6 para los datos DAY y 8 para los datos STARTING). Dado que los caracteres '&' y '<' se sustituirán, no será suficiente un factor de expansión de 1.0. El atributo de longitud de la función necesitará soportar un incremento de 7 a 14 bytes para los datos CLASS_CODE de formato nuevo.

  Sin embargo, puesto que se sabe que el valor DAY no tendrá nunca más de 1 dígito de longitud, se añaden al total 5 unidades adicionales no utilizadas de longitud. Por lo tanto, la expansión sólo necesita gestionar un incremento de 2. Dado que CLASS_CODE es la única columna de serie de caracteres de la lista de argumentos, éstos son los únicos datos de columna a los que se aplica el factor de expansión. Para obtener un incremento de 2 para la longitud, sería necesario un factor de expansión de 9/7 (1.2857 aproximadamente). Se utilizará un factor de expansión de 1.3.

  ```
  SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
    FROM CL_SCHED
    WHERE CLASS_CODE = '&43<FIE'
  ```

  Este ejemplo devuelve la serie VARCHAR(167) siguiente:

  ```
  <record>
  <column name="CLASS_CODE">&amp;43&lt;FIE</column>
  <column name="DAY">5</column>
  <column name="STARTING">06:45:00</column>
  </record>
  ```

- Supongamos que se han añadido filas nuevas a la tabla EMP_RESUME de la base de datos de ejemplo. Las nuevas filas almacenan los resúmenes como series de XML válido. Se utiliza la *serie-formato* COLATTVAL_XML para que no se lleve a cabo la sustitución de caracteres. Ninguno de los resúmenes tiene más de 3500 bytes de longitud. Se utiliza la consulta siguiente para seleccionar la versión XML de los resúmenes de la tabla EMP_RESUME y formatearla en un fragmento de documento XML.

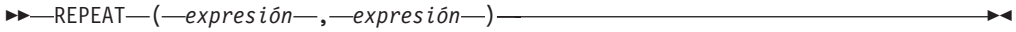
  ```
  SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
    FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
      FROM EMP_RESUME
      WHERE RESUME_FORMAT = 'XML')
    AS EMP_RESUME_XML
  ```

  Este ejemplo devuelve una fila para cada empleado que tiene un resumen en formato XML. Cada fila devuelta será una serie con el formato siguiente:

```
<row>
<column name="EMPNO">{número de empleado}</column>
<column name="RESUME_XML">{resumen en XML}</column>
</row>
```

Donde ″{número de empleado}″ es el valor EMPNO real para la columna y ″{resumen en XML}″ es el valor de serie de fragmento XML real que constituye el resumen.

REPEAT



El esquema es SYSFUN.

Devuelve una serie de caracteres compuesta por el primer argumento repetido el número de veces especificado por el segundo argumento. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El primer argumento es una serie de caracteres o un tipo de serie binaria. Para un VARCHAR, la longitud máxima es de 4.000 bytes y para un CLOB o una serie binaria, la longitud máxima es de 1.048.576 bytes. El segundo argumento puede ser SMALLINT o INTEGER.

El resultado de la función es:

- VARCHAR(4000) si el primer argumento es VARCHAR (no excede de 4.000 bytes) o CHAR
- CLOB(1M) si el primer argumento es CLOB.
- BLOB(1 M) si el primer argumento es BLOB.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Liste la frase 'REPITA ESTO' cinco veces.

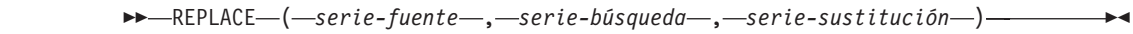
```
VALUES CHAR(REPEAT('REPEAT THIS', 5), 60)
```

  Este ejemplo produce lo siguiente:

```
1
------------------------------------------------------------
REPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTOREPITA ESTO
```

  Tal como se ha mencionado, el resultado de la función REPEAT es VARCHAR(4000). En este ejemplo, se ha utilizado la función CHAR para limitar el resultado de REPEAT a 60 bytes.

REPLACE



El esquema es SYSIBM. La versión SYSFUN de la función REPLACE continúa estando disponible pero no es sensible a la clasificación de base de datos.

Sustituye todas las ocurrencias de *serie-búsqueda* en *serie-fuente* por *serie-sustitución*. Si no se encuentra *serie-búsqueda* en *serie-fuente*, se devuelve *serie-búsqueda* sin modificar. La búsqueda se realiza utilizando la clasificación de la base de datos a menos que *serie-fuente*, *serie-búsqueda* o *serie-sustitución* se haya definido como FOR BIT DATA, en cuyo caso la búsqueda se realiza utilizando una comparación binaria.

serie-fuente
Expresión que especifica la serie fuente. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

serie-búsqueda
Expresión que especifica la serie que se debe eliminar de la serie fuente. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

serie-sustitución
Expresión que especifica la serie de sustitución. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función. Si la expresión es una serie vacía, la serie que se elimina de la serie fuente no se sustituye por nada.

La longitud real de cada serie debe ser de 32.672 bytes o menos para series de caracteres o de 16.336 o menos para series gráficas. Los tres argumentos deben tener tipos de datos compatibles.

Si *serie-fuente*, *serie-búsqueda* o *serie-sustitución* se ha definido como FOR BIT DATA, el resultado es VARCHAR FOR BIT DATA. Si *serie-fuente* es una serie de caracteres, el resultado es VARCHAR. Si *serie-fuente* es una serie gráfica, el resultado es VARGRAPHIC. Si un argumento es de tipo carácter FOR BIT DATA, los demás argumentos no deben ser gráficos (SQLSTATE 42846).

El atributo de longitud del resultado depende de los argumentos:

- Si el atributo de longitud de *serie-sustitución* es menor que o igual al atributo de longitud *serie-búsqueda*, el atributo de longitud del resultado es el atributo de longitud de *serie-fuente*.
- Si el atributo de longitud de *serie-sustitución* es mayor que el atributo de longitud de *serie-búsqueda*, el atributo de longitud del resultado se determina del modo siguiente, en función del tipo de datos del resultado:
  - Para VARCHAR:

- Si L1 < = 4000, el atributo de longitud del resultado es `MIN(4000, (L3*(L1/L2)) + MOD(L1,L2))`
- De lo contrario, el atributo de longitud del resultado es `MIN(32672, (L3*(L1/L2)) + MOD(L1,L2))`

– Para VARGRAPHIC:

- Si L1 < = 2000, el atributo de longitud del resultado es `MIN(2000, (L3*(L1/L2)) + MOD(L1,L2))`
- De lo contrario, el atributo de longitud del resultado es `MIN(16336, (L3*(L1/L2)) + MOD(L1,L2))`

donde:

– L1 es el atributo de longitud de *serie-fuente*

– L2 es el atributo de longitud de la *serie-búsqueda* si la serie de búsqueda es una constante de serie. De lo contrario, L2 es 1.

– L3 es el atributo de longitud de *serie-sustitución*

Si el resultado es una serie de caracteres, el atributo de longitud del resultado no debe exceder de 32.672. Si el resultado es una serie gráfica, el atributo de longitud del resultado no debe exceder de 16.336.

La longitud real del resultado es la longitud real de la *serie-fuente* más el número de apariciones de *serie-búsqueda* que existen en la *serie-fuente* multiplicado por la longitud real de *serie-sustitución* menos la longitud real de la *serie-búsqueda*.

Si la longitud real de la *serie-sustitución* excede el máximo del tipo de datos de retorno, se devuelve un error. Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Sustituya todas las apariciones de la letra 'N' en la palabra 'DINING' por 'VID'.

  ```
  VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
  ```

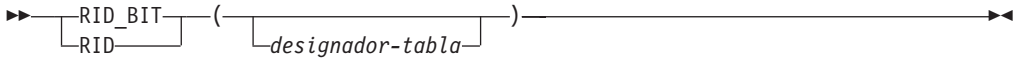
  El resultado es la serie 'DIVIDIVIDG'.

- En una base de datos Unicode con clasificación no sensible a las mayúsculas y minúsculas UCA500R1_LEN_S1, sustituya la palabra 'QUICK' por la palabra 'LARGE'.

  ```
  VALUES REPLACE
  ('The quick brown fox', 'QUICK', 'LARGE')
  ```

  El resultado es la serie 'The LARGE brown fox'.

RID_BIT y RID



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

Las funciones RID_BIT y RID devuelven el identificador de fila (RID) de una fila en formatos diferentes. El RID se utiliza para identificar de manera exclusiva una fila. Es posible que cada función devuelva valores diferentes cuando se invoca varias veces para una fila. Por ejemplo, después de ejecutar el programa de utilidad reorg para una tabla, es posible que las funciones RID_BIT y RID devuelvan valores de una fila diferentes de los que se habrían devuelto antes de ejecutar el programa de utilidad. Las funciones RID_BIT y RID no son determinantes. El resultado de la función RID_BIT, a diferencia de la función RID, contiene información de tabla para evitar usarla involuntariamente con una tabla diferente. No se da soporte a la función RID en un entorno de bases de datos particionadas.

designador-tabla
Identifica de manera exclusiva una tabla base, una vista o una expresión de tabla anidada (SQLSTATE 42703). Si *designador-tabla* especifica una vista o una expresión de tabla anidada, las funciones RID_BIT y RID devuelven el RID de la tabla base de la vista o expresión de tabla anidada. La vista o expresión de tabla anidada especificada debe contener sólo una tabla base en su subselección exterior (SQLSTATE 42703). El *designador-tabla* debe ser suprimible (SQLSTATE 42703). Para obtener más información sobre vistas suprimibles, consulte el apartado “Notas” o “CREATE VIEW”.

Si no se especifica el *designador-tabla*, la cláusula FROM debe contener únicamente un elemento que se puede calcular para que sea el designador de tabla (SQL STATE 42703).

El resultado de la función RID_BIT es VARCHAR (16) FOR BIT DATA. El resultado puede ser nulo. El resultado de la función RID es BIGINT. El resultado puede ser nulo.

Notas:

- Para implementar un bloqueo optimista en una aplicación, utilice los valores devueltos por la expresión ROW CHANGE TOKEN como argumentos para la función escalar RID_BIT.
- **Compatibilidades:** se admite la sintaxis siguiente, pero no es estándar y no debería utilizarse:
  - Se puede utilizar la pseudocolumna “ROWID” para hacer referencia a RID. Una referencia ROWID no calificada equivale a RID_BIT() y un ROWID calificado como EMPLOYEE.ROWID equivale a RID_BIT(EMPLOYEE).

Ejemplos:

- Devolver el RID y el apellido de todos los trabajadores del departamento 20 a partir de la tabla EMPLOYEE.

```
SELECT RID_BIT (EMPLOYEE), ROW CHANGE TOKEN FOR EMPLOYEE, LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = '20'
```

- A partir de la tabla EMP1, que se define de la manera siguiente:

```
CREATE TABLE EMP1 (
  EMPNO   CHAR(6),
  NAME    CHAR(30),
  SALARY       DECIMAL(9,2),
  PICTURE BLOB(250K),
  RESUME  CLOB(32K)
)
```

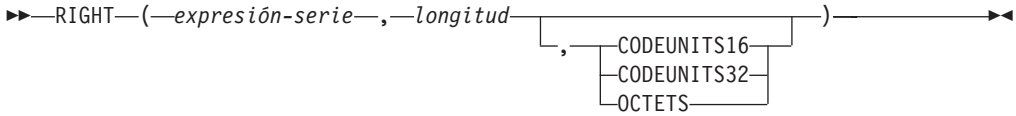
Establecer la variable del lenguaje principal HV_EMP_RID en el valor de la función escalar incorporada RID_BIT y HV_EMP_RCT en el valor de la expresión ROW CHANGE TOKEN para la fila correspondiente al trabajador número 3500.

```
SELECT RID_BIT(EMP1), ROW CHANGE TOKEN FOR EMP1
  INTO :HV_EMP_RID, :HV_EMP_RCT FROM EMP1
  WHERE EMPNO = '3500'
```

Utilizando el valor RID para identificar al trabajador y la función definida por el usuario UPDATE_RESUME, aumentar el salario del trabajador en 1.000€ y actualizar el currículum del trabajador.

```
UPDATE EMP1 SET
  SALARY = SALARY + 1000,
  RESUME = UPDATE_RESUME(:HV_RESUME)
WHERE RID_BIT(EMP1) = :HV_EMP_RID
AND ROW CHANGE TOKEN FOR EMP1 = :HV_EMP_RCT
```

RIGHT



El esquema es SYSIBM. La versión SYSFUN de la función RIGHT continúa estando disponible.

La función RIGHT devuelve la serie situada más a la derecha de *expresión-serie* de longitud *longitud*, expresado en la unidad de serie especificada. Si *expresión-serie* es una serie de caracteres, el resultado es una serie de caracteres. Si *expresión-serie* es una serie gráfica, el resultado es una serie gráfica

expresión-serie
Una expresión que especifica la serie de la que se deriva el resultado. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. Una subserie de *expresión-serie* es cero o más elementos de código contiguos de *expresión-serie*.

longitud
Una expresión que especifica la longitud del resultado. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor debe estar entre 0 y la longitud de *expresión-serie*, expresado en unidades que se especifiquen implícita o explícitamente (SQLSTATE 22011). Si se especifica OCTETS y el resultado son datos gráficos, el valor debe ser un número par entre 0 y el doble del atributo de longitud de *expresión-serie* (SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de serie de *longitud*.

CODEUNITS16 especifica que *longitud* se expresa en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *longitud* se expresa en unidades de código UTF-32 de 32 bits. OCTETS especifica que *longitud* se expresa en bytes.

Si la unidad de serie se especifica como CODEUNITS16 o CODEUNITS32 y *expresión-serie* es una serie binaria o datos de bit, se devuelve un error (SQLSTATE 428GC). Si la unidad de serie se especifica como OCTETS y *expresión-serie* es una serie gráfica, *longitud* debe ser un número par; de lo contrario, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *longitud* se expresa en unidades de dos bytes; de lo contrario, se expresa en bytes. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado “Unidades de serie en funciones incorporadas” en “Series de caracteres”.

La *expresión-serie* se rellena a la derecha con el número necesario de caracteres de relleno para que exista siempre la subserie especificada de *expresión-serie*. El carácter utilizado para el relleno es el mismo carácter que se utiliza para rellenar la

serie en contextos donde se debe producir relleno. Para obtener más información sobre el relleno, consulte “Asignaciones de serie” en “Asignaciones y comparaciones”.

El resultado de la función es una serie de longitud variable con un atributo de longitud que es el mismo atributo de longitud que el de *expresión-serie* y un tipo de datos que depende del tipo de datos de *expresión-serie*:

- VARCHAR si *expresión-serie* es CHAR o VARCHAR
- CLOB si *expresión-serie* es CLOB
- VARGRAPHIC si *expresión-serie* es GRAPHIC o VARGRAPHIC
- DBCLOB si *expresión-serie* es DBCLOB
- BLOB si *serie-expresión* es BLOB

La longitud real del resultado (en unidades de serie) es *longitud*.

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos:

- Suponga que la variable ALPHA tiene un valor de ’ABCDEF’. La siguiente sentencia:

  ```
  SELECT RIGHT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  devuelve ’DEF’, que son los tres caracteres situados más a la derecha en ALPHA.
- Suponga que la variable NAME, que se define como VARCHAR(50), tiene un valor de ’KATIE AUSTIN’ y la variable entera LASTNAME_LEN tiene un valor de 6. La sentencia siguiente:

  ```
  SELECT RIGHT(NAME,LASTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  devuelve el valor ’AUSTIN’.
- La siguiente sentencia devuelve una serie de longitud cero.

  ```
  SELECT RIGHT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```
- La columna FIRSTNME de la tabla EMPLOYEE se define como VARCHAR(12). Se busca el nombre de un empleado cuyo apellido es ’BROWN’ y se debe devolver el nombre en una serie de 10 bytes.

  ```
  SELECT RIGHT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  devuelve una serie VARCHAR(12) que tiene el valor ’DAVID’ seguido de cinco caracteres en blanco.
- En una base de datos Unicode, FIRSTNAME es una columna VARCHAR(12). Uno de sus valores es la serie de 6 caracteres ’Jürgen’. Cuanto FIRSTNAME tiene este valor:

  ```
  Función...                      Devuelve...

  RIGHT(FIRSTNAME,5,CODEUNITS32)  'ürgen' -- x'C3BC7267656E'
  RIGHT(FIRSTNAME,5,CODEUNITS16)  'ürgen' -- x'C3BC7267656E'
  RIGHT(FIRSTNAME,5,OCTETS)       'rgen'  -- x'207267656E', una serie truncada
  ```

- El ejemplo siguiente funciona con la serie Unicode '&N~AB', siendo '&' el carácter de clave G de símbolo musical y '~' el carácter de tilde de combinación. A continuación se muestra esta cadena en distintos formatos de codificación Unicode:

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

Suponga que la variable UTF8_VAR, con un atributo de longitud de 20 bytes, contiene la representación UTF-8 de la serie.

```
SELECT RIGHT(UTF8_VAR, 2, CODEUNITS16),
  RIGHT(UTF8_VAR, 2, CODEUNITS32),
  RIGHT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'AB', 'AB' y 'AB', respectivamente.

```
SELECT RIGHT(UTF8_VAR, 5, CODEUNITS16),
  RIGHT(UTF8_VAR, 5, CODEUNITS32),
  RIGHT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '?N~AB', '&N~AB' y 'N~AB', respectivamente, donde '?' es X'EDB49E'.

```
SELECT RIGHT(UTF8_VAR, 10, CODEUNITS16),
  RIGHT(UTF8_VAR, 10, CODEUNITS32),
  RIGHT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores '&N~ABbbbb', '&N~ABbbbbb' y '&N~ABb', respectivamente, donde 'b' representa el carácter en blanco.

Suponga que la variable UTF16_VAR, con un atributo de longitud de 20 unidades de código, contiene la representación UTF-16BE de la serie.

```
SELECT RIGHT(UTF16_VAR, 2, CODEUNITS16),
  RIGHT(UTF16_VAR, 2, CODEUNITS32),
  RIGHT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

devuelve los valores 'AB', 'AB' y 'B', respectivamente.

```
SELECT RIGHT(UTF16_VAR, 5, CODEUNITS16),
  RIGHT(UTF16_VAR, 5, CODEUNITS32),
  RIGHT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

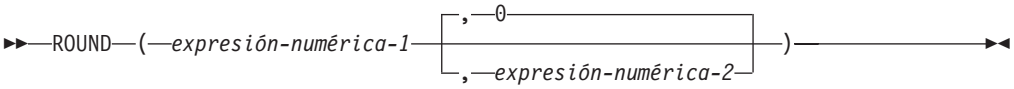
devuelve los valores '?N~AB', '&N~AB' y '~AB', respectivamente, donde '?' es el carácter de sustitución bajo autónomo X'DD1E'.

```
SELECT RIGHT(UTF16_VAR, 10, CODEUNITS16),
  RIGHT(UTF16_VAR, 10, CODEUNITS32),
  RIGHT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

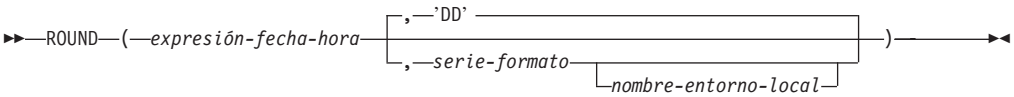
devuelve los valores '&N~ABbbbb', '&N~ABbbbbb' y '?N~AB', respectivamente, donde 'b' representa el carácter en blanco y '?' es X'DD1E'.

ROUND

ROUND numérico:



ROUND fecha-hora:



El esquema es SYSIBM. La versión de SYSFUN de la función ROUND numérico sigue estando disponible.

La función ROUND devuelve un valor redondeado de:

- Un número, redondeado al número de posiciones a la derecha o izquierda de la coma decimal, si el resultado del primer argumento es un valor numérico.
- Un valor de fecha y hora, redondeado a la unidad que especifica *serie-formato*, si el primer argumento es un valor DATE, TIME o TIMESTAMP

ROUND numérico

Si *expresión-numérica-1* es un valor positivo, un dígito con el valor 5 o un valor superior indica que se redondeará al número positivo siguiente. Por ejemplo, ROUND(3.5,0) = 4. Si *expresión-numérica-1* es un valor negativo, un dígito con el valor 5 o un valor superior indica que se redondeará al número negativo anterior. Por ejemplo, ROUND(-3.5,0) = -4.

expresión-numérica-1
Una expresión que debe devolver un valor que sea un tipo de datos numérico o CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos numérico, se convierte de forma implícita en DECFLOAT(34) antes de evaluar la función.

Si la expresión es un tipo de datos de coma flotante decimal, la modalidad de redondeo DECFLOAT no se utilizará. El comportamiento de redondeo de ROUND corresponde a un valor de ROUND_HALF_UP. Si se desea otro comportamiento de redondeo, utilice la función QUANTIZE.

expresión-numérica-2
Expresión que devuelve un valor que es un tipo de datos numérico incorporado. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si *expresión-numérica-2* no es negativa, *expresión-numérica-1* se redondea al valor absoluto de *expresión-numérica-2* posiciones a la derecha de la coma decimal.

Si *expresión-numérica-2* es negativa, *expresión-numérica-1* se redondea al valor absoluto de *expresión-numérica-2*+1 número de posiciones a la izquierda de la coma decimal.

Si el valor absoluto de una *expresión-numérica-2* negativa es superior al número de dígitos a la izquierda de la coma decimal, el resultado es 0. Por ejemplo, ROUND(748.58,-4) = 0. Si *expresión-numérica-1* es un valor positivo, un dígito con el valor 5 se redondeará al número positivo siguiente. Si *expresión-numérica-1* es un valor negativo, un dígito con el valor 5 se redondeará al número negativo anterior.

El tipo de datos y el atributo de longitud del resultado coinciden con el tipo de datos y el atributo de longitud del primer argumento, con la excepción de que la precisión aumenta en uno si *expresión-numérica-1* es DECIMAL y la precisión es menor que 31. Por ejemplo, un argumento con un tipo de datos de DECIMAL(5,2) da como resultado DECIMAL(6,2). Un argumento con un tipo de datos de DECIMAL(31,2) da como resultado DECIMAL(31,2). La escala es igual que la escala del primer argumento.

Si cualquiera de los argumentos puede ser nulo o si el argumento no es un número de coma flotante decimal y la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES, el resultado puede ser nulo. Si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

Esta función no queda afectada por el valor del registro especial CURRENT DECFLOAT ROUNDING MODE, incluso para argumentos de coma flotante decimal. El comportamiento de redondeo de ROUND corresponde a un valor de ROUND_HALF_UP. Si para un valor de coma flotante decimal desea un comportamiento que se ajuste a la modalidad de redondeo especificada por el registro especial CURRENT DECFLOAT ROUNDING MODE, utilice la función QUANTIZE en su lugar.

ROUND fecha-hora

Si *expresión-fecha-hora* tiene un tipo de datos de fecha y hora, la función ROUND devuelve *expresión-fecha-hora* redondeada a la unidad especificada por la *serie-formato*. Si no se especifica *serie-formato*, *expresión-fecha-hora* se redondea al día más cercano, como si se especificara 'DD' para *serie-formato*.

expresión-fecha-hora

Una expresión que debe devolver un valor que sea una fecha, una hora o una indicación de fecha y hora. Las representaciones de serie de estos tipos de datos no reciben soporte y deben convertirse explícitamente en un valor DATE, TIME o TIMESTAMP para poder utilizarlas con esta función; de forma alternativa, puede utilizar la función ROUND_TIMESTAMP para una representación de serie de una fecha o indicación de fecha y hora.

serie-formato

Una expresión que devuelve un tipo de datos de serie de caracteres incorporada con una longitud real que no supere los 254 bytes. El elemento de formato de *serie-formato* especifica cómo debe redondearse *expresión-fecha-hora*. Por ejemplo, si *serie-formato* es 'DD', una indicación de fecha y hora que se represente mediante una *expresión-fecha-hora* se redondea al día más cercano. Los espacios en blanco iniciales y finales se eliminan de la serie y la subserie resultante debe ser un elemento de formato válido para el tipo de *expresión-fecha-hora* (SQLSTATE 22007). El valor por omisión es 'DD', que no se puede utilizar si el tipo de datos de *expresión-fecha-hora* es TIME.

Los valores permitidos para *serie-formato* se listan en la tabla de elementos de formato que se encuentra a continuación.

nombre-entorno-local
Una constante de caracteres que especifica el entorno local que se utiliza para determinar el primer día de la semana cuando se utilizan los valores de elemento de formato DAY, DY o D. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función tiene el mismo tipo DATE que *expresión-fecha-hora*. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Los elementos de formato siguientes se utilizan para identificar la unidad de redondeo o truncamiento del valor de fecha y hora en las funciones ROUND, ROUND_TIMESTAMP, TRUNCATE y TRUNC_TIMESTAMP.

Tabla 56. Elementos de formato para ROUND, ROUND_TIMESTAMP, TRUNCATE y TRUNC_TIMESTAMP

Elemento de formato	Unidad de redondeo o truncamiento	Ejemplo de ROUND	Ejemplo de TRUNCATE
CC SCC	Siglo Redondea al inicio del siglo siguiente a partir del año 50 del siglo (por ejemplo en 1951–01–01–00.00.00). No es válido para el argumento TIME.	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1901-01-01-00.00.00.000000	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1801-01-01-00.00.00.000000
SYYYY YYYY YEAR SYEAR YYY YY S	Año Redondea a partir del 1 de julio al 1 de enero del año siguiente. No es válido para el argumento TIME.	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1898-01-01-00.00.00.000000	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1897-01-01-00.00.00.000000
IYYY IYY IY I	Año ISO Redondea a partir del 1 de julio al primer día del año ISO siguiente. El primer día del año ISO es el lunes de la primera semana ISO. No es válido para el argumento TIME.	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1898-01-03-00.00.00.000000	Valor de entrada: 1897-12-04-12.22.22.000000 Resultado: 1897-01-04-00.00.00.000000

Tabla 56. Elementos de formato para ROUND, ROUND_TIMESTAMP, TRUNCATE y TRUNC_TIMESTAMP (continuación)

Elemento de formato	Unidad de redondeo o truncamiento	Ejemplo de ROUND	Ejemplo de TRUNCATE
Q	Trimestre Redondea a partir del día 16 del segundo mes del trimestre. No es válido para el argumento TIME.	Valor de entrada: 1999-06-04-12.12.30.000000 Resultado: 1999-07-01-00.00.00.000000	Valor de entrada: 1999-06-04-12.12.30.000000 Resultado: 1999-04-01-00.00.00.000000
MONTH MON MM RM	Mes Redondea a partir del día 16 del mes. No es válido para el argumento TIME.	Valor de entrada: 1999-06-18-12.12.30.000000 Resultado: 1999-07-01-00.00.00.000000	Valor de entrada: 1999-06-18-12.12.30.000000 Resultado: 1999-06-01-00.00.00.000000
WW	Mismo día de la semana que el primer día del año. Redondea a partir de las 12 horas del cuarto día de la semana con respecto al primer día del año. No es válido para el argumento TIME.	Valor de entrada: 2000-05-05-12.12.30.000000 Resultado: 2000-05-06-00.00.00.000000	Valor de entrada: 2000-05-05-12.12.30.000000 Resultado: 2000-04-29-00.00.00.000000
IW	Mismo día de la semana que el primer día del año ISO. Consulte “Función escalar WEEK_ISO” para obtener más detalles. Redondea a partir de las 12 horas del cuarto día de la semana con respecto al primer día del año ISO. No es válido para el argumento TIME.	Valor de entrada: 2000-05-05-12.12.30.000000 Resultado: 2000-05-08-00.00.00.000000	Valor de entrada: 2000-05-05-12.12.30.000000 Resultado: 2000-05-01-00.00.00.000000

Tabla 56. Elementos de formato para ROUND, ROUND_TIMESTAMP, TRUNCATE y TRUNC_TIMESTAMP (continuación)

Elemento de formato	Unidad de redondeo o truncamiento	Ejemplo de ROUND	Ejemplo de TRUNCATE
W	Mismo día de la semana que el primer día del año. Redondea a partir de las 12 horas del cuarto día de la semana con respecto al primer día del mes. No es válido para el argumento TIME.	Valor de entrada: 2000-06-21-12.12.30.000000 Resultado: 2000-06-22-00.00.00.000000	Valor de entrada: 2000-06-21-12.12.30.000000 Resultado: 2000-06-15-00.00.00.000000
DDD DD J	Día Redondea a partir de las 12 horas del día. No es válido para el argumento TIME.	Valor de entrada: 2000-05-17-12.59.59.000000 Resultado: 2000-05-18-00.00.00.000000	Valor de entrada: 2000-05-17-12.59.59.000000 Resultado: 2000-05-17-00.00.00.000000
DAY DY D	Primer día de la semana. Redondea con respecto a las 12 horas del cuarto día de la semana. El primer día de la semana depende del entorno local (vea el *nombre-entorno-local*). No es válido para el argumento TIME.	Valor de entrada: 2000-05-17-12.59.59.000000 Resultado: 2000-05-21-00.00.00.000000	Valor de entrada: 2000-05-17-12.59.59.000000 Resultado: 2000-05-14-00.00.00.000000
HH HH12 HH24	Hora Redondea a partir del minuto 30.	Valor de entrada: 2000-05-17-23.59.59.000000 Resultado: 2000-05-18-00.00.00.000000	Valor de entrada: 2000-05-17-23.59.59.000000 Resultado: 2000-05-17-23.00.00.000000
MI	Minuto Redondea a partir del segundo 30.	Valor de entrada: 2000-05-17-23.58.45.000000 Resultado: 2000-05-17-23.59.00.000000	Valor de entrada: 2000-05-17-23.58.45.000000 Resultado: 2000-05-17-23.58.00.000000
SS	Segundo Redondea a partir de medio segundo.	Valor de entrada: 2000-05-17-23.58.45.500000 Resultado: 2000-05-17-23.58.46.000000	Valor de entrada: 2000-05-17-23.58.45.500000 Resultado: 2000-05-17-23.58.45.000000

Nota: Los elementos de formato de la Tabla 56 en la página 554 deben especificarse en mayúsculas.

Si para un argumento de fecha se especifica un elemento de formato que se aplica a la parte de hora de un valor, el argumento de fecha se devuelve sin cambios. Si para un argumento de hora se especifica un elemento de formato que no es válido para este tipo de argumento, se devuelve un error (SQLSTATE 22007).

Notas

- **Determinismo:** ROUND es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME.
  - Redondeo de un valor de fecha y hora cuando no se especifica explícitamente *nombre-entorno-local* y cuando una de las afirmaciones siguientes es verdadera:
    - *serie-formato* no es una constante
    - *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local

  Las invocaciones de la función que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales.

Ejemplos

- Calcule el valor de 873.726, redondeado a 2, 1, 0, -1, -2, -3 y -4 posiciones decimales, respectivamente.

```
VALUES (
  ROUND(873.726, 2),
  ROUND(873.726, 1),
  ROUND(873.726, 0),
  ROUND(873.726,-1),
  ROUND(873.726,-2),
  ROUND(873.726,-3),
  ROUND(873.726,-4) )
```

  Este ejemplo devuelve:

```
1         2         3         4         5         6         7
--------- --------- --------- --------- --------- --------- ---------
  873.730   873.700   874.000   870.000   900.000  1000.000     0.000
```

- Realice el cálculo utilizando los números positivos y negativos.

```
VALUES (
  ROUND(3.5, 0),
  ROUND(3.1, 0),
  ROUND(-3.1, 0),
  ROUND(-3.5,0) )
```

  Este ejemplo devuelve:

```
1    2    3    4
---- ---- ---- ----
 4.0  3.0 -3.0 -4.0
```

- Calcule el número de coma flotante decimal 3.12350 redondeado a tres decimales.

```
VALUES (
  ROUND(DECFLOAT('3.12350'), 3))
```

  Este ejemplo devuelve:

```
1
-----
3.12400
```

- Establecer la variable del lenguaje principal RND_DT con la fecha de entrada redondeada al valor de mes más cercano.

```
SET :RND_DATE = ROUND(DATE('2000-08-16'), 'MONTH');
```

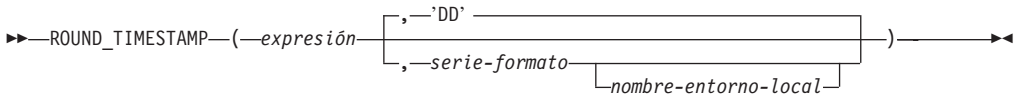
  El valor establecido es 2000-09-01.

- Establecer la variable del lenguaje principal RND_TMSTMP con la indicación de fecha y hora de entrada redondeada al valor de año más cercano.

```
SET :RND_TMSTMP = ROUND(TIMESTAMP('2000-08-14-17.30.00'),
                        'YEAR');
```

  El valor establecido es 2001-01-01-00.00.00.000000.

ROUND_TIMESTAMP



El esquema es SYSIBM.

La función escalar ROUND_TIMESTAMP devuelve un valor TIMESTAMP que es la *expresión* redondeada de la unidad que especifica *serie-formato*. Si no se especifica *serie-formato*, *expresión* se redondea al día más cercano, como si se especificara 'DD' para *serie-formato*.

expresión
Una expresión que devuelve un valor de uno de los siguientes tipos de datos incorporados: valor DATE o valor TIMESTAMP.

serie-formato
Una expresión que devuelve un tipo de datos de serie de caracteres incorporada con una longitud real que no supere los 254 bytes. El elemento de formato de *serie-formato* especifica cómo debe redondearse *expresión*. Por ejemplo, si serie-formato es 'DD', la indicación de fecha y hora representada mediante *expresión* se redondea al día más cercano. Los espacios en blanco iniciales y finales se eliminan de la serie y la subserie resultante debe ser un elemento de formato válido para una indicación de fecha y hora (SQLSTATE 22007). El valor por omisión es 'DD'.

Los valores permitidos para *serie-formato* se listan en la tabla de elementos de formato que se encuentra en la descripción de la función ROUND.

nombre-entorno-local
Constante de caracteres que especifica el entorno local que se utiliza para determinar el primer día de la semana cuando se utilizan los valores de modelo de formato DAY, DY o D. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función es TIMESTAMP con una precisión de indicación de fecha y hora de:

- *p* cuando el tipo de datos de *expresión* es TIMSTAMP(*p*)
- 0 cuando el tipo de datos de *expresión* es DATE
- 6 en caso contrario.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Notas

- **Determinismo:** ROUND_TIMESTAMP es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME.
  - Redondeo de un valor de fecha o de indicación de fecha y hora cuando no se especifica explícitamente *nombre-entorno-local* y cuando una de las afirmaciones siguientes es verdadera:

- *serie-formato* no es una constante
- *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local

Las invocaciones de la función que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales.

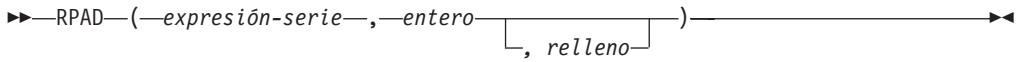
Ejemplo

- Establecer la variable del lenguaje principal *RND_TMSTMP* con la indicación de fecha y hora de entrada redondeada al valor de año más cercano.

```
SET :RND_TMSTMP = ROUND_TIMESTAMP('2000-08-14-17.30.00', 'YEAR');
```

El valor establecido es 2001-01-01-00.00.00.000000.

RPAD



El esquema es SYSIBM.

La función RPAD devuelve una serie compuesta de *expresión-serie* que se rellena por la derecha con *relleno* o con espacios en blanco. La función RPAD trata los blancos iniciales o finales de *expresión-serie* como significativos. El relleno solo se utiliza si la longitud real de *expresión-serie* es menor que *entero* y si *relleno* no es una serie vacía.

expresión-serie
Expresión que especifica la serie fuente. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

entero
Expresión entera que especifica la longitud del resultado. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor debe ser cero o un entero positivo que sea menor o igual que *n*, siendo *n* 32.672 si *expresión-serie* es una serie de caracteres o 16.336 si *expresión-serie* es una serie gráfica.

relleno
Expresión que especifica la serie con la que se realizará el relleno. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

Si *relleno* no se especifica, el carácter de relleno se determina del modo siguiente:

- Carácter en blanco SBCS si *expresión-serie* es una serie de caracteres.
- Carácter en blanco ideográfico si *expresión-serie* es una serie gráfica. Para las series gráficas de una base de datos EUC, se utiliza X'3000'. Para las series gráficas de una base de datos Unicode, se utiliza X'0020'.

El resultado de la función es una serie de longitud variable que tiene la misma página de códigos que *expresión-serie*. El valor de *expresión-serie* y el valor de *relleno* deben tener tipos de datos compatibles. Si *expresión-serie* y *relleno* tienen páginas de códigos diferentes, *relleno* se convertirá a la página de códigos de *expresión-serie*. Si *expresión-serie* o *relleno* es FOR BIT DATA, no se lleva a cabo ninguna conversión de caracteres.

El atributo de longitud del resultado depende de si el valor de *entero* está disponible cuando se compila la sentencia de SQL que contiene la invocación de la función (por ejemplo, si se especifica como constante o como expresión de constante) o de si está disponible únicamente cuando se ejecuta la función (por ejemplo, si se especifica como resultado de la invocación de una función). En el caso de que el valor esté disponible cuando se compila la sentencia de SQL que

contiene la invocación de la función, si *entero* es mayor que cero, el atributo de longitud del resultado es *entero*. En el caso de que *entero* sea 0, el atributo de longitud del resultado es 1. Si el valor está disponible únicamente cuando se ejecuta la función, el atributo de longitud del resultado se determina según lo indicado en la tabla siguiente:

Tabla 57. Cómo se determina la longitud del resultado si entero está disponible únicamente cuando se ejecuta la función

Tipo de datos de *expresión-serie*	**Longitud del tipo de datos del resultado**
CHAR(*n*) o VARCHAR(*n*)	Mínimo de *n*+100 y 32.672
GRAPHIC(*n*) o VARGRAPHIC(*n*)	Mínimo de *n*+100 y 16.336

La longitud real del resultado se determina a partir de *entero*. Si *entero* es 0, la longitud real es 0 y el resultado es una serie vacía. Si *entero* es menor que la longitud real de *expresión-serie*, la longitud real es *entero* y el resultado se trunca.

Si cualquiera de los argumentos puede ser nulo, el resultado puede ser nulo; si alguno de los argumentos es nulo, el resultado es el valor NULL.

Ejemplos

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La consulta siguiente rellenaría completamente un valor con puntos por la derecha:

  ```
  SELECT RPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME ---------------
  Chris..........
  Meg............
  Jeff...........
  ```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La consulta siguiente rellenaría completamente un valor con *relleno* por la derecha (observe que en algunos casos hay una instancia parcial de la especificación de relleno):

  ```
  SELECT RPAD(NAME,15,'123' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME ---------------
  Chris1231231231
  Meg123123123123
  Jeff12312312312
  ```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. La consulta siguiente solo rellenaría cada valor hasta una longitud de 5:

  ```
  SELECT RPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  devuelve:

  ```
  NAME -----
  Chris
  Meg..
  Jeff.
  ```

- Supongamos que NAME es una columna CHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. Observe que el resultado de RTRIM es una serie de longitud variable en la que se han eliminado los espacios en blanco:

  ```
  SELECT RPAD(RTRIM(NAME),15,'.' ) AS NAME FROM T1;
  ```

devuelve:

```
NAME ---------------
Chris.........
Meg...........
Jeff..........
```

- Supongamos que NAME es una columna VARCHAR(15) que contiene los valores “Chris”, “Meg” y “Jeff”. Observe que “Chris” se trunca, “Meg” se rellena y “Jeff” no cambia:

```
SELECT RPAD(NAME,4,'.' ) AS NAME FROM T1;
```

devuelve:

```
NAME ----
Chri
Meg.
Jeff
```

RTRIM

El esquema es SYSIBM. (La versión SYSFUN de esta función sigue estando disponible con el soporte para los argumentos CLOB).

La función RTRIM elimina los blancos del final de la *expresión-serie*.

La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

- Si el argumento es una serie gráfica de una base de datos DBCS o EUC, se eliminan los blancos de doble byte de cola.
- Si el argumento es una serie gráfica de una base de datos Unicode, se eliminan los blancos UCS-2 de cola.
- De lo contrario, se eliminan los blancos de un solo byte de cola.

El tipo de datos del resultado de la función es:

- VARCHAR si el tipo de datos de *expresión-serie* es VARCHAR o CHAR
- VARGRAPHIC si el tipo de datos de *expresión-serie* es VARGRAPHIC o GRAPHIC

El parámetro de longitud del tipo devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para las series de caracteres es la de *expresión-serie* menos el número de bytes eliminados debido a caracteres en blanco. La longitud real del resultado para series gráficas es la longitud (en número de caracteres de doble byte) de *expresión-serie* menos el número de caracteres en blanco de doble byte eliminados. Si elimina todos los caracteres se obtiene una serie vacía de longitud variable (longitud de cero).

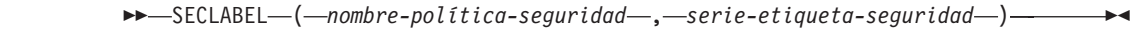
Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo: Supongamos que la variable del lenguaje principal HELLO está definida como CHAR(9) y tiene el valor ' Hola'.

```
VALUES RTRIM(:HELLO)
```

El resultado es 'Hola'.

SECLABEL



El esquema es SYSIBM.

La función SECLABEL devuelve una etiqueta de seguridad sin nombre, con un tipo de datos de DB2SECURITYLABEL. Utilice la función SECLABEL para insertar una etiqueta de seguridad con los valores de los componentes proporcionados sin tener que crear una etiqueta de seguridad con nombre.

nombre-política-seguridad
Una serie identifica una política de seguridad que exista en el servidor actual (SQLSTATE 42704). La serie debe ser una constante de serie gráfica o de caracteres o una variable del lenguaje principal.

serie-etiqueta-seguridad
Expresión que devuelve una representación válida de una etiqueta de seguridad para la política de seguridad nombrada por *nombre-política-seguridad* (SQLSTATE 4274I). La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

Ejemplos:

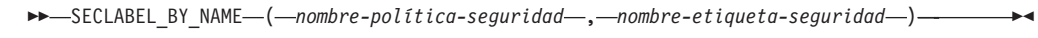
- La sentencia siguiente inserta una fila en la tabla REGIONS que está protegida por la política de seguridad CONTRIBUTIONS. SECLABEL proporciona la etiqueta de seguridad para la fila que se debe insertar. La política de seguridad CONTRIBUTIONS tiene dos componentes. La etiqueta de seguridad proporcionada tiene el elemento LIFE MEMBER para el primer componente y los elementos BLUE y YELLOW para el segundo componente.

```
INSERT INTO REGIONS
  VALUES (SECLABEL('CONTRIBUTIONS', 'LIFE MEMBER:(BLUE,YELLOW)'),
    1, 'Northeast')
```

- La sentencia siguiente inserta una fila en la tabla CASE_IDS que está protegida por la política de seguridad TS_SECPOLICY, que tiene tres componentes. La función SECLABEL proporciona la etiqueta de seguridad. La etiqueta de seguridad insertada tiene el elemento HIGH PROFILE para el primer componente, el valor vacío para el segundo componente y el elemento G19 para el tercer componente.

```
INSERT INTO CASE_IDS
  VALUES (SECLABEL('TS_SECPOLICY', 'HIGH PROFILE:():G19') , 3, 'KLB')
```

SECLABEL_BY_NAME



El esquema es SYSIBM.

La función SECLABEL_BY_NAME devuelve la etiqueta de seguridad especificada. La etiqueta de seguridad devuelta tiene un tipo de datos de DB2SECURITYLABEL. Utilice esta función para insertar una etiqueta de seguridad con nombre.

nombre-política-seguridad
Una serie identifica una política de seguridad que exista en el servidor actual (SQLSTATE 42704). La serie debe ser una constante de serie gráfica o de caracteres o una variable del lenguaje principal.

nombre-etiqueta-seguridad
Expresión que devuelve el nombre de una etiqueta de seguridad existente en el servidor actual para la política de seguridad nombrada por *nombre-política-seguridad* (SQLSTATE 4274I). La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR, VARCHAR, GRAPHIC o VARGRAPHIC.

Ejemplos:

- La usuaria Tina intenta insertar una fila en la tabla REGIONS, protegida por la política de seguridad CONTRIBUTIONS. Tina desea que la etiqueta de seguridad EMPLOYEESECLABEL proteja a la fila. Esta sentencia falla porque CONTRIBUTIONS.EMPLOYEESECLABEL es un identificador desconocido:

  ```
  INSERT INTO REGIONS
  VALUES (CONTRIBUTIONS.EMPLOYEESECLABEL, 1, 'Southwest')   -- incorrecto
  ```

  Esta sentencia falla porque el primer valor es una serie; no tiene un tipo de datos de DB2SECURITYLABEL:

  ```
  INSERT INTO REGIONS
  VALUES ('CONTRIBUTIONS.EMPLOYEESECLABEL', 1, 'Southwest') -- incorrecto
  ```

  Esta sentencia es satisfactoria porque la función SECLABEL_BY_NAME devuelve una etiqueta de seguridad que tiene un tipo de datos de DB2SECURITYLABEL:

  ```
  INSERT INTO REGIONS
  VALUES (SECLABEL_BY_NAME('CONTRIBUTIONS', 'EMPLOYEESECLABEL'),
    1, 'Southwest')                                        -- correcto
  ```

SECLABEL_TO_CHAR

El esquema es SYSIBM.

La función SECLABEL_TO_CHAR acepta una etiqueta de seguridad y devuelve una serie que contiene todos los elementos de la etiqueta de seguridad. La serie tiene el formato de la serie de la etiqueta de seguridad.

nombre-política-seguridad
Una serie identifica una política de seguridad que exista en el servidor actual (SQLSTATE 42704). La serie debe ser una constante de serie gráfica o de caracteres o una variable del lenguaje principal.

etiqueta-seguridad
Expresión que devuelve un valor de etiqueta de seguridad válido para la política de seguridad nombrada por *nombre-política-seguridad* (SQLSTATE 4274I). La expresión debe devolver un valor que sea un tipo diferenciado incorporado SYSPROC.DB2SECURITYLABEL.

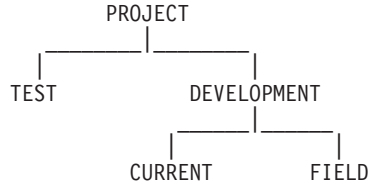
Notas

- Si el ID de autorización de la sentencia ejecuta esta función en una etiqueta de seguridad que se lee desde una columna con un tipo de datos de DB2SECURITYLABEL, entonces las credenciales LBAC de los ID de autorización pueden afectar a la salida de la función. En tal caso no se incluye un elemento en la salida si el ID de autorización no dispone de acceso de lectura a ese elemento. Un ID de autorización tiene acceso de lectura a un elemento si sus credenciales LBAC permitiesen leer datos protegidos por una etiqueta de seguridad que sólo contiene ese elemento, y no otros.

  Para el conjunto de normas DB2LBACRULES, sólo los componentes del tipo TREE pueden contener elementos para los que no tenga acceso de lectura. Para otros tipos de componentes, si alguno de los elementos bloquea el acceso de lectura, el usuario no podrá leer la fila. De este modo, sólo los componentes de tipo árbol tendrán elementos excluidos.

Ejemplo:

- La tabla EMP tiene dos columnas: RECORDNUM y LABEL. RECORDNUM tiene tipo de datos INTEGER y LABEL tiene tipo de datos DB2SECURITYLABEL. La tabla EMP está protegida por la política de seguridad DATA_ACCESSPOLICY, que utiliza el conjunto de normas DB2LBACRULES y que tiene un único componente (GROUPS, de tipo TREE). GROUPS tiene cinco elementos: PROJECT, TEST, DEVELOPMENT, CURRENT y FIELD. El diagrama siguiente muestra la relación de estos elementos entre sí:



  La tabla EMP contiene los datos siguientes:

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

El usuario cuyo ID es Djavan tiene una etiqueta de seguridad para lectura que sólo contiene el elemento DEVELOPMENT. Esto significa que Djavan tiene acceso de lectura para los elementos DEVELOPMENT, CURRENT y FIELD:

```
SELECT RECORDNUM, SECLABEL_TO_CHAR('DATA_ACCESSPOLICY', LABEL) FROM EMP
```

devuelve:

```
RECORDNUM  LABEL
---------  ----------------
        2  FIELD
        3  (CURRENT, FIELD)
```

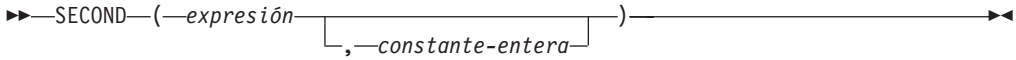
La fila con un valor de 1 para RECORDNUM no está incluida en la salida, porque las credenciales LBAC de Djavan no le permiten leer esta fila. En la fila con un valor de 2 para RECORDNUM, el elemento TEST no está incluido en la salida, porque Djavan no tiene acceso de lectura a ese elemento; Djavan no podría haber accedido a la fila si TEST fuera el único elemento en la etiqueta de seguridad. Los elementos CURRENT y FIELD aparecen en la salida, porque Djavan tiene acceso de lectura a ellos.

Ahora Djavan tiene otorgada una exención para la norma DB2LBACREADTREE. Esto significa que ningún elemento de un componente de tipo TREE bloqueará el acceso de lectura. La misma consulta devuelve:

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

Esta vez la salida contiene todas las filas y todos los elementos, porque la exención concede a Djavan el acceso de lectura para todos los elementos.

SECOND



El esquema es SYSIBM.

La función SECOND devuelve la parte correspondiente a los segundos de un valor con segundos fraccionarios opcionales.

expresión
Una expresión que devuelve un valor que debe ser de tipo DATE, TIME, TIMESTAMP, una duración de hora, una duración de indicación de fecha y hora o una representación de serie válida de una fecha, hora o indicación de fecha y hora que no sea un CLOB ni un DBCLOB.

Si *expresión* es un valor DATE o una representación de serie válida de una fecha, ésta primero se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00).

Sólo las bases de datos Unicode dan soporte a un argumento que sea una representación de serie gráfica de una fecha, una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Si *expresión* es una serie de caracteres, los espacios en blanco iniciales se incluyen y los espacios en blanco finales se eliminan antes de convertir el valor en un valor de fecha y hora. Para conocer los formatos válidos de las representaciones de serie de los valores de fecha y hora, consulte "Representación mediante series de los valores de fecha y hora" en "Valores de fecha y hora".

constante-entera
Constante de entero que representa la escala de los segundos fraccionarios. El valor debe estar comprendido entre 0 y 12.

El resultado de la función con un solo argumento es un entero grande. El resultado de la función con dos argumentos es DECIMAL(2+*s*,*s*), donde *s* es el valor de la *constante-entero*. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del primer argumento y el número de argumentos:

- Si el primer argumento es un valor DATE, TIME, TIMESTAMP o una representación de serie válida de una fecha, hora o indicación de fecha y hora:
  - Si sólo se especifica un argumento, el resultado es la parte correspondiente a los segundos del valor (de 0 a 59).
  - Si se especifican los dos argumentos, el resultado es la parte correspondiente a los segundos del valor (de 0 a 59) y *constante-entero* dígitos de la parte correspondiente a los segundos fraccionarios del valor, si procede. Si el valor no contiene segundos fraccionarios, se devuelven ceros.
- Si el primer argumento es una duración de hora o una duración de indicación de fecha y hora:
  - Si sólo se especifica un argumento, el resultado es la parte correspondiente a los segundos del valor (de -99 a 99).

– Si se especifican los dos argumentos, el resultado es la parte correspondiente a los segundos del valor (de -99 a 99) y *constante-entero* dígitos de la parte correspondiente a los segundos fraccionarios del valor, si procede. Si el valor no contiene segundos fraccionarios, se devuelven ceros. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos

- Supongamos que la variable del lenguaje principal TIME_DUR (decimal(6,0)) tiene el valor 153045.

```
SELECT SECOND(:TIME_DUR)
  FROM SYSIBM.SYSDUMMY1
```

  Devuelve el valor 45.

- Supongamos que la columna RECEIVED (cuyo tipo de datos es TIMESTAMP) tiene un valor interno equivalente a 1988-12-25-17.12.30.000000.

```
SELECT SECOND(RECEIVED)
  FROM IN_TRAY
```

  Devuelve el valor 30.

- Obtener los segundos con segundos fracciones de una indicación de fecha y hora con milisegundos.

```
SELECT SECOND (CURRENTTIMESTAMP(3), 3)
  FROM SYSIBM.SYSDUMMY1
```

  Devuelve un valor DECIMAL(5,3) en la indicación de fecha y hora actual que puede ser algo parecido a 54.321.

SIGN

El esquema es SYSIBM. (La versión SYSFUN de la función SIGN continúa estando disponible.)

Devuelve un indicador del signo del argumento. Si el argumento es menor que cero, se devuelve -1. Si el argumento es el valor de coma flotante decimal de -0, se devuelve el valor de coma flotante decimal de -0. Si el argumento es igual a cero, devuelve 0. Si el argumento es mayor que cero, devuelve 1.

El argumento puede ser de cualquier tipo de datos interno. Los valores DECIMALES y REALES se convierten a números de coma flotante de precisión doble para que los procese la función.

El resultado de la función es:

- SMALLINT si el argumento es SMALLINT
- INTEGER si el argumento es INTEGER
- BIGINT si el argumento es BIGINT
- DECFLOAT(*n*) si el argumento es DECFLOAT(*n*)
- de lo contrario, DOUBLE.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo:

- Suponga que la variable del lenguaje principal PROFIT es un entero grande con un valor de 50000.

  ```
  VALUES SIGN(:PROFIT)
  ```

  Devuelve el valor 1.

SIN

El esquema es SYSIBM. (La versión SYSFUN de la función SIN continúa estando disponible).

Devuelve el seno del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

SINH

El esquema es SYSIBM.

Devuelve el seno hiperbólico del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

SMALLINT

De numérico a smallint:

De serie a smallint:

El esquema es SYSIBM.

La función SMALLINT devuelve una representación de entero pequeño de:

- Un número
- Una representación de serie de un número

De numérico a smallint:

expresión-numérica
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno.

El resultado es el mismo número que el que se generaría si se asignara el argumento a una variable o columna de enteros pequeños. La parte fraccional del argumento se trunca. Si la parte completa del argumento no está dentro del rango de los enteros pequeños, se devuelve un error (SQLSTATE 22003).

De serie a smallint:

expresión-serie
Una expresión que devuelve un valor de serie de caracteres o la representación de serie gráfica Unicode de un número con una longitud no superior a la longitud máxima de una constante de caracteres.

El resultado es el mismo número que generaría CAST (*expresión-serie*AS SMALLINT). Los espacios en blanco iniciales y finales se eliminan y la serie resultante debe ajustarse a las normas para formar una constante de entero, decimal, de coma flotante o coma flotante decimal (SQLSTATE 22018). Si la parte completa del argumento no está dentro del rango de los enteros pequeños, se devuelve un error (SQLSTATE 22003). El tipo de datos de *expresión-serie* no debe ser CLOB ni DBCLOB (SQLSTATE 42884).

El resultado de la función es un entero pequeño. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones. Para obtener más información, consulte la "especificación CAST".

Ejemplo

Utilizando la tabla EMPLOYEE, seleccione una lista que contenga el salario (SALARY) dividido por el nivel de formación (EDLEVEL). Trunque cualquier decimal en el cálculo. La lista también debe contener los valores utilizados en el cálculo y el número de empleado (EMPNO).

```
SELECT SMALLINT(SALARY / EDLEVEL), SALARY, ESDLEVEL, EMPNO
  FROM EMPLOYEE
```

SOUNDEX

El esquema es SYSFUN.

Devuelve un código de 4 caracteres que representa el sonido de las palabras del argumento. El resultado se puede utilizar para compararlo con el sonido de otras series.

El argumento puede ser una serie de caracteres de tipo CHAR o VARCHAR, cuya longitud no sea mayor que 4.000 bytes. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función. La función interpreta los datos que se le pasan como si se tratase de caracteres ASCII, aunque la codificación sea UTF-8.

El resultado de la función es CHAR(4). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La función SOUNDEX es útil para buscar series de las que se conoce el sonido pero no su ortografía exacta. Realiza suposiciones de la manera en que el sonido de las letras y de la combinación de letras puede ayudar a buscar palabras con sonidos similares. La comparación puede realizarse directamente o pasando las series como argumentos a la función DIFFERENCE.

Ejemplo:

Utilizando la tabla EMPLOYEE, busque el EMPNO y el LASTNAME del empleado cuyo apodo suena como 'Loucesy'.

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
  WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

Este ejemplo devuelve lo siguiente:

```
EMPNO  LASTNAME
------ ---------------
000110 LUCCHESSI
```

SPACE

El esquema es SYSFUN.

Devuelve una serie de caracteres que consta de espacios en blanco con la longitud especificada por el argumento.

El argumento puede ser SMALLINT o INTEGER.

El resultado de la función es VARCHAR(4000). El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

SQRT

El esquema es SYSIBM. (La versión SYSFUN de la función SQRT continúa estando disponible.)

La función SQRT devuelve la raíz cuadrada de un número.

El argumento debe ser una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento es de coma flotante decimal, la operación se realiza como coma flotante decimal; en caso contrario, el argumento se convierte a coma flotante de precisión doble para que la procese la función.

Si el argumento es DECFLOAT(*n*), el resultado es DECFLOAT(*n*); en caso contrario, el resultado es un número de coma flotante de precisión doble.

El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Notas

- ***Resultados que implican valores especiales de DECFLOAT:*** Si el argumento es un valor de coma flotante decimal especial, se aplicarán las normas para las operaciones aritméticas generales para la coma flotante decimal. Consulte el apartado "General arithmetic operation rules for decimal floating-point" en "Normas generales de operaciones aritméticas para coma flotante decimal" en la página 236.

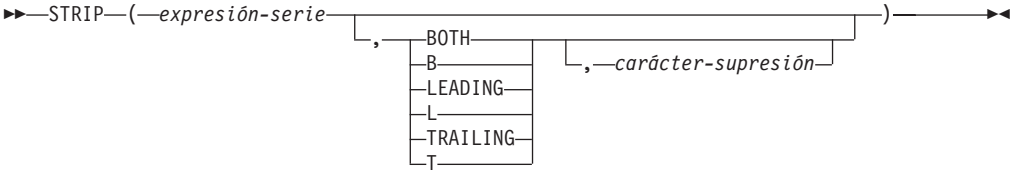
Ejemplos

- Supongamos que SQUARE es una variable del lenguaje principal DECIMAL(2,1) con un valor de 9,0.

  ```
  VALUES SQRT(:SQUARE)
  ```

  Devuelve el valor aproximado 3,00.

STRIP



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función.

La función STRIP suprime blancos o las apariciones de otro carácter especificado del final o del principio de una expresión de serie.

La función STRIP es idéntica a la función escalar TRIM.

expresión-serie
Una expresión que especifica la serie de la que se deriva el resultado. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

BOTH, LEADING o TRAILING
Especifica si se suprimen los caracteres del principio, del final o de ambos extremos de la expresión de serie. Si no se especifica este argumento, se suprimen los caracteres del final y del principio de la serie.

carácter-supresión
Una constante de un sólo carácter que especifica el carácter que se ha de suprimir. El *carácter-supresión* puede ser cualquier carácter cuya codificación UTF-32 sea un carácter individual o un valor numérico de un solo dígito. Se compara la representación binaria del carácter.

Si no se especifica el *carácter-supresión* y:

- Si la *expresión-serie* es una serie de gráficos DBCS, el valor por omisión para *carácter-supresión* es un blanco DBCS, cuyo elemento de código depende de la página de códigos de base de datos
- Si la *expresión-serie* es una serie gráfica UCS-2, el *carácter-supresión* por omisión es un espacio en blanco UCS-2 (X'0020')
- De lo contrario, el *carácter-supresión* por omisión es un espacio en blanco SBCS (X'20')

El resultado es una serie de longitud variable con la misma longitud máxima que el atributo de longitud de la *expresión-serie*. La longitud real del resultado es la longitud de la *expresión-serie* menos el número de bytes que se ha de suprimir. Si se suprimen todos los caracteres, el resultado es una serie de longitud variable vacía. La página de códigos del resultado es la misma que la página de códigos de la *expresión-serie*.

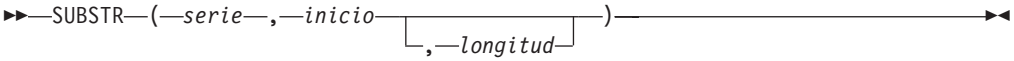
Ejemplo:

- Supongamos que la variable BALANCE de sistema principal de tipo CHAR(9) tiene un valor '000345.50'.

```
SELECT STRIP(:BALANCE, LEADING, '0'),
FROM SYSIBM.SYSDUMMY1
```

devuelve el valor ′345.50′.

SUBSTR



El esquema es SYSIBM.

La función SUBSTR devuelve una subserie de una serie.

serie
Una expresión que especifica la serie de la que se deriva el resultado.

La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente en VARCHAR antes de evaluar la función. Si la *serie* es una serie de caracteres o una serie binaria, una subserie de *serie* es cero o más bytes contiguos de la *serie*. Si la *serie* es una serie gráfica, una subserie de *serie* es cero o más caracteres de doble byte contiguos de *serie*.

inicio
Una expresión que especifica la posición del primer byte del resultado de una serie de caracteres o de una serie binaria o la posición del primer carácter del resultado de una serie gráfica. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe estar comprendido entre 1 y la longitud o longitud máxima de la *serie*, en función de si la *serie* es de longitud fija o de longitud variable (SQLSTATE 22011, si está fuera de rango). Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación.

longitud
Una expresión que especifica la longitud del resultado. Si se especifica, la expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero debe estar en el rango comprendido entre 0 y *n*, donde *n* es igual (el atributo de longitud de la *serie*) - *inicio* + 1 (SQLSTATE 22011, si está fuera de rango).

Si *longitud* se especifica explícitamente, *serie* se rellena por la derecha con el número necesario de caracteres en blanco (de un solo byte para series de caracteres; de doble byte para series gráficas) o caracteres cero hexadecimales (para las series BLOB) para que la subserie especificada de *serie* exista siempre. El valor por omisión para la *longitud* es el número de bytes desde el byte especificado por el *inicio* hasta el último byte de la *serie* en el caso de la serie de caracteres o la serie binaria o el número de caracteres de doble byte del carácter especificado por el *inicio* al último carácter de la *serie* en el caso de una serie gráfica. Sin embargo, si la *serie* es una serie de longitud variable con una longitud menor que el *inicio*, el valor por omisión es cero y el resultado es la serie vacía. Se debe especificar como número de bytes en el contexto de la página de códigos de la base de datos, no de la página de códigos de la aplicación. (Por ejemplo, la columna NAME con un tipo de datos de VARCHAR(18) y un valor de 'MCKNIGHT' dará una serie vacía con `SUBSTR(NAME,10)`.)

Si la *serie* es una serie de caracteres, el resultado de la función es una serie de caracteres representada en la página de códigos del primer argumento. Si es una serie binaria, el resultado de la función es una serie binaria. Si es una serie gráfica, el resultado de la función es una serie gráfica representada en la página de códigos del primer argumento. Si el primer argumento es una variable del lenguaje principal que no es una serie binaria y no es una serie de caracteres FOR BIT DATA, la página de códigos del resultado es la página de códigos de la base de datos. Si cualquier argumento de la función SUBSTR puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La Tabla 58 muestra que el tipo del resultado y la longitud de la función SUBSTR depende del tipo y los atributos de sus entradas.

Tabla 58. Tipo de datos y longitud del resultado de SUBSTR

Tipo de datos del argumento de la serie	**Argumento de la longitud**	**Tipo de datos del resultado**
CHAR(A)	constant (*l*<255)	CHAR(*l*)
CHAR(A)	no especificado pero el argumento *inicio* es una constante	CHAR(A-*inicio*+1)
CHAR(A)	no es una constante	VARCHAR(A)
VARCHAR(A)	constant (*l*<255)	CHAR(*l*)
VARCHAR(A)	constant (254<*l*<32673)	VARCHAR(*l*)
VARCHAR(A)	no es una constante o no se especifica	VARCHAR(A)
CLOB(A)	constant (*l*)	CLOB(*l*)
CLOB(A)	no es una constante o no se especifica	CLOB(A)
GRAPHIC(A)	constant (*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	no especificado pero el argumento *inicio* es una constante	GRAPHIC(A-*inicio*+1)
GRAPHIC(A)	no es una constante	VARGRAPHIC(A)
VARGRAPHIC(A)	constant (*l*<128)	GRAPHIC(*l*)
VARGRAPHIC(A)	constant (127<*l*<16337)	VARGRAPHIC(*l*)
VARGRAPHIC(A)	no es una constante	VARGRAPHIC(A)
DBCLOB(A)	constant (*l*)	DBCLOB(*l*)
DBCLOB(A)	no es una constante o no se especifica	DBCLOB(A)
BLOB(A)	constant (*l*)	BLOB(*l*)
BLOB(A)	no es una constante o no se especifica	BLOB(A)

Nota: los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos y no se recomiendan.

Si la *serie* es una serie de longitud fija, la omisión de la *longitud* equivale implícitamente a especificar LENGTH(*serie*) - *inicio* + 1. Si la *serie* es una serie de longitud variable, la omisión de la *longitud* equivale implícitamente a especificar 0 o LENGTH(*serie*) - *inicio* + 1, lo que sea mayor.

Notas

- En SQL dinámico, la *serie*, el *inicio* y la *longitud* pueden representarse mediante un marcador de parámetro. Si se utiliza un marcador de parámetro para la *serie*, el tipo de datos del operando será VARCHAR y el operando podrá contener nulos.
- Aunque no se indica explícitamente en las definiciones de resultados anteriores, la semántica implica que si la *serie* es una serie de caracteres que combina caracteres de un solo byte y de varios bytes, el resultado podría contener fragmentos de caracteres de varios bytes, en función de los valores de *inicio* y *longitud*. Por ejemplo, el resultado podría empezar con el segundo byte de un carácter de varios bytes o bien finalizar con el primer byte de un carácter de varios bytes. La función SUBSTR no detecta los fragmentos de este tipo, ni proporciona ningún proceso especial en caso de que se produzcan.

Ejemplos

- Supongamos que la variable del lenguaje principal NAME (VARCHAR(50)) tiene un valor de 'BLUE JAY' y la variable del lenguaje principal SURNAME_POS (int) tiene un valor de 6.

  ```
  SUBSTR(:NAME, :SURNAME_POS)
  ```

  Devuelve el valor 'JAY'

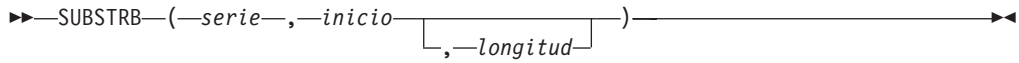
  ```
  SUBSTR(:NAME, :SURNAME_POS,1)
  ```

  Devuelve el valor 'J'.
- Seleccione todas las filas de la tabla PROJECT para las que el nombre del proyecto (PROJNAME) empiece por la palabra 'OPERATION '.

  ```
  SELECT * FROM PROJECT
    WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
  ```

  El espacio al final de la constante es necesario como preludio de las palabras iniciales como 'OPERATIONS'.

SUBSTRB



El esquema es SYSIBM.

La función SUBSTRB devuelve una subserie de una serie, empezando desde una posición especificada de la serie. Las longitudes se calculan en bytes.

La función SUBSTRB está disponible a partir de la versión 9.7, Fixpack 1.

serie
Una expresión que especifica la serie de la que se deriva el resultado.

La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, éste se convierte implícitamente en un valor VARCHAR antes de evaluarse la función. Una subserie de la *serie* tiene cero o más bytes contiguos de la *serie*. En una base de datos Unicode, si el valor es un tipo de datos de gráficos, éste se convierte implícitamente en un tipo de datos de serie de caracteres antes de evaluarse la función.

inicio
Expresión que especifica la posición inicial en la *serie* del inicio de la subserie resultante. La expresión debe devolver un valor que sea un tipo de datos numérico incorporado, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si el *inicio* es positivo, la posición inicial se calcula a partir del inicio de la serie. Si el *inicio* es mayor que la longitud de la *serie*, se devuelve una serie de longitud cero.

Si el *inicio* es negativo, la posición inicial se calcula a partir del final de la serie, y contando en sentido inverso. Si el valor absoluto de la *serie* es mayor que la longitud de la *serie*, se devuelve una serie de longitud cero.

Si el *inicio* es 0, se utiliza la posición inicial 1.

longitud
Expresión que especifica la longitud del resultado en bytes. Si se especifica, la expresión debe devolver un valor que sea un tipo de datos numérico incorporado, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si el valor de la *longitud* es mayor que el número de bytes desde la posición inicial hasta el final de la serie, la longitud resultante es la longitud del primer argumento menos la posición inicial más uno.

Si el valor de la *longitud* es menor que o igual a cero, el resultado de SUBSTRB es una serie de longitud cero.

El valor por omisión para la *longitud* es el número de bytes desde la posición que especifica *inicio* hasta el último byte de la *serie*.

Si la *serie* es un tipo de datos CHAR o VARCHAR, el resultado de la función es un tipo de datos VARCHAR. Si es un CLOB, el resultado de la función es un CLOB. Si es un BLOB, el resultado de la función es un BLOB. Si el primer argumento es

una variable del lenguaje principal que no es una serie binaria y no es una serie de caracteres FOR BIT DATA, la página de códigos del resultado es la página de códigos de la sección; de lo contrario, es la página de códigos del primer argumento.

El atributo de longitud del resultado es igual al atributo de longitud del primer argumento a menos que los argumentos *inicio* y *longitud*, ambos, se hayan especificado y definido como constantes. En este caso, el atributo de longitud del resultado se determina de la forma siguiente:

- Si la *longitud* es una constante menor que o igual a cero, el atributo de longitud del resultado es cero.
- Si el *inicio* no es una constante, pero la *longitud* sí es una constante, el atributo de longitud del resultado es el menor entre el atributo de longitud del primer argumento y la *longitud*.
- Si el *inicio* es una constante, pero la *longitud* no es una constante o no se ha especificado, el atributo de longitud del resultado es el atributo de longitud del primer argumento menos la posición inicial más uno.
- Si el *inicio* y la *longitud* son constantes, el atributo de longitud del resultado es el menor entre los valores siguientes:
  - *longitud*
  - El atributo de longitud del primer argumento menos la posición inicial más uno

Si algún argumento de la función SUBSTRB puede ser nulo, el resultado puede ser nulo; si algún argumento es nulo, el resultado es el valor nulo.

Notas

- En SQL dinámico, la *serie*, el *inicio* y la *longitud* pueden representarse mediante un marcador de parámetro. Si se utiliza un marcador de parámetro para la *serie*, el tipo de datos del operando será VARCHAR y el operando podrá contener nulos.
- Aunque no se indica explícitamente en las definiciones de resultados anteriores, la semántica implica que si la *serie* es una serie de caracteres que combina caracteres de un solo byte y caracteres de varios bytes, el resultado podría contener fragmentos de caracteres de varios bytes, en función de los valores de *inicio* y *longitud*. Por ejemplo, el resultado podría empezar con el segundo byte de un carácter de varios bytes o bien finalizar con el primer byte de un carácter de varios bytes. La función SUBSTRB detectará estos caracteres parciales y sustituirá cada byte de un carácter incompleto por un único carácter en blanco.
- SUBSTRB es similar a la función SUBSTR existente, con las excepciones que se indican a continuación:
  - SUBSTRB da soporte a un valor de *inicio* negativo, que indica que el proceso debe iniciarse desde el final de la serie.
  - SUBSTRB permite que la *longitud* sea mayor que la longitud resultante calculada. En este caso, se devolverá una serie más corta, en lugar de devolverse un error.
  - Los datos de entrada de gráficos no reciben soporte de forma nativa para el primer argumento de SUBSTRB. En una base de datos Unicode, los datos gráficos reciben soporte, pero primero se convierten en datos de caracteres, antes de evaluarse la función, y las longitudes se calculan en bytes.
  - El tipo de datos resultante de SUBSTRB es VARCHAR si el tipo de datos de entrada es CHAR.

– El atributo de longitud del resultado para SUBSTRB es igual al atributo de longitud del primer argumento o bien se obtiene en función de los atributos de *inicio* o *longitud*, si cualquiera de éstos son constantes.

Ejemplos

- Imaginemos que la variable del lenguaje principal NAME (VARCHAR(50)) tiene el valor 'BLUE JAY' y que la variable del lenguaje principal SURNAME_POS (INTEGER) tiene el valor 6.

  ```
  SUBSTRB(:NAME, :SURNAME_POS)
  ```

  Devuelve el valor 'JAY'.

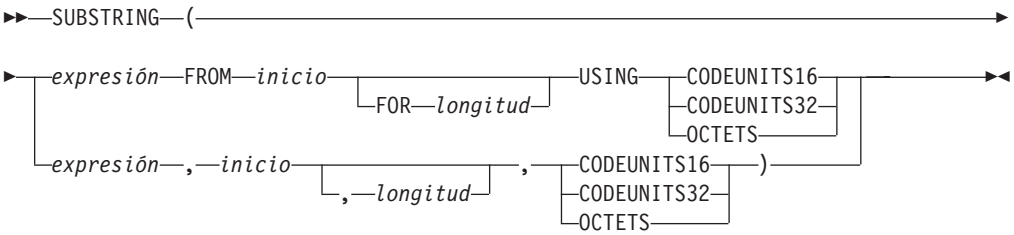
  ```
  SUBSTRB(:NAME, :SURNAME_POS,1)
  ```

  Devuelve el valor 'J'.

- Seleccione todas las filas de la tabla PROJECT que acaban en 'ING'.

  ```
  SELECT * FROM PROJECT
     WHERE SUBSTRB(PROJNAME,-3) = 'ING'
  ```

SUBSTRING



El esquema es SYSIBM.

La función SUBSTRING devuelve una subserie de una serie.

expresión
Una expresión que especifica la serie de la que se deriva el resultado. La expresión debe devolver un valor que sea de un tipo de datos de serie incorporada, numérico o de fecha y hora. Si el valor no es un tipo de datos de serie, se convierte implícitamente a VARCHAR antes de evaluar la función. Si la *expresión* es una serie de caracteres, el resultado es una serie de caracteres. Si la *expresión* es una serie gráfica, el resultado de la función es una serie gráfica. Si la *expresión* es una serie binaria, el resultado de la función es una serie binaria.

Una subserie de la *expresión* son cero o más unidades de la serie de caracteres contiguas de la *expresión*.

inicio
Expresión que especifica la posición dentro de *expresión* que será la primera unidad de serie del resultado. La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función. El valor del entero puede ser positivo, negativo o cero; un valor de 1 indica que la primera unidad de serie del resultado es la primera unidad de serie de *expresión*. Si se especifica OCTETS y *expresión* corresponde a datos gráficos, el valor del entero debe ser impar; de lo contrario, se devolverá un error (SQLSTATE 428GC).

longitud
La expresión debe devolver un valor que sea un tipo de datos incorporado numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si la *expresión* es una serie de longitud fija, la omisión de *longitud* será una especificación implícita de *expresión* USING *unidad-serie*) - *inicio* + 1, que es el número de *unidades-serie* (CODEUNITS16, CODEUNITS32 u OCTETS) desde el *inicio* hasta la última posición de la *expresión*. Si *expresión* es una serie de caracteres de longitud variable, la omisión de *longitud* será una especificación implícita de cero o CHARACTER_LENGTH(*expresión* USING *unidad-serie*) - *inicio* + 1, el que sea mayor. Si la longitud deseada es cero, el resultado es una serie de caracteres vacía.

Si el valor no es del tipo INTEGER, se convierte implícitamente en INTEGER antes de evaluar esta función. El valor debe ser igual o mayor que cero. Si se especifica un valor superior a *n*, donde *n* sea (atributo de longitud de *expresión*) - *inicio* + 1, *n* se utilizará como longitud de la subserie resultante. El valor se

expresa en las unidades que se especifican explícitamente. Si se especifica OCTETS y *expresión* corresponde a datos gráficos, el valor tiene que ser par (SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de la serie de *inicio* y *longitud*. CODEUNITS16 especifica que *inicio* y *longitud* deben expresarse en unidades de código UFT-16 de 16 bits. CODEUNITS32 especifica que el *inicio* y la *longitud* se han de expresar en unidades de código UTF-32 de 32 bits. OCTETS especifica que *inicio* y *longitud* deben expresarse en bytes.

Si la unidad de la serie se especifica como CODEUNITS16 o CODEUNITS32 y la *expresión* es una serie binaria o datos de bits, se devuelve un error (SQLSTATE 428GC). Si la unidad de la serie se especifica como OCTETS y *expresión* es una serie binaria, se devolverá un error (SQLSTATE 42815).

Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado “Unidades de serie en funciones incorporadas” en “Series de caracteres”.

Cuando se invoca la función SUBSTRING utilizando OCTETS, y la *serie-fuente* se codifica en una página de códigos que requiere más de un byte por elemento de código (combinados o MBCS), la operación SUBSTRING debe dividir un elemento de código de varios bytes y la subserie resultante puede comenzar o finalizar con un elemento de código parcial. Si sucede esto, la función sustituye los bytes de los elementos de código parciales iniciales o finales por espacios en blanco de modo que no se modifique la longitud de bytes del resultado. Consulte, a continuación, un ejemplo relacionado.

El atributo de longitud del resultado es igual al atributo de longitud de la *expresión*. Si cualquier argumento de la función puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo. El resultado no se rellena con ningún carácter. Si la *expresión* tiene la longitud real 0, el resultado también tiene la longitud real 0.

Notas:

- El atributo de longitud del resultado es igual al atributo de longitud de la expresión de la serie de entrada. Este comportamiento es diferente del comportamiento de la función SUBSTR, donde el atributo de longitud se obtiene a partir de los argumentos *inicio* y *longitud* de la función.

Ejemplos:

- FIRSTNAME es una columna VARCHAR(12) de la tabla T1. Uno de sus valores es la serie de 6 caracteres ’Jürgen’. Cuanto FIRSTNAME tiene este valor:

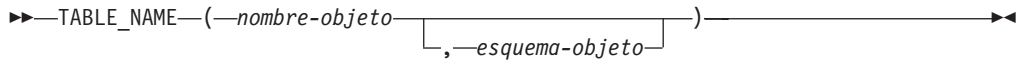
```
Función ...                          Devuelve ...
-----------------------------------  -----------------------------------
SUBSTRING(FIRSTNAME,1,2,CODEUNITS32)  'Jü' -- x'4AC3BC'
SUBSTRING(FIRSTNAME,1,2,CODEUNITS16)  'Jü' -- x'4AC3BC'
SUBSTRING(FIRSTNAME,1,2,OCTETS)       'J ' -- x'4A20' (una serie truncada)
SUBSTRING(FIRSTNAME,8,CODEUNITS16)    una serie de longitud cero
SUBSTRING(FIRSTNAME,8,4,OCTETS)       una serie de longitud cero
```

- El ejemplo siguiente ilustra cómo SUBSTRING sustituye los bytes de los elementos de código de múltiples bytes parciales iniciales o finales por espacios en blanco cuando la unidad de longitud de la serie de caracteres es OCTETS. Presupone que UTF8_VAR contiene la representación UTF-8 de la serie Unicode ’&N~AB’, donde ’&’ es el símbolo musical de la clave de sol y ’~’ es el carácter de tilde de combinación.

```
SUBSTRING(UTF8_VAR, 2, 5, OCTETS)
```

Tres bytes en blanco preceden la 'N' y un byte en blanco sigue a la 'N'.

TABLE_NAME



El esquema es SYSIBM.

La función TABLE_NAME devuelve un nombre no calificado del objeto encontrado después de que se haya resuelto cualquier cadena de alias. El *nombre-objeto* especificado (y el *esquema-objeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un alias, se devuelve el nombre no calificado del punto de inicio. El nombre resultante puede ser de una tabla, de una vista o de un objeto no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombre-objeto
Una expresión de caracteres que representa el nombre no calificado (normalmente de un alias existente) que se ha de resolver. El tipo de datos de *esquema-objeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

esquema-objeto
Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombre-objeto* suministrado antes de la resolución. El tipo de datos de *nombre-objeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

Si no se suministra el *esquema-objeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombre-objeto* puede ser nulo, el resultado puede ser nulo; si *nombre-objeto* es nulo, el resultado es el valor nulo. Si *esquema-objeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre no calificado. El nombre del resultado puede representar uno de los siguientes elementos:

tabla El valor para el *nombre-objeto* era un nombre de tabla (se devuelve el valor de entrada) o un alias que se ha resuelto en la tabla cuyo nombre se devuelve.

vista El valor para el *nombre-objeto* era un nombre de vista (se devuelve el valor de entrada) o un alias que se ha resuelto en la vista cuyo nombre se devuelve.

objeto no definido

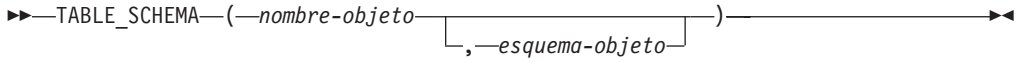
El valor para el *nombre-objeto* era un objeto no definido (se devuelve el valor de entrada) o un alias que se ha resuelto en el objeto no definido cuyo nombre se devuelve.

Por lo tanto, si se da un valor no nulo a esta función, siempre se devuelve un valor, incluso si no existe ningún objeto con el nombre del resultado.

Nota: Para mejorar el rendimiento en configuraciones de bases de datos particionadas evitando la comunicación innecesaria que se produce entre la partición de coordinación y la partición de catálogo al utilizar las funciones

escalares TABLE_SCHEMA y TABLE_NAME, en su lugar se puede utilizar la función de tabla BASE_TABLE.

TABLE_SCHEMA



El esquema es SYSIBM.

La función TABLE_SCHEMA devuelve el nombre de esquema del objeto encontrado después de que se haya resuelto cualquier cadena de alias. El *nombre-objeto* especificado (y el *esquema-objeto*) se utilizan como el punto de inicio de la resolución. Si el punto de inicio no hace referencia a un alias, se devuelve el nombre de esquema del punto de inicio. El nombre de esquema resultante puede ser de una tabla, de una vista o de un objeto no definido. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

nombre-objeto
Una expresión de caracteres que representa el nombre no calificado (normalmente de un alias existente) que se ha de resolver. El tipo de datos de *esquema-objeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

esquema-objeto
Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombre-objeto* suministrado antes de la resolución. El tipo de datos de *nombre-objeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

Si no se suministra el *esquema-objeto*, se utiliza el esquema por omisión para el calificador.

El tipo de datos del resultado de la función es VARCHAR(128). Si *nombre-objeto* puede ser nulo, el resultado puede ser nulo; si *nombre-objeto* es nulo, el resultado es el valor nulo. Si *esquema-objeto* es el valor nulo, se utiliza el nombre de esquema por omisión. El resultado es la serie de caracteres que representa un nombre de esquema. El esquema del resultado puede representar el nombre de esquema para uno de los siguientes elementos:

tabla El valor para el *nombre-objeto* era un nombre de tabla (se devuelve la entrada o el valor por omisión de *esquema-objeto*) o un alias que se ha resuelto en una tabla para la que se devuelve el nombre de esquema.

vista El valor para el *nombre-objeto* era un nombre de vista (se devuelve la entrada o el valor por omisión de *esquema-objeto*) o un alias que se ha resuelto en una vista para la que se devuelve el nombre de esquema.

objeto no definido
El valor para el *nombre-objeto* era un objeto no definido (se devuelve la entrada o el valor por omisión de *esquema-objeto*) o un alias que se ha resuelto en un objeto no definido para el que se devuelve el nombre de esquema.

Por lo tanto, si se da a esta función un valor de *nombre-objeto* que no es nulo, siempre se devuelve un valor, incluso si el nombre de objeto con el nombre de esquema del resultado no existe. Por ejemplo, `TABLE_SCHEMA('DEPT', 'PEOPLE')` devuelve 'PEOPLE ' si no se encuentra la entrada del catálogo.

Nota: Para mejorar el rendimiento en configuraciones de bases de datos particionadas evitando la comunicación innecesaria que se produce entre la partición de coordinación y la partición de catálogo al utilizar las funciones escalares TABLE_SCHEMA y TABLE_NAME, en su lugar se puede utilizar la función de tabla BASE_TABLE.

Ejemplos:

- PBIRD intenta seleccionar las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un alias PBIRD.A1 definido en la tabla HEDGES.T1.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('A1')
  AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

  Las estadísticas solicitadas para HEDGES.T1 se recuperan del catálogo.

- Seleccione las estadísticas para un objeto llamado HEDGES.X1 de SYSCAT.TABLES utilizando HEDGES.X1. Utilice TABLE_NAME y TABLE_SCHEMA ya que no se conoce si HEDGES.X1 es un alias o una tabla.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
  AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

  Suponiendo que HEDGES.X1 sea una tabla, las estadísticas solicitadas para HEDGES.X1 se recuperan del catálogo.

- Seleccione las estadísticas para una tabla determinada de SYSCAT.TABLES utilizando un alias PBIRD.A2 definido en HEDGES.T2 donde HEDGES.T2 no existe.

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
  AND TABSCHEMA = TABLE_SCHEMA ('A2',PBIRD')
```

  La sentencia devuelve 0 registros ya que no se encuentra ninguna entrada en SYSCAT.TABLES que coincida donde TABNAME = 'T2' y TABSCHEMA = 'HEDGES'.

- Seleccione el nombre calificado de cada entrada en SYSCAT.TABLES junto con el nombre de referencia final para cualquier entrada de alias.

```
SELECT TABSCHEMA AS SCHEMA,  TABNAME AS NAME,
  TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
  TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_NAME
  FROM SYSCAT.TABLES
```

  La sentencia devuelve el nombre calificado para cada objeto en el catálogo y el nombre de referencia final (después de haberse resuelto el alias) para cualquier entrada de alias. Para todas las entradas que no son alias, BASE_TABNAME y BASE_TABSCHEMA son nulos, por lo tanto las columnas REAL_SCHEMA y REAL_NAME contendrán nulos.

TAN

El esquema es SYSIBM. (La versión SYSFUN de la función TAN continúa estando disponible).

Devuelve la tangente del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

TANH

El esquema es SYSIBM.

Devuelve la tangente hiperbólica del argumento, donde el argumento es un ángulo expresado en radianes.

El argumento puede ser de cualquier tipo de datos numéricos incorporado (excepto DECFLOAT). Se convierte a un número de coma flotante de precisión doble para que lo procese la función.

El resultado de la función es un número de coma flotante de precisión doble. El resultado puede ser nulo si el argumento puede ser nulo o si la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

TIME

El esquema es SYSIBM.

La función TIME devuelve una hora de un valor.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie válida de una fecha, hora o indicación de fecha y hora que no sea un CLOB ni un DBCLOB.

Sólo las bases de datos Unicode dan soporte a un argumento que sea una representación de serie gráfica de una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un valor TIME. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento:

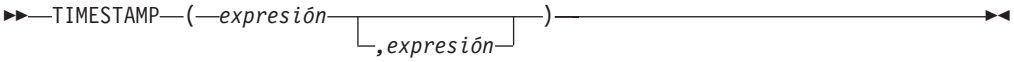
- Si el argumento es un valor DATE o una representación de serie de una fecha:
  - El resultado es medianoche.
- Si el argumento es un valor TIME:
  - El resultado es dicha hora.
- Si el argumento es un valor TIMESTAMP:
  - El resultado es la parte correspondiente a la hora de la indicación de fecha y hora.
- Si el argumento es una representación de serie de la hora o la indicación de fecha y hora:
  - El resultado es la hora representada por la serie.

Ejemplo

- Seleccione todas las notas de la tabla de ejemplo IN_TRAY que se hayan recibido como mínimo una hora más tarde (de cualquier día) que la hora actual.

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



El esquema es SYSIBM.

La función TIMESTAMP devuelve una indicación de fecha y hora a partir de un valor o par de valores.

Sólo las bases de datos Unicode dan soporte a un argumento que es una representación de serie gráfica de una fecha, una hora o una indicación de fecha y hora. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

Las normas para los argumentos dependen de si el segundo argumento se especifica y del tipo de datos del segundo argumento.

- Si sólo se especifica un único argumento, éste debe ser una expresión que devuelva un valor de uno de los tipos de datos incorporados siguientes: un valor DATE, TIMESTAMP o una serie de caracteres que no sea un CLOB. Si el argumento es una serie de caracteres, debe ser una de las siguientes:
  - Una representación de serie de caracteres de fecha o indicación de fecha y hora válida. Para conocer los formatos válidos de las representaciones de serie de los valores de fecha y de indicación de fecha y hora, consulte "Representación mediante series de los valores de fecha y hora" en "Valores de fecha y hora".
  - Una serie de caracteres con una longitud real de 13 que se adopte como resultado de la función GENERATE_UNIQUE.
  - Una serie de longitud 14 que es una serie de dígitos que representa una fecha y hora válidas con el formato *aaaaxxddhhmmss*, donde *aaaa* es el año, *xx* es el mes, *dd* es el día, *hh* es la hora, *mm* es el minuto y *ss* son los segundos.
- Si se especifican ambos argumentos:
  - Si el tipo de datos del segundo argumento no es un entero:
    - El primer argumento debe ser un valor DATE o una representación de serie válida de una fecha y el segundo argumento debe ser un valor TIME o una representación de serie válida de una hora.
  - Si el tipo de datos del segundo argumento es un entero:
    - El primer argumento debe ser un valor DATE, TIMESTAMP o una representación de serie válida de una indicación de fecha y hora o de una fecha. El segundo argumento debe ser una constante de entero entre 0 y 12 que represente la precisión de la indicación de fecha y hora.

El resultado de la función es un valor TIMESTAMP.

La precisión de indicación de fecha y hora y otras normas dependen de si se especifica el segundo argumento:

- Si se especifican los dos argumentos y el segundo argumento no es un entero:
  - El resultado es un valor TIMESTAMP(6), con la fecha que especifica el primer argumento y la hora que especifica el segundo argumento. La parte correspondiente a los segundos fraccionarios de la indicación de fecha y hora es cero.
- Si se especifican los dos argumentos y el segundo argumento es un entero:

  - El resultado es un valor TIMESTAMP, con la precisión que se especifica en el segundo argumento.
- Si sólo se especifica un único argumento y éste es un valor TIMESTAMP(*p*):
  - El resultado es ese valor TIMESTAMP(*p*).
- Si sólo se especifica un único argumento y éste es un valor DATE:
  - El resultado es esa fecha, dándose por supuesta la conversión de la hora de la medianoche a TIMESTAMP(0).
- Si sólo se especifica un argumento y es una serie:
  - El resultado es el valor TIMESTAMP(6) representado mediante esa serie ampliada tal como se describía anteriormente, con la información de hora que podría faltar. Si el argumento es una serie de longitud 14, el valor TIMESTAMP tiene una parte de cero segundos fraccionarios.

Si los argumentos sólo incluyen información de fecha, la información de hora del valor del resultado es todo ceros. Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Supongamos que la columna START_DATE (cuyo tipo de datos es DATE) tiene un valor equivalente a 1988-12-25 y que la columna START_TIME (cuyo tipo de datos es TIME) tiene un valor equivalente a 17.12.30.

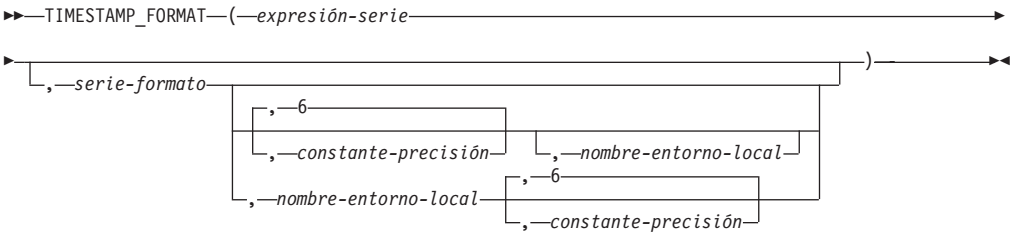
  ```
  TIMESTAMP(START_DATE, START_TIME)
  ```

  Devuelve el valor '1988-12-25-17.12.30.000000'.
- Convertir una serie de indicación de fecha y hora con 7 dígitos de segundos fraccionarios en un valor TIMESTAMP(9).

  ```
  TIMESTAMP('2007-09-24-15.53.37.2162474',9)
  ```

  Devuelve el valor '2007-09-24-15.53.37.216247400'.

TIMESTAMP_FORMAT



El esquema es SYSIBM.

La función TIMESTAMP_FORMAT devuelve una indicación de fecha y hora que se basa en la interpretación de la serie de entrada utilizando el formato especificado.

expresión-serie
La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR o VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función. La *expresión-serie* debe contener los componentes de una indicación de fecha y hora que corresponden al formato especificado por *serie-formato*.

serie-formato
La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR o VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función. La longitud real no debe superar los 254 bytes (SQLSTATE 22007). El valor es una plantilla para que pueda interpretarse la *expresión-serie* y luego convertirse en un valor de indicación de fecha y hora.

Una *serie-formato* válida debe contener como mínimo un elemento de formato, no debe contener varias especificaciones para cualquier componente de una indicación de fecha y hora y puede contener cualquier combinación de los elementos de formato, a menos que se indique lo contrario en Tabla 59 en la página 599 (SQLSTATE 22007). Por ejemplo, *serie-formato* no puede contener YY e YYYY, porque ambos se utilizan para interpretar el componente de año de *expresión-serie*. Consulte la tabla para determinar qué elementos de formato no se pueden especificar juntos. Dos elementos de formato se pueden separar opcionalmente por uno o varios de los siguientes caracteres separadores:

- signo menos (-)
- punto (.)
- barra inclinada (/)
- coma (,)
- apóstrofo (')
- punto y coma (;)
- dos puntos (:)
- espacio en blanco ( )

También se pueden especificar caracteres separadores al principio o al final de *serie-formato*. Estos caracteres separadores pueden utilizarse en cualquier combinación en la serie de formato, por ejemplo, 'AAA/MM-DD HH:MM.SS'.

Tabla 59. Elementos de formato para la función TIMESTAMP_FORMAT

Elemento de formato	Componentes relacionados de una indicación de fecha y hora	Descripción
AM o PM	hora	Indicador de meridiano (mañana o tarde) sin puntos. Este elemento de formato depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
A.M. o P.M.	hora	Indicador de meridiano (mañana o tarde) con puntos. Este elemento de formato utiliza las series exactas "A.M." o "P.M." y es independiente del nombre vigente del entorno local.
DAY, Day o day	ninguna	Nombre del día todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
DY, Dy o dy	ninguna	Nombre abreviado del día todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
D	ninguna	Día de la semana (1-7). El primer día de la semana depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
DD	día	Día del mes (01-31).
DDD	mes, día	Día del año (001-366).

Tabla 59. Elementos de formato para la función TIMESTAMP_FORMAT (continuación)

Elemento de formato	Componentes relacionados de una indicación de fecha y hora	Descripción
FF o FF*n*	Segundos fraccionarios	Segundos fraccionarios (0-999999999999). El número *n* se utiliza para especificar el número de dígitos esperados en la *expresión-serie*. Los valores válidos para *n* son del 1 al 12 sin ceros iniciales. Especificar FF equivale a especificar FF6. Si FF es el último elemento del formato y el número de dígitos de los segundos fraccionarios es inferior a lo que especifica el formato, no se rellena ningún dígito de la derecha de los dígitos especificados.
HH	hora	HH se comporta igual que HH12.
HH12	hora	Hora del día (01-12) en formato de 12 horas. AM es el indicador de meridiano por omisión.
HH24	hora	Hora del día (00-24) en formato de 24 horas.
J	año, mes y día	Día del calendario juliano (número de días desde el 1 de enero, 4713 AC).
MI	minuto	Minuto (00-59).
MM	mes	Mes (01-12).
MONTH, Month o month	mes	Nombre del mes todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
MON, Mon o mon	mes	Nombre abreviado del mes todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
NNNNNN	microsegundos	Microsegundos (000000-999999). Igual que FF6.

Tabla 59. Elementos de formato para la función TIMESTAMP_FORMAT (continuación)

Elemento de formato	Componentes relacionados de una indicación de fecha y hora	Descripción
RR	año	Dos últimos dígitos del año ajustado (00-99).
RRRR	año	Año ajustado de 4 dígitos (0000-9999).
SS	segundos	Segundos (00-59).
SSSSS	horas, minutos y segundos	Segundos desde la medianoche anterior (00000-86400).
S	año	Último dígito del año (0-9). Se utilizan los tres primeros dígitos del año actual para determinar el año completo de 4 dígitos.
YY	año	Dos últimos dígitos del año (00-99). Se utilizan los dos primeros dígitos del año actual para determinar el año completo de 4 dígitos.
YYY	año	Tres últimos dígitos del año (000-999). Se utiliza el primer dígito del año actual para determinar el año completo de 4 dígitos.
YYYY	año	Año de 4 dígitos (0000-9999).

Nota: Los elementos de formato de la Tabla 59 en la página 599 no son sensibles a mayúsculas y minúsculas, salvo los siguientes:

- AM, PM
- A.M., P.M.
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

Los elementos de formato DAY, Day, day, DY, Dy, dy y D no influyen en ningún componente del indicador de fecha y hora resultante. Sin embargo, debe haberse especificado un valor correcto para cualquiera de estos elementos de formato para la combinación de los componentes de año, mes y día de la indicación de fecha y hora resultante (SQLSTATE 22007). Por ejemplo, si suponemos que se utiliza un valor de 'en_US' para *nombre-entorno-local*, el valor 'Monday 2008-10-06' para *expresión-serie* será válido para el valor de 'Day YYYY-MM-DD'. Sin embargo, el valor 'Tuesday 2008-10-06' para *expresión-serie* daría como resultado un error para la misma *serie-formato*.

Los elementos de formato RR y RRRR se pueden utilizar para modificar cómo se debe interpretar una especificación de un año ajustando el valor para producir un valor de 2 dígitos o un valor de 4 dígitos en función de los dos dígitos situados más a la izquierda del año actual de acuerdo con la tabla

siguiente.

Dos últimos dígitos del año actual	Año de dos dígitos en *expresión-serie*	Dos primeros dígitos del componente de año de la indicación de fecha y hora
00-50	00-49	Dos primeros dígitos del año actual
51-99	00-49	Dos primeros dígitos del año actual + 1
00-50	50-99	Dos primeros dígitos del año actual - 1
51-99	50-99	Dos primeros dígitos del año actual

Por ejemplo, si el año actual es 2007, '86' con el formato 'RR' significa 1986, pero si el año actual es 2052, significa 2086.

Se utilizan los valores por omisión siguientes cuando una *serie-formato* no incluye un elemento de formato para uno de los siguientes componentes de una indicación de fecha y hora:

Componente de indicación de fecha y hora	Valor por omisión
año	año actual, en 4 dígitos
mes	mes actual, en 2 dígitos
día	01 (primer día del mes)
hora	00
minuto	00
segundo	00
segundos fraccionarios	número de ceros que corresponden a la precisión de la indicación de fecha y hora del resultado.

Se pueden especificar ceros iniciales para cualquier componente del valor de indicación de fecha y hora (es decir mes, día, hora, minutos, segundos) que no tenga el número máximo de dígitos significativos para el elemento de formato correspondiente en *serie-formato*.

Una subserie de *expresión-serie* que representa un componente de una indicación de fecha y hora (por ejemplo año, mes, día, hora, minutos, segundos) puede incluir un número menor de dígitos que el número máximo para dicho componente de la indicación de fecha y hora indicada por el elemento de formato correspondiente. Los dígitos que faltan toman por omisión cero. Por ejemplo, con una *serie-formato* de 'YYYY-MM-DD HH24:MI:SS', un valor de entrada de '999-3-9 5:7:2' producirá el mismo resultado que '0999-03-09 05:07:02'.

Si no se especifica *serie-formato*, la *expresión-serie* se interpretará mediante un formato por omisión basado en el valor del registro especial CURRENT LOCALE LC_TIME.

constante-precisión
Constante de entero que especifica la precisión de la indicación de fecha y hora del resultado. El valor debe estar comprendido entre 0 y 12. Si no se especifica, el valor por omisión de la precisión de la indicación de fecha y hora es 6.

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local utilizado para los elementos de formato siguientes:

- AM, PM
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery" en la publicación *Globalization Guide* . Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función es un valor TIMESTAMP con una precisión basada en *constante-precisión*. Si alguno de los dos primeros argumentos puede ser nulo, el resultado podrá ser nulo; si alguno de los dos primeros argumentos es nulo, el resultado será el valor nulo.

Notas

- ***Calendario juliano y gregoriano***: esta función tiene en cuenta la transición del calendario juliano al calendario gregoriano el 15 de octubre de 1582.
- ***Determinismo:*** TIMESTAMP_FORMAT es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME o CURRENT TIMESTAMP.
  - Si la *serie-formato* no se especifica explícitamente, o si *nombre-entorno-local* no se especifica explícitamente y se da una de las condiciones siguientes:
    - *serie-formato* no es una constante
    - *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local
    - *serie-formato* es una constante y no incluye un elemento de formato que defina enteramente el año (es decir, J o AAAA), por lo que utiliza el valor del año actual
    - *serie-formato* es una constante y no incluye un elemento de formato que defina enteramente el mes (es decir, J, MM, MONTH o MON), por lo que utiliza el valor del mes actual

  Estas invocaciones que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales (SQLSTATE 42621 o 428EC).
- ***Alternativas de sintaxis:*** TO_DATE y TO_TIMESTAMP son sinónimos de TIMESTAMP_FORMAT.

Ejemplos

- Inserte una fila en la tabla IN_TRAY con una indicación de fecha y hora de recepción que sea igual a un segundo antes que el principio del año 2000 (31 de diciembre de 1999 a las 23:59:59).

```
INSERT INTO IN_TRAY (RECEIVED)
  VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
    'AAAA-MM-DD HH24:MI:SS'))
```

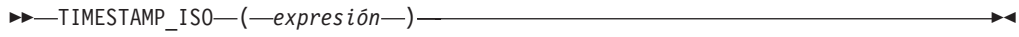
- Una aplicación recibe series de información de fecha en una variable denominada INDATEVAR. Este valor no se formatea de forma estricta y puede incluir dos o cuatro dígitos para años y uno o dos dígitos para meses y días. Los componentes de fecha pueden estar separados por caracteres de signo menos (-) o por una barra inclinada (/) y se espera que guarden el orden día, mes y año. La información de hora consta de horas (en formato de 24 horas) y minutos y se suele separar mediante dos puntos. '15/12/98 13:48' y '9-3-2004 8:02' son valores de ejemplo. Inserte valores de este tipo en la tabla IN_TRAY.

  ```
  INSERT INTO IN_TRAY (RECEIVED)
    VALUES (TIMESTAMP_FORMAT(:INDATEVAR,
      'DD/MM/RRRR HH24:MI'))
  ```

  El uso de RRRR en el formato permite valores de año de 2 y 4 dígitos y asigna los dos primeros dígitos que faltan basándose en el año actual. Si se utiliza YYYY, los valores de entrada con un año de 2 dígitos tendrá ceros iniciales. El separador de barra inclinada también permite el carácter de signo menos. Suponiendo un año actual de 2007, las indicaciones de fecha y hora resultantes de los valores de ejemplo son:

  ```
  '15/12/98 13:48' --> 1998-12-15-13.48.00.000000
  '9-3-2004 8:02'  --> 2004-03-09-08.02.00.000000
  ```

TIMESTAMP_ISO



El esquema es SYSFUN.

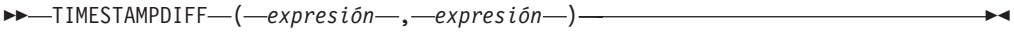
Devuelve un valor de indicación de fecha y hora basado en un argumento de fecha, de hora o de indicación de fecha y hora. Si el argumento es un valor DATE, insertará ceros para todos los elementos de la hora. Si el argumento es un valor TIME, insertará el valor del registro especial CURRENT DATE para los elementos de fecha y cero para el elemento de fracción de hora.

La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR, VARCHAR, DATE, TIME o TIMESTAMP. En una base de datos Unicode, si un argumento proporcionado tiene un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a una serie de caracteres antes de evaluar la función. Una expresión de serie debe devolver una representación de serie de caracteres válida de una fecha o de una indicación de fecha y hora.

La función TIMESTAMP_ISO se define generalmente como determinista. Si el primer argumento tiene el tipo de datos TIME, la función no es determinista ya que CURRENT DATE se utiliza para la parte de la fecha de un valor de indicación de fecha y hora.

El resultado de la función es un valor TIMESTAMP. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

TIMESTAMPDIFF



El esquema es SYSFUN.

Devuelve un número estimado de intervalos del tipo definido por el primer argumento, basándose en la diferencia entre dos indicaciones de la hora.

El primer argumento puede ser INTEGER o SMALLINT. Los valores válidos de intervalo (el primer argumento) son:

1 Fracciones de segundo

2 Segundos

4 Minutos

8 Horas

16 Días

32 Semanas

64 Meses

128 Trimestres

256 Años

El segundo argumento es el resultado de restar dos indicaciones de fecha y hora y convertir el resultado a CHAR(22). El valor de la serie no debe tener más de 6 dígitos a la derecha de la coma decimal. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Se pueden utilizar las suposiciones siguientes al estimar una diferencia:

- Hay 365 días en un año.
- Hay 30 días en un mes.
- Hay 24 horas en un día.
- Hay 60 minutos en una hora.
- Hay 60 segundos en un minuto.

Estas suposiciones se utilizan al convertir la información del segundo argumento, que es una duración de indicación de fecha y hora, al tipo de intervalo especificado en el primer argumento. La estimación que se devuelve puede variar en unos días. Por ejemplo, si se pide el número de días (intervalo 16) para la diferencia entre '1997-03-01-00.00.00' y '1997-02-01-00.00.00', el resultado es 30. Esto es debido a que la diferencia entre las indicaciones de fecha y hora es de 1 mes y se aplica la suposición de que hay 30 días en un mes.

Ejemplo:

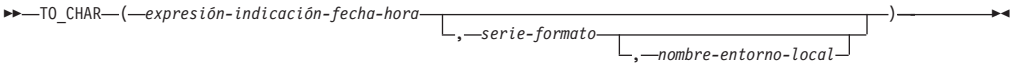
El ejemplo siguiente devuelve 4277, el número de minutos entre dos indicaciones de fecha y hora:

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP('2001-09-29-11.25.42.483219') -
  TIMESTAMP('2001-09-26-12.07.58.065497')))
```

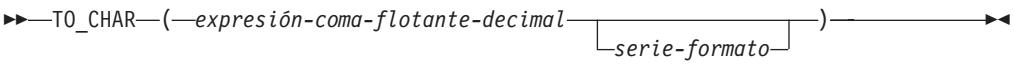
TO_CHAR

De carácter a varchar

De indicación de fecha y hora a varchar:



De coma flotante decimal a varchar:

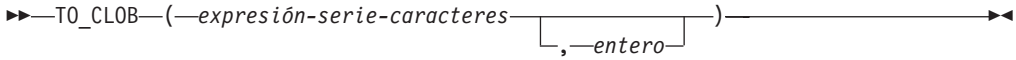


El esquema es SYSIBM.

La función TO_CHAR devuelve una representación de caracteres de una expresión de entrada a la que se le ha dado formato con una plantilla de caracteres.

La función escalar TO_CHAR es un sinónimo de la función escalar VARCHAR_FORMAT.

TO_CLOB

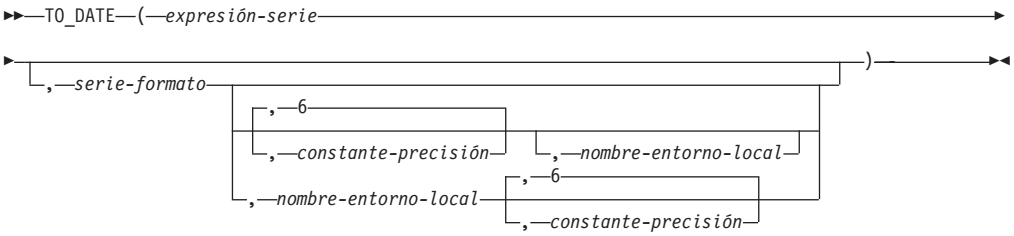


El esquema es SYSIBM.

La función TO_CLOB devuelve una representación CLOB de un tipo de serie de caracteres.

La función escalar TO_CLOB es un sinónimo de la función escalar CLOB.

TO_DATE

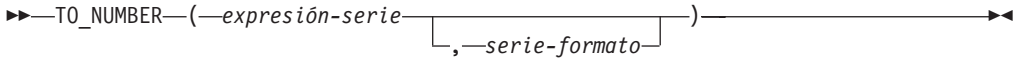


El esquema es SYSIBM.

La función TO_DATE devuelve una indicación de fecha y hora que se basa en la interpretación de la serie de entrada utilizando el formato especificado.

La función escalar TO_DATE es un sinónimo de la función escalar TIMESTAMP_FORMAT.

TO_NUMBER

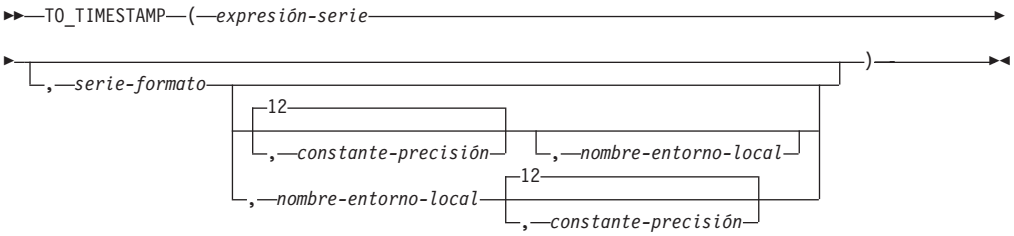


El esquema es SYSIBM.

La función TO_NUMBER devuelve un valor DECFLOAT(34) que se basa en la interpretación de la serie de entrada utilizando el formato especificado.

La función escalar TO_NUMBER es un sinónimo de la función escalar DECFLOAT_FORMAT.

TO_TIMESTAMP

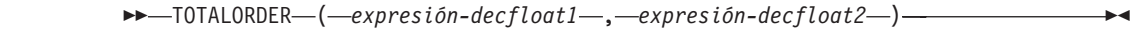


El esquema es SYSIBM.

La función TO_TIMESTAMP devuelve una indicación de fecha y hora basada en la interpretación de la serie de entrada utilizando el formato especificado.

La función escalar TO_TIMESTAMP es un sinónimo de la función escalar TIMESTAMP_FORMAT, con la diferencia de que el valor por omisión de *constante-precisión* es 12.

TOTALORDER



El esquema es SYSIBM.

La función TOTALORDER devuelve un valor SMALLINT de -1, 0 ó 1 que indica el orden de comparación de dos argumentos.

expresión-decfloat1
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es DECFLOAT(34), se convierte de forma lógica a DECFLOAT(34) para procesarse.

expresión-decfloat2
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es un valor de coma flotante decimal, se convertirá a DECFLOAT(34) para su proceso.

La comparación numérica es exacta y el resultado se determina para operandos finitos como si el rango y la precisión fueran ilimitados. Una condición de desbordamiento o subdesbordamiento no puede producirse.

Si un valor es DECFLOAT(16) y el otro es DECFLOAT(34), el valor DECFLOAT(16) se convierte a DECFLOAT(34) antes de que se efectúe la comparación.

La semántica de la función TOTALORDER se basa en las normas de predicado de orden total de IEEE 754R. TOTALORDER devuelve los valores siguientes:

- -1 si *expresión-decfloat1* es inferior en orden en comparación con *expresión-decfloat2*
- 0 si tanto *expresión-decfloat1* como *expresión-decfloat2* tienen el mismo orden
- -1 si *expresión-decfloat1* es superior en orden en comparación con *expresión-decfloat2*

El orden de los valores especiales y números finitos es como sigue:

```
-NAN<-SNAN<-INFINITY<-0.10<-0.100<-0<0<0.100<0.10<INFINITY<SNAN<NAN
```

El resultado de la función es un valor SMALLINT. Si el argumento puede ser nulo, el resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Ejemplos:

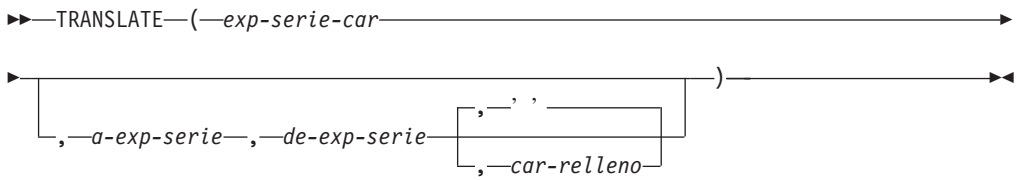
- Los ejemplos siguientes muestran el uso de la función TOTALORDER para comparar valores de coma flotante decimal:

```
TOTALORDER(-INFINITY, -INFINITY) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.0)) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.00)) = -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-0.5)) = -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(0.5)) = -1
TOTALORDER(DECFLOAT(-1.0), INFINITY) = -1
TOTALORDER(DECFLOAT(-1.0), SNAN) = -1
TOTALORDER(DECFLOAT(-1.0), NAN) = -1
TOTALORDER(NAN, DECFLOAT(-1.0)) = 1
TOTALORDER(-NAN, -NAN) = 0
TOTALORDER(-SNAN, -SNAN) = 0
TOTALORDER(NAN, NAN) = 0
TOTALORDER(SNAN, SNAN) = 0
```

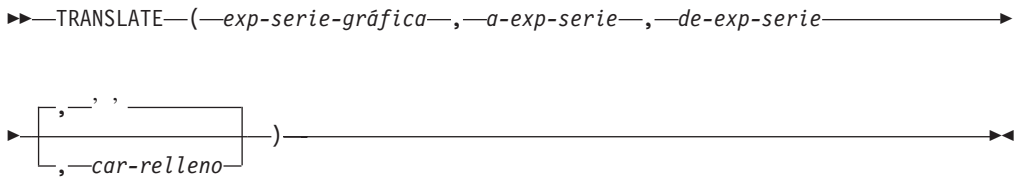
```
TOTALORDER(-1.0, -1.0) = 0
TOTALORDER(-1.0, -1.00) = -1
TOTALORDER(-1.0, -0.5) = -1
TOTALORDER(-1.0, 0.5) = -1
TOTALORDER(-1.0, INFINITY) = -1
TOTALORDER(-1.0, SNAN) = -1
TOTALORDER(-1.0, NAN) = -1
```

TRANSLATE

Expresión de serie de caracteres:



expresión de serie gráfica:



El esquema es SYSIBM.

La función TRANSLATE devuelve un valor ene l que uno o más caracteres de una expresión de serie es posible que se hayan convertido a otros caracteres.

La función convierte todos los caracteres de *exp-serie-car* o *exp-serie-gráfica* que también se producen en *de-exp-serie* en los caracteres correspondientes en *a-exp-serie* o, si no existen caracteres correspondientes, los convierte en el carácter de relleno especificado por *exp-car-relleno*.

exp-serie-car **o** *exp-serie-gráfica*
Especifica la serie que debe convertirse.La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

a-exp-serie
Especifica una serie de caracteres a la que se convertirán determinados caracteres de *exp-serie-car*.

La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función. Si no se especifica ningún valor para *a-exp-serie*, y el tipo de datos no es gráfico, todos los caracteres de *exp-serie-car* se pasarán a mayúsculas; es decir, todos los caracteres a-z se convertirán en los caracteres A-Z, y los demás caracteres se convertirán en sus equivalentes en mayúsculas, si existen. Por ejemplo, en la página de códigos 850, é se correlaciona con É, pero ÿ no se correlaciona, porque la página de códigos 850 no incluye Ÿ. Si la longitud de elemento de código del carácter de resultado no es la misma longitud que la del elemento de código del carácter fuente, el carácter fuente no se convierte.

de-exp-serie
Especifica una serie de caracteres que, si se encuentra en *exp-serie-car*, se convertirá en el carácter correspondiente en *a-exp-serie*.

La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función. Si *de-exp-serie* contiene caracteres duplicados, se utilizará el primero que se encuentre y los duplicados se omitirán. Si *a-exp-serie* es más largo que *de-exp-serie*, no se tendrán en cuenta los caracteres que sobren. Si se especifica *a-exp-serie*, también debe especificarse *de-exp-serie*.

exp-car-relleno
Especifica un único carácter que se utilizará para rellenar *a-exp-serie* si la longitud de *a-exp-serie* es inferior a la de *de-exp-serie*. La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función. El valor debe tener un atributo de longitud igual a uno. Si no se especifica un valor, se adoptará un carácter en blanco de un byte.

Con *exp-serie-gráfica*, sólo *exp-car-relleno* es opcional (si no se especifica ningún valor, se presupone el carácter en blanco de doble byte), y cada argumento, incluido el carácter de relleno, debe corresponder al tipo de datos gráfico.

El tipo de datos y la página de códigos del resultado son los mismos que el tipo de datos y la página de códigos del primer argumento. Si el primer argumento es una variable de lenguaje principal, la página de códigos del resultado es una página de códigos la base de datos. Cada argumento, que no sea el primer argumento, se convierte a la página de códigos de resultado a menos que éste o el primer argumento se haya definido como FOR BIT DATA, en cuyo caso no se realiza ninguna conversión.

En una base de datos Unicode donde los datos de tipo carácter y gráfico se consideran tipos de datos equivalentes, existen las siguientes excepciones:

- La página de códigos del resultado es 1208 si cualquier argumento, excepto el primer argumento, es FOR BIT DATA.
- La página de códigos del resultado es la página de códigos que aparece más frecuentemente en el conjunto de argumentos, cuando ningún argumento es FOR BIT DATA.
- La página de códigos del resultado es 1200 cuando dos páginas de códigos aparecen con la misma frecuencia en el conjunto de argumentos, cuando ningún argumento es FOR BIT DATA.

El atributo de longitud del resultado es el mismo que el del primer argumento. Si cualquier argumento puede ser nulo, el resultado puede ser nulo. Si ningún argumento es nulo, el resultado es el valor nulo.

Si los argumentos so tipos de datos CHAR o VARCHAR, los caracteres correspondientes en *a-exp-serie* y *de-exp-serie* deben tener el mismo número de bytes. Por ejemplo, no es válido convertir un carácter de un único byte en un carácter de múltiples bytes, ni a la inversa. El argumento *exp-car-relleno* no puede ser el primer byte de un carácter de múltiples bytes (SQLSTATE 42815).

Los caracteres se comparan utilizando una comparación binaria. No se utiliza la clasificación de base de datos.

Si sólo se especifica *exp-serie-car*, los caracteres de un único byte se convertirán a mayúsculas, y los caracteres de múltiples bytes no sufrirán ningún cambio.

Ejemplos:

- Supongamos que la variable del lenguaje principal SITE (VARCHAR(30)) tiene el valor 'Hanauma Bay'.

  ```
  TRANSLATE(:SITE)
  ```

  Devuelve el valor 'HANAUMA BAY'.

  ```
  TRANSLATE(:SITE, 'j', 'B')
  ```

  Devuelve el valor 'Hanauma jay'.

  ```
  TRANSLATE(:SITE,'ei','aa')
  ```

  Devuelve el valor 'Heneume Bey'.

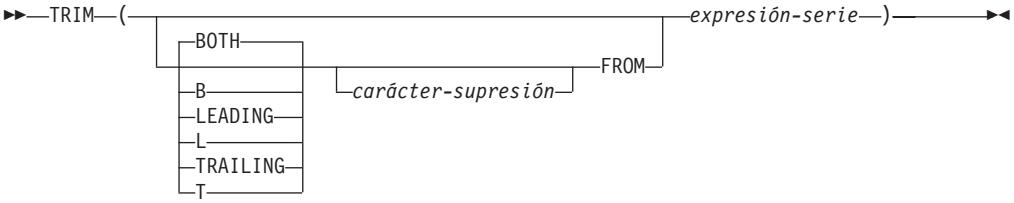
  ```
  TRANSLATE(:SITE, 'bA', 'Bay', '%')
  ```

  Devuelve el valor 'HAnAumA bA%'.

  ```
  TRANSLATE(:SITE,'r','Bu')
  ```

  Devuelve el valor 'Hana ma ray'.

TRIM



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función.

La función TRIM suprime blancos o las apariciones de otro carácter especificado del final o del principio de una expresión de serie.

BOTH, LEADING o TRAILING
Especifica si se suprimen los caracteres del principio, del final o de ambos extremos de la expresión de serie. Si no se especifica este argumento, se suprimen los caracteres del final y del principio de la serie.

carácter-supresión
Una constante de un sólo carácter que especifica el carácter que se ha de suprimir. El *carácter-supresión* puede ser cualquier carácter cuya codificación UTF-32 sea un carácter individual o un valor numérico de un solo dígito. Se compara la representación binaria del carácter.

Si no se especifica el *carácter-supresión* y:

- Si la *expresión-serie* es una serie de gráficos DBCS, el valor por omisión para *carácter-supresión* es un blanco DBCS, cuyo elemento de código depende de la página de códigos de base de datos
- Si la *expresión-serie* es una serie gráfica UCS-2, el *carácter-supresión* por omisión es un espacio en blanco UCS-2 (X'0020')
- De lo contrario, el *carácter-supresión* por omisión es un espacio en blanco SBCS (X'20')

FROM *expresión-serie*
La expresión debe devolver un valor que sea un tipo de datos de indicación de fecha y hora, numérico, CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos CHAR, VARCHAR, GRAPHIC o VARGRAPHIC, se convierte implícitamente a VARCHAR antes de evaluar la función.

El resultado es una serie de longitud variable con la misma longitud máxima que el atributo de longitud de la *expresión-serie*. La longitud real del resultado es la longitud de la *expresión-serie* menos el número de bytes que se ha de suprimir. Si se suprimen todos los caracteres, el resultado es una serie de longitud variable vacía. La página de códigos del resultado es la misma que la página de códigos de la *expresión-serie*.

Ejemplos:

- Supongamos que la variable HELLO de sistema principal de tipo CHAR(9) tiene un valor 'Hello'.

```
SELECT TRIM(:HELLO),
  TRIM(TRAILING FROM :HELLO)
FROM SYSIBM.SYSDUMMY1
```

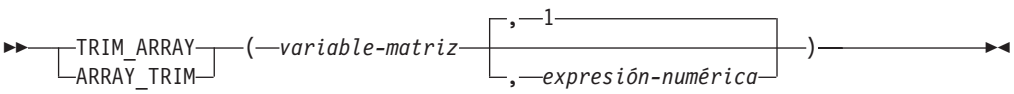
devuelve los valores 'Hello' y 'Hello', respectivamente.

- Supongamos que la variable BALANCE de sistema principal de tipo CHAR(9) tiene un valor '000345.50'.

```
SELECT TRIM(L '0' FROM :BALANCE),
FROM SYSIBM.SYSDUMMY1
```

devuelve el valor '345.50'.

TRIM_ARRAY



El esquema es SYSIBM.

La función TRIM_ARRAY suprime elementos del final de una matriz.

variable-matriz
Una variable de SQL, un parámetro de SQL o una variable global de un tipo de matriz común, o una especificación CAST de un marcador de parámetro a un tipo de matriz común. Un tipo de datos de matriz asociativa no puede especificarse (SQLSTATE 42884).

expresión-numérica
Especifica el número de elementos recortados del final de la matriz. *expresión-numérica* puede ser cualquier tipo de datos numérico con un valor que puede convertirse en INTEGER. El valor de *expresión-numérica* debe estar entre 0 y la cardinalidad de variable-matriz (SQLSTATE 2202E). El valor por omisión es 1.

La función devolverá un valor con el mismo tipo de matriz que *variable-matriz* pero con la cardinalidad reducida por el valor de INTEGER(*expresión-numérica*). El resultado puede ser nulo; el resultado es el valor nulo.

Normas

No se da soporte a la función TRIM_ARRAY en las matrices asociativas (SQLSTATE 42884).

La función TRIM_ARRAY solo puede utilizarse a la derecha de una sentencia de asignación en contextos en los que se da soporte a las matrices (SQLSTATE 42884).

Ejemplos

- Elimine el último elemento de la variable de matriz RECENT_CALLS.

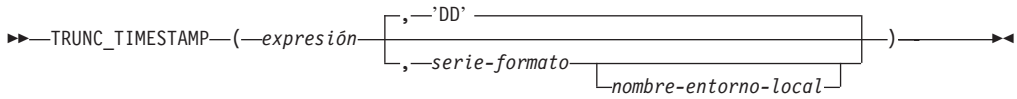
```
SET RECENT_CALLS = TRIM_ARRAY(RECENT_CALLS, 1)
```

- Asigne solamente los dos primeros elementos de la variable de matriz SPECIALNUMBERS a la variable de matriz de SQL EULER_CONST:

```
SET EULER_CONST = TRIM_ARRAY(SPECIALNUMBERS, 8)
```

  El resultado es que a EULER_CONST se le asignará una matriz con dos elementos: el valor del primer elemento es 2.71828183 y el valor del segundo elemento es el valor nulo.

TRUNC_TIMESTAMP



El esquema es SYSIBM.

La función escalar TRUNC_TIMESTAMP devuelve un valor TIMESTAMP que es la *expresión* truncada en la unidad que especifica *serie-formato*. Si no se especifica la *serie-formato*, la *expresión* se truncará en el día más próximo, como si se hubiera especificado 'DD' para la *serie-formato*.

expresión
Una expresión que devuelve un valor de uno de los siguientes tipos de datos incorporados: valor DATE o valor TIMESTAMP.

serie-formato
Una expresión que devuelve un tipo de datos de serie de caracteres incorporada con una longitud real que no supere los 254 bytes. El elemento de formato de *serie-formato* especifica cómo debe truncarse *expresión*. Por ejemplo, si *serie-formato* es 'DD', una indicación de fecha y hora que se represente mediante una *expresión* se trunca en el día más cercano. Los espacios en blanco iniciales y finales se eliminan de la serie y la subserie resultante debe ser un elemento de formato válido para una indicación de fecha y hora (SQLSTATE 22007). El valor por omisión es 'DD'.

Los valores permitidos para *serie-formato* se listan en la tabla de elementos de formato que se encuentra en la descripción de la función ROUND.

nombre-entorno-local
Constante de caracteres que especifica el entorno local que se utiliza para determinar el primer día de la semana cuando se utilizan los valores de modelo de formato DAY, DY o D. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función es TIMESTAMP con la misma precisión de indicación de fecha y hora que *expresión*. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

El resultado de la función es TIMESTAMP con una precisión de indicación de fecha y hora de:

- *p* cuando el tipo de datos de expresión es TIMESTAMP(*p*)
- 0 cuando el tipo de datos de expresión es DATE
- 6 de lo contrario.

El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

Notas

- **Determinismo:** TRUNC_TIMESTAMP es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME.

– Truncación de un valor de fecha o de indicación de fecha y hora cuando no se especifica explícitamente *nombre-entorno-local* y cuando una de las afirmaciones siguientes es verdadera:
  - *serie-formato* no es una constante
  - *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local

  Las invocaciones de la función que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales.

Ejemplos

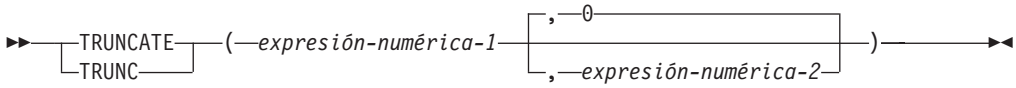
- Establecer la variable del lenguaje principal TRNK_TMSTMP con el año actual redondeado al valor de año más cercano.

```
SET :TRNK_TMSTMP = TRUNC_TIMESTAMP('2000-03-14-17.30.00', 'YEAR');
```

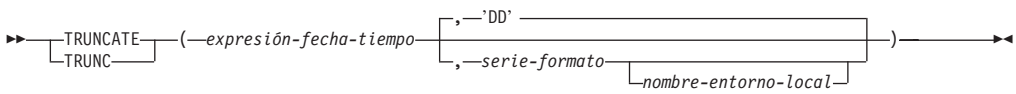
  La variable del lenguaje principal TRNK_TMSTMP se establece con el valor 2000-01-01-00.00.00.000000.

TRUNCATE o TRUNC

TRUNCATE numérico:



TRUNCATE fecha-hora:



El esquema es SYSIBM. La versión de SYSFUN de la función TRUNCATE numérico sigue estando disponible.

La función TRUNCATE devuelve un valor truncado de:

- Un número, truncado en el número de posiciones especificado, a la derecha o izquierda de la coma decimal, si el resultado del primer argumento es un valor numérico.
- Un valor de fecha y hora, truncado en la unidad que especifica *serie-formato*, si el primer argumento es un valor DATE, TIME o TIMESTAMP

TRUNCATE numérico

Si *expresión-numérica-1* tiene un tipo de datos numérico, la función TRUNCATE devuelve la *expresión-numérica-1* truncada en *expresión-numérica-2* posiciones a la derecha de la coma decimal si *expresión-numérica-2* es positiva o a la izquierda de la coma decimal si *expresión-numérica-2* es cero o negativa. Si no se especifica *expresión-numérica-2*, *expresión-numérica-1* se trunca en cero posiciones a la izquierda de la coma decimal.

expresión-numérica-1

Una expresión que debe devolver un valor que sea un tipo de datos numérico o CHAR, VARCHAR, GRAPHIC o VARGRAPHIC incorporado. Si el valor no es un tipo de datos numérico, se convierte de forma implícita en DECFLOAT(34) antes de evaluar la función.

Si la expresión es un tipo de datos de coma flotante decimal, la modalidad de redondeo DECFLOAT no se utilizará. El comportamiento de redondeo de TRUNCATE corresponde a un valor de ROUND_DOWN. Si se desea otro comportamiento de redondeo, utilice la función QUANTIZE.

expresión-numérica-2

Expresión que devuelve un valor que es un tipo de datos incorporado. Si el valor no es del tipo INTEGER, se convierte de forma implícita en INTEGER antes de evaluar la función.

Si *expresión-numérica-2* no es negativa, *expresión-numérica-1* se trunca en el valor absoluto de *expresión-numérica-2* posiciones a la derecha de la coma decimal.

Si *expresión-numérica-2* es negativa, *expresión-numérica-1* se trunca en el valor absoluto de *expresión-numérica-2* +1 número de posiciones a la izquierda de la coma decimal. Si el valor absoluto de una

expresión-numérica-2 es superior al número de dígitos a la izquierda de la coma decimal, el resultado es 0. Por ejemplo:

```
TRUNCATE(748.58,-4) = 0
```

El tipo de datos, la longitud y los atributos de escala del resultado son iguales que el tipo de datos, la longitud y los atributos de escala del primer argumento.

Si algún argumento puede ser nulo, el resultado puede ser nulo. Si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

TRUNCATE fecha-hora

Si *expresión-fecha-hora* tiene un tipo de datos de fecha y hora, la función TRUNCATE devuelve *expresión-fecha-hora* redondeada a la unidad especificada por la *serie-formato*. Si no se especifica *serie-formato*, *expresión-fecha-hora* se trunca en el día más cercano, como si se especificara 'DD' para *serie-formato*.

expresión-fecha-hora
Una expresión que debe devolver un valor que sea de tipo DATE, TIME o TIMESTAMP. Las representaciones de serie de estos tipos de datos no reciben soporte y deben convertirse explícitamente en un valor DATE, TIME o TIMESTAMP para poder utilizarlas con esta función; de forma alternativa, puede utilizar la función TRUNC_TIMESTAMP para una representación de serie de una fecha o indicación de fecha y hora.

serie-formato
Una expresión que devuelve un tipo de datos de serie de caracteres incorporada con una longitud real que no supere los 254 bytes. El elemento de formato de *serie-formato* especifica cómo debe truncarse *expresión-fecha-hora*. Por ejemplo, si *serie-formato* es 'DD', una indicación de fecha y hora que se represente mediante una *expresión-fecha-hora*, se trunca en el día más cercano. Los espacios en blanco iniciales y finales se eliminan de la serie y la subserie resultante debe ser un elemento de formato válido para el tipo de *expresión-fecha-hora* (SQLSTATE 22007). El valor por omisión es 'DD', que no se puede utilizar si el tipo de datos de *expresión-fecha-hora* es TIME.

Los valores permitidos para *serie-formato* se listan en la tabla de elementos de formato que se encuentra en la descripción de la función ROUND.

nombre-entorno-local
Una constante de caracteres que especifica el entorno local que se utiliza para determinar el primer día de la semana cuando se utilizan los valores de elemento de formato DAY, DY o D. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery". Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado de la función tiene el mismo tipo de datos que *expresión-fecha-hora*. El resultado puede ser nulo; si cualquier argumento es nulo, el resultado es el valor nulo.

El tipo de datos y el atributo de longitud del resultado de la función tiene el mismo que el tipo de datos y el atributo de longitud del primer argumento.

El resultado puede ser nulo si el argumento puede ser nulo o si el argumento no es un número de coma flotante decimal y la base de datos se ha configurado con **dft_sqlmathwarn** establecido en YES; el resultado es el valor nulo si el argumento es nulo.

Notas

- **Determinismo:** TRUNCATE es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME.
  - Truncación de un valor de fecha y hora cuando no se especifica explícitamente *nombre-entorno-local* y cuando una de las afirmaciones siguientes es verdadera:
    - *serie-formato* no es una constante
    - *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local

    Las invocaciones de la función que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales.

Ejemplos

- Utilizando la tabla EMPLOYEE, calcule el salario medio mensual del empleado mejor pagado. Trunque el resultado dos posiciones a la derecha de la coma decimal.

```
SELECT TRUNCATE(MAX(SALARY)/12,2)
  FROM EMPLOYEE;
```

  Supongamos que el empleado mejor pagado gana 52750 euros al año; el ejemplo devuelve 4395,83.

- Muestre el número 873,726 truncado 2, 1, 0, -1 y -2 posiciones decimales, respectivamente.

```
VALUES (
  TRUNCATE(873.726,2),
  TRUNCATE(873.726,1),
  TRUNCATE(873.726,0),
  TRUNCATE(873.726,-1),
  TRUNCATE(873.726,-2),
  TRUNCATE(873.726,-3) );
```

  Este ejemplo devuelve 873.720, 873.700, 873.000, 870.000, 800.000 y 0.000.

- Visualice el número de coma flotante decimal 873.726 truncado 0 decimales.

```
VALUES(TRUNCATE(DECFLOAT(873.726),0))
```

  Devuelve el valor 873.

- Establecer la variable vTRNK_DT con la fecha de entrada redondeada al valor de mes más cercano.

```
SET vTRNK_DT = TRUNC(DATE('2000-08-16'), 'MONTH');
```

  El valor establecido es 2000-08-01.

- Establecer la variable del lenguaje principal TRNK_TMSTMP con el año actual redondeado al valor de año más cercano.

```
SET :TRNK_TMSTMP = TRUNCATE('2000-03-14-17.30.00'), 'YEAR');
```

El valor establecido es 2000-01-01-00.00.00.000000.

TYPE_ID

El esquema es SYSIBM.

La función TYPE_ID devuelve el identificador de tipo interno del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario).

El tipo de datos del resultado de la función es INTEGER. Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo.

El valor devuelto por la función TYPE_ID no es portátil a través de las bases de datos. El valor puede ser diferente aunque el esquema de tipo y el nombre de tipo del tipo de datos dinámico sean iguales. Cuando especifique el código para permitir la portabilidad, utilice las funciones TYPE_SCHEMA y TYPE_NAME para determinar el esquema de tipo y el nombre de tipo.

Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el identificador de tipo interno de la fila que corresponda a la referencia.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_ID(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  La función DEREF se utiliza para devolver el objeto correspondiente a la fila.

TYPE_NAME

El esquema es SYSIBM.

La función TYPE_NAME devuelve el nombre no calificado del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario. (Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario).

El tipo de datos del resultado de la función es VARCHAR(18). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_SCHEMA para determinar el nombre de esquema del nombre de tipo devuelto por TYPE_NAME.

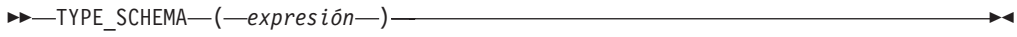
Ejemplos:

- Existe una jerarquía de tablas que tiene una tabla raíz EMPLOYEE de tipo EMP y una subtabla MANAGER de tipo MGR. Otra tabla ACTIVITIES incluye una columna denominada WHO_RESPONSIBLE que se define como REF(EMP) SCOPE EMPLOYEE. Para cada referencia de ACTIVITIES, visualice el tipo de la fila que corresponda a la referencia.

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
      TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  La función DEREF se utiliza para devolver el objeto correspondiente a la fila.

TYPE_SCHEMA



El esquema es SYSIBM.

La función TYPE_SCHEMA devuelve el nombre de esquema del tipo de datos dinámico de la *expresión*.

El argumento debe ser un tipo estructurado definido por el usuario. Esta función no puede utilizarse como una función fuente cuando se crea una función definida por el usuario. Como acepta cualquier tipo de datos estructurado como argumento, no es necesario crear signaturas adicionales para dar soporte a los diferentes tipos definidos por el usuario.

El tipo de datos del resultado de la función es VARCHAR(128). Si *expresión* puede tener un valor nulo, el resultado puede ser nulo; si *expresión* tiene un valor nulo, el resultado es el valor nulo. Utilice la función TYPE_NAME para determinar el nombre de tipo asociado con el nombre de esquema devuelto por TYPE_SCHEMA.

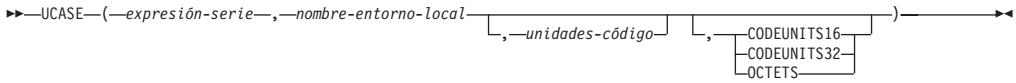
UCASE

El esquema es SYSIBM.

La función UCASE es idéntica a la función TRANSLATE excepto en que sólo se especifica el primer argumento (*exp-serie-car*).

UCASE es sinónimo de UPPER.

UCASE (sensible al entorno local)



El esquema es SYSIBM.

La función UCASE devuelve una serie en la que todos los caracteres se han convertido a mayúsculas utilizando las normas asociadas con el entorno local especificado.

UCASE es sinónimo de UPPER.

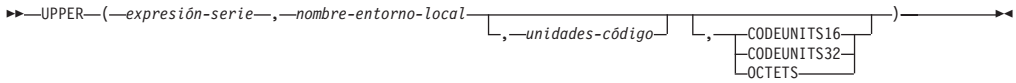
UPPER

El esquema es SYSIBM. (La versión SYSFUN de esta función sigue estando disponible para la compatibilidad con versiones superiores.)

La función UPPER es idéntica a la función TRANSLATE excepto en que sólo se especifica el primer argumento (*exp-serie-car*).

En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

UPPER (sensible al entorno local)



El esquema es SYSIBM.

La función UPPER devuelve una serie en la que todos los caracteres se han convertido a mayúsculas utilizando las normas asociadas con el entorno local especificado.

expresión-serie
Expresión que devuelve una serie CHAR, VARCHAR, GRAPHIC o VARGRAPHIC. Si *expresión-serie* es CHAR o VARCHAR, la expresión no debe ser FOR BIT DATA (SQLSTATE 42815).

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local que define las normas de conversión a caracteres en mayúsculas. El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery".

unidades-código
Constante entera que especifica el número de unidades de código en el resultado. Si se especifica, *unidades-código* debe ser un entero entre 1 y 32.672 si el resultado son datos de tipo carácter o entre 1 y 16.336 si el resultado son datos gráficos (SQLSTATE 42815). Si *unidades-código* no se especifica de forma explícita, es implícitamente el atributo de longitud de *expresión-serie*. Si se especifica OCTETS y el resultado son datos gráficos, el valor de *unidades-código* debe ser par (SQLSTATE 428GC).

CODEUNITS16, CODEUNITS32 u OCTETS
Especifica la unidad de serie de *unidades-código*.

CODEUNITS16 especifica que *unidades-código* se expresa en unidades de código UTF-16 de 16 bits. CODEUNITS32 especifica que *unidades-código* se expresa en unidades de código UTF-32 de 32 bits. OCTETS especifica que *unidades-código* se expresa en bytes.

Si la unidad de la serie no se especifica de forma explícita, el tipo de datos del resultado determina la unidad que se utiliza. Si el resultado son datos gráficos, *unidades-código* se expresa en unidades de dos bytes; de lo contrario, se expresa en bytes. Para obtener más información sobre CODEUNITS16, CODEUNITS32 y OCTETS, consulte el apartado "Unidades de serie en funciones incorporadas" en "Series de caracteres".

El resultado de la función es VARCHAR si *expresión-serie* es CHAR o VARCHAR y VARGRAPHIC si *expresión-serie* es GRAPHIC o VARGRAPHIC.

El atributo de longitud del resultado lo determina el valor implícito o explícito de *unidades-código*, la unidad de serie implícita o explícita y el tipo de datos de resultado, como se muestra en la tabla siguiente:

Tabla 60. Atributo de longitud del resultado de UPPER como función de unidad de serie y tipo de resultado

Unidad de serie	Tipo de resultado de caracteres	Tipo de resultado de gráfico
CODEUNITS16	MIN(*unidades-código* * 3, 32768)	*unidades-código*
CODEUNITS32	MIN(*unidades-código* * 4, 32768)	MIN(*unidades-código* * 2, 16336)
OCTETS	*unidades-código*	MIN(*unidades-código* / 2, 16336)

La longitud real del resultado puede ser mayor que la longitud de *expresión-serie*. Si la longitud real del resultado es mayor que el atributo de longitud del resultado, se devuelve un error (SQLSTATE 42815). Si el número de unidades de código del resultado excede el valor de *unidades-código*, se devuelve un error (SQLSTATE 42815).

Si *expresión-serie* no está en UTF-16, esta función realiza la conversión de página de códigos de *expresión-serie* a UTF-16 y del resultado de UTF-16 a la página de códigos de *expresión-serie*. Si cualquiera de las conversiones de página de códigos produce como mínimo un carácter de sustitución, el resultado incluye el carácter de sustitución, se devuelve un aviso (SQLSTATE 01517) y el distintivo de aviso SQLWARN8 de la SQLCA se establece en 'W'.

Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Ejemplos:

- Asegúrese de que los caracteres del valor de la columna JOB de la tabla EMPLOYEE se devuelvan en mayúsculas.

```
SELECT UPPER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

  El resultado es el valor 'MANAGER'.

- Busque las mayúsculas de todos los caracteres 'I' de una serie en idioma turco.

```
VALUES UPPER('IIii', 'tr_TR', CODEUNITS16)
```

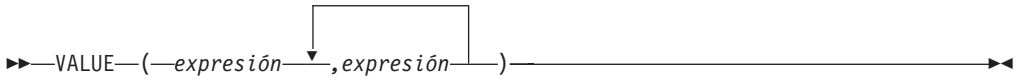
  El resultado es la serie 'IIII'.

- Busque la forma en mayúsculas de la letra 'ß' alemana (S sorda).

```
VALUES UPPER('ß', 'de', 2, CODEUNITS16)
```

  El resultado es la serie 'SS'. Tenga en cuenta que en este ejemplo se deben especificar *unidades-código* porque hay más unidades de código en el resultado que en *expresión-serie*.

VALUE



El esquema es SYSIBM.

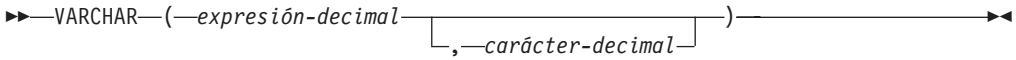
La función VALUE devuelve el primer argumento que no es nulo.

VALUE es sinónimo de COALESCE.

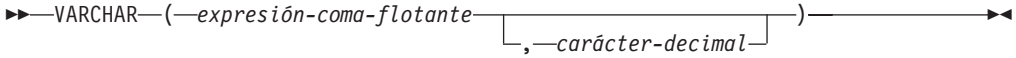
VARCHAR

De entero a varchar

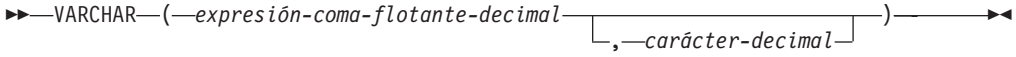
De decimal a varchar



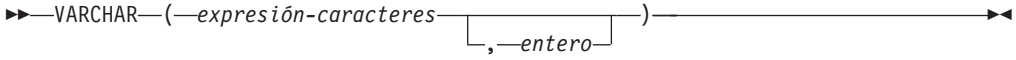
De coma flotante a varchar



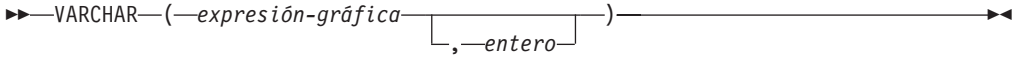
De coma flotante decimal a varchar



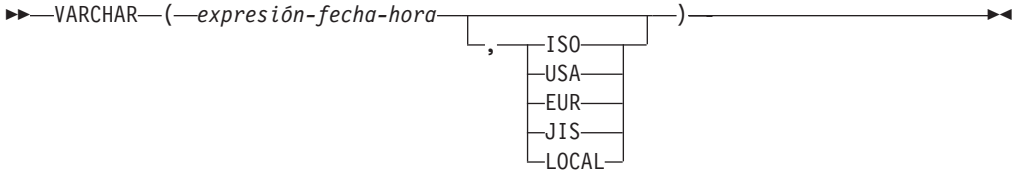
De carácter a varchar



De gráfico a varchar



De fecha y hora a varchar



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función.

La función VARCHAR devuelve una representación de serie de caracteres de longitud variable de:

- Un número entero, si el primer argumento es SMALLINT, INTEGER o BIGINT
- Un número decimal, si el primer argumento es un número decimal
- Un número de coma flotante de precisión doble, si el primer argumento es DOUBLE o REAL

- Un número de coma flotante decimal, si el primer argumento es un (DECFLOAT)
- Una serie de caracteres, si el primer argumento es cualquier tipo de serie de caracteres
- Una serie gráfica (sólo para bases de datos Unicode), si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora, si el primer argumento es DATE, TIME o TIMESTAMP

En una base de datos Unicode, cuando en la serie de salida un carácter de varios bytes aparece truncado en parte:

- Si la salida era una serie de caracteres, el carácter parcial se sustituye por uno o varios blancos
- Si la salida era una serie gráfica, el carácter parcial se sustituye por una serie vacía

No confíe en ninguno de estos comportamientos, porque podrían cambiar en los releases futuros.

El resultado de la función es una serie de caracteres de longitud variable. Si el primer argumento puede ser nulo, el resultado puede ser nulo. Si el primer argumento es nulo, el resultado es el valor nulo.

De entero a varchar

expresión-entero
Una expresión que devuelve un valor que es un tipo de datos de entero (SMALLINT, INTEGER o BIGINT).

El resultado es la representación de serie de longitud variable de la *expresión-entero* con el formato de una constante de entero SQL. El atributo de longitud del resultado depende de si *expresión-entero* es un entero pequeño, grande o superior, como se indica a continuación:

- Si el primer argumento es un entero pequeño, la longitud máxima del resultado es de 6.
- Si el primer argumento es un entero grande, la longitud máxima del resultado es de 11.
- Si el primer argumento es un entero superior, la longitud máxima del resultado es de 20.

La longitud real del resultado es el número menor de caracteres que se puede utilizar para representar el valor del argumento. Los ceros iniciales no se incluyen. Si el argumento es negativo, el primer carácter del resultado es un signo menos. Si no, el primer carácter es un dígito.

La página de códigos del resultado es la página de códigos de la sección.

De decimal a varchar

expresión-decimal
Una expresión que devuelve un valor que es de un tipo de datos decimal. La función escalar DECIMAL puede utilizarse para cambiar la precisión y la escala.

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el

signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud variable de la *expresión-decimal* con el formato de una constante decimal SQL. El atributo de longitud del resultado es 2+*p*, donde *p* es la precisión de la *expresión-decimal*. La longitud real del resultado es el número menor de caracteres que se puede utilizar para representar el resultado, salvo que los ceros finales se incluyen. Los ceros iniciales no se incluyen. Si expresión-decimal es negativa, el primer carácter de doble byte del resultado es un signo menos; en caso contrario, el primer carácter es un dígito del carácter decimal. Si la escala de *expresión-decimal* es cero, el carácter decimal no se devuelve.

La página de códigos del resultado es la página de códigos de la sección.

De coma flotante a varchar

expresión-coma-flotante
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud variable de la *expresión-coma-flotante* con el formato de una constante de coma flotante SQL.

La longitud máxima del resultado es 24. La longitud real del resultado es el número menor de caracteres que puede representar el valor de expresión-coma-flotante, de manera que la mantisa conste de un solo dígito que no sea cero seguido del *carácter-decimal* y una secuencia de dígitos. Si *expresión-coma-flotante* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si *expresión-coma-flotante* es cero, el resultado es 0E0.

La página de códigos del resultado es la página de códigos de la sección.

De coma flotante decimal a varchar

expresión-coma-flotante-decimal
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante decimal (DECFLOAT).

carácter-decimal
Especifica la constante de caracteres de un solo byte que se utiliza para delimitar los dígitos decimales en la serie de caracteres del resultado. La constante de caracteres no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie de caracteres de longitud variable de *expresión-coma-decimal-flotante* con el formato de una constante de coma flotante decimal SQL. La longitud máxima del resultado es 42. La longitud real del resultado es el número menor de caracteres que puede

representar el valor de *expresión-coma-flotante-decimal*. Si *expresión-coma-flotante-decimal* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si *expresión-coma-flotante-decimal* es cero, el resultado es 0.

Si el valor de *expresión-coma-flotante-decimal* es el valor especial Infinity, sNaN o NaN, se devuelven las series "INFINITY", "SNAN" y "NAN", respectivamente. Si el valor especial es negativo, el primer carácter del resultado es un signo menos. El valor especial de coma flotante decimal sNaN no genera un aviso cuando se convierte en una serie.

La página de códigos del resultado es la página de códigos de la sección.

De carácter a varchar

expresión-caracteres
Una expresión que devuelve un valor que es una serie de caracteres incorporados (CHAR, VARCHAR o CLOB).

entero
El atributo de longitud de la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672.

Si no se especifica el segundo argumento:

- Si la *expresión-caracteres* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento.

La longitud real del resultado es el atributo de longitud mínimo del resultado y la longitud real de *expresión-caracteres*. Si la longitud de la *expresión-caracteres* es superior al atributo de longitud del resultado, éste se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y la *expresión-caracteres* no sea CLOB.

De gráfico a varchar

expresión-gráfica
Una expresión que devuelve un valor que es un tipo de datos de serie gráfica incorporada (GRAPHIC, VARGRAPHIC o DBCLOB).

entero
El atributo de longitud de la serie de caracteres de longitud variable resultante. El valor debe estar entre 0 y 32 672.

Si el segundo argumento no se especifica:

- Si la *expresión-gráfica* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es igual al atributo de longitud del resultado multiplicado por 3 del primer argumento.

La longitud real del resultado es el atributo de longitud mínimo del resultado y la longitud real de *expresión-gráfica*. Si la longitud de la *expresión-gráfica* es superior al atributo de longitud del resultado, éste se trunca sin que se devuelva ningún aviso.

De fecha y hora a varchar

expresión-fecha-hora
Una expresión que sea uno de los tipos de datos siguientes:

DATE El resultado es la representación de serie de caracteres de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIME El resultado es la representación de serie de caracteres de la hora en el formato especificado por el segundo argumento. La longitud del resultado es 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIMESTAMP
El resultado es la representación de serie de caracteres de la indicación de fecha y hora. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(0), la longitud del resultado es 19. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(*n*), donde *n* es un número entre 1 y 12, la longitud del resultado es 20+*n*. De lo contrario, la longitud del resultado es 26. El segundo argumento no se debe especificar (SQLSTATE 42815).

La página de códigos del resultado es la página de códigos de la sección.

Notas:

- La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones cuando el primer argumento es numérico o el primer argumento es una serie y se especifica el argumento de longitud. Para obtener más información, consulte la “especificación CAST”.
- Se permite una serie binaria como primer argumento de la función y la serie de longitud variable resultado es una serie de caracteres FOR BIT DATA.

Ejemplos

- *Ejemplo 1:* crear una longitud variable EMPNO con una longitud de 10.

```
SELECT VARCHAR(EMPNO,10)
  INTO :VARHV
  FROM EMPLOYEE
```

- *Ejemplo 2:* establecer la variable del lenguaje principal JOB_DESC, definida como VARCHAR(8), en el equivalente VARCHAR de la descripción de trabajo (que es el valor de la columna JOB), definida como CHAR(8), para la empleada Dolores Quintana.

```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

- *Ejemplo 3:* la columna EDLEVEL se define como SMALLINT. Lo siguiente devuelve el valor como serie de caracteres de longitud variable.

```
SELECT VARCHAR(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

Genera el valor ’18’.

- *Ejemplo 4:* las columnas SALARY y COMM se definen como DECIMAL con una precisión de 9 y una escala de 2. Se devuelven los ingresos totales del empleado Haas con el carácter de coma decimal.

```
SELECT VARCHAR(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

Genera el valor ′56970,00′.

VARCHAR_BIT_FORMAT

El esquema es SYSIBM.

La función VARCHAR_BIT_FORMAT devuelve una representación de serie de bits de una serie de caracteres que se ha formateado utilizando una plantilla de caracteres. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

expresión-caracteres
Expresión que devuelve un valor que es una serie de caracteres incorporada que no es un CLOB (SQLSTATE 42815). La longitud necesaria la determinan la serie de formato especificada y el modo en que se interpreta el valor.

serie-formato
Una constante de tipo carácter que contiene una plantilla sobre cómo se deben interpretar los bytes de *expresión-caracteres*.

Las series de formato válidas incluyen: `'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'` y `'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'` (SQLSTATE 42815) donde cada 'x' o 'X' corresponde a un dígito hexadecimal en el resultado.

El resultado de la función es una serie de caracteres de longitud variable FOR BIT DATA con el atributo de longitud y la longitud real basada en la serie de formato. Para las dos series de formato válidas listadas más arriba, el atributo de longitud del resultado es 36 y la longitud real es 16. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Representar un identificador exclusivo universal en su formato binario:

```
VARCHAR_BIT_FORMAT('d83d6360-1818-11db-9804-b622a1ef5492',
                   'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  Resultado devuelto:

```
x'D83D6360181811DB9804B622A1EF5492'
```

- Representar un identificador exclusivo universal en su formato binario:

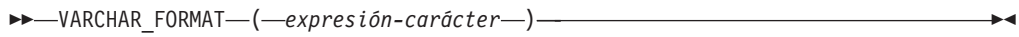
```
VARCHAR_BIT_FORMAT('D83D6360-1818-11DB-9804-B622A1EF5492',
                   'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

  Resultado devuelto:

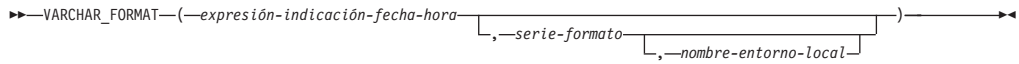
```
x'D83D6360181811DB9804B622A1EF5492'
```

VARCHAR_FORMAT

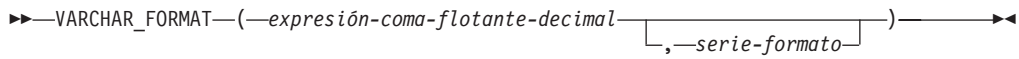
De carácter a varchar



De indicación de fecha y hora a varchar:



De coma flotante decimal a varchar:



El esquema es SYSIBM.

La función VARCHAR_FORMAT devuelve una serie de caracteres según la aplicación del argumento de serie de formato especificado, si se proporciona, al valor del primer argumento. Si cualquier argumento de la función VARCHAR_FORMAT puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

La expresión debe tener un formato que se ajuste a una plantilla de caracteres especificada.

De carácter a varchar

expresión-caracteres
Expresión que devuelve un valor que es un tipo de datos incorporado CHAR o VARCHAR. En una base de datos Unicode, si un argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función.

El resultado es un valor VARCHAR con un atributo de longitud que coincide con el atributo de longitud del argumento. El valor del resultado es el mismo que el valor de *expresión-carácter*.

De indicación de fecha y hora a varchar

expresión-indicación-fecha-hora
Expresión que devuelve un valor que debe ser de tipo DATE o TIMESTAMP o una representación de serie válida de una fecha o indicación de fecha y hora que no sea un CLOB ni un DBCLOB. Si el argumento es una serie, el argumento de *serie-formato* debe especificarse. En una base de datos Unicode, si un argumento proporcionado es una representación de serie gráfica de un dato, hora o indicación de fecha y hora, se convertirá a una serie de caracteres antes de que se evalúe la función.

Si *expresión-indicación-fecha-hora* es un valor DATE o una representación de serie válida de una fecha, ésta primero se convierte en un valor TIMESTAMP(0), dándose por supuesta la hora exacta de la medianoche (00.00.00).

Para conocer los formatos válidos de las representaciones de serie de los valores de fecha y hora, consulte "Representación mediante series de los valores de fecha y hora" en "Valores de fecha y hora".

serie-formato
La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR, VARCHAR, numérico o de fecha y hora. Si el valor no es un tipo de datos CHAR o VARCHAR, se convierte implícitamente a VARCHAR antes de evaluar la función. En una base de datos Unicode, si el argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función. La longitud real no debe superar los 254 bytes (SQLSTATE 22007). El valor es una plantilla del formato que debe tener *expresión-fecha-hora*.

Una *serie-formato* válida debe contener una combinación de los elementos de formato listados más abajo (SQLSTATE 22007). Dos elementos de formato se pueden separar opcionalmente por uno o varios de los siguientes caracteres separadores:

- signo menos (-)
- punto (.)
- barra inclinada (/)
- coma (,)
- apóstrofo (')
- punto y coma (;)
- dos puntos (:)
- espacio en blanco ( )

También se pueden especificar caracteres separadores al principio o al final de *serie-formato*.

Tabla 61. Elementos de formato para la función VARCHAR_FORMAT

Elemento de formato	Descripción
AM o PM	Indicador de meridiano (mañana o tarde) sin puntos. Este elemento de formato depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
A.M. o P.M.	Indicador de meridiano (mañana o tarde) con puntos. Este elemento de formato utiliza las series exactas 'A.M.' o 'P.M.' y depende del nombre de entorno local en vigor.
CC	Siglo (01-99). Si los dos últimos dígitos del año en formato de cuatro dígitos son cero, el resultado es los dos primeros dígitos del año; si no, el resultado es los dos primeros dígitos del año más uno.
DAY, Day o day	Nombre del día todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.

Tabla 61. Elementos de formato para la función VARCHAR_FORMAT (continuación)

Elemento de formato	Descripción
DY, Dy o dy	Nombre abreviado del día todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
D	Día de la semana (1-7). El primer día de la semana depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
DD	Día del mes (01-31).
DDD	Día del año (001-366).
FF o FF*n*	Segundos fraccionarios (0-999999999999). El número *n* se utiliza para especificar el número de dígitos esperados en la *expresión-serie*. Los valores válidos para *n* son del 1 al 12 sin ceros iniciales. Especificar FF equivale a especificar FF6. Si la precisión de indicación de fecha y hora de *expresión-indicación-fecha-hora* es inferior a lo que especifica el formato, no se rellena ningún dígito a la derecha de los dígitos especificados.
HH	HH se comporta igual que HH12.
HH12	Hora del día (01-12) en formato de 12 horas.
HH24	Hora del día (00-24) en formato de 24 horas.
IW	Semana ISO del año (01-53). La semana empieza el lunes e incluye siete días. La semana 1 es la primera semana del año que contiene un jueves, lo que es equivalente a la primera semana del año que contiene el 4 de enero.
I	Año ISO (0-9). Último dígito del año según la semana ISO que se devuelva.
IY	Año ISO (00-99). Últimos dos dígitos del año según la semana ISO que se devuelva.
IYY	Año ISO (000-999). Últimos tres dígitos del año según la semana ISO que se devuelva.
IYYY	Año ISO (0000-9999). Año de cuatro dígitos según la semana ISO que se devuelva.
J	Día del calendario juliano (número de días desde el 1 de enero, 4713 AC).
MI	Minuto (00-59).
MM	Mes (01-12).
NNNNNN	Microsegundos (000000-999999). Igual que FF6.

Tabla 61. Elementos de formato para la función VARCHAR_FORMAT (continuación)

Elemento de formato	Descripción
MONTH, Month o month	Nombre del mes todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
MON, Mon o mon	Nombre abreviado del mes todo en mayúsculas, con mayúscula inicial o en minúsculas. El idioma utilizado depende del *nombre-entorno-local*, si se especifica; si no, depende del valor del registro especial CURRENT LOCALE LC_TIME.
Q	Trimestre (1-4), donde los meses de enero a marzo devuelven 1.
RR	RR se comporta igual que YY.
RRRR	RRRR se comporta igual que YYYY.
SS	Segundos (00-59).
SSSSS	Segundos desde la medianoche anterior (00000-86400).
W	Semana del mes (1-5), donde la semana 1 empieza el primer día del mes y acaba el séptimo día del mes.
WW	Semana del año (01-53), donde la semana 1 empieza el 1 de enero y acaba el 7 de enero.
S	Último dígito del año (0-9).
YY	Dos últimos dígitos del año (00-99).
YYY	Tres últimos dígitos del año (000-999).
YYYY	Año de 4 dígitos (0000-9999).

Nota: Los elementos de formato de la Tabla 61 en la página 644 no son sensibles a mayúsculas y minúsculas, salvo los siguientes:

- AM, PM
- A.M., P.M.
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

En los casos en los que los elementos de formato son ambiguos, los elementos de formato que no son sensibles a las mayúsculas y minúsculas se tomarán en consideración en primer lugar. Por ejemplo, DDYYYY se interpretaría como DD seguido de YYYY, en lugar de D seguido de DY, seguido de YYY.

Si no se especifica *serie-formato*, el formato que se utiliza se basa en el valor del registro especial CURRENT LOCALE LC_TIME.

nombre-entorno-local
Constante de tipo carácter que especifica el entorno local utilizado para los elementos de formato siguientes:

- AM, PM
- DAY, Day, day
- DY, Dy, dy
- D
- MONTH, Month, month
- MON, Mon, mon

El valor de *nombre-entorno-local* no es sensible a las mayúsculas y minúsculas y debe ser un entorno local válido (SQLSTATE 42815). Para obtener información sobre los entornos locales válidos y su significado, consulte "Nombres de entorno local para SQL y XQuery" en la publicación *Globalization Guide* . Si no se especifica *nombre-entorno-local*, se utiliza el valor del registro especial CURRENT LOCALE LC_TIME.

El resultado es una representación de *expresión-indicación-fecha-hora* en el formato especificado por *serie-formato*. La *serie-formato* se interpreta como una serie de elementos de formato que opcionalmente se pueden separar mediante uno o más caracteres separadores. Una serie de caracteres en *serie-formato* se interpreta como el elemento de formato coincidente más largo de Tabla 61 en la página 644. Si dos elementos de formato que contienen los mismos caracteres no están delimitados por un carácter separador, la especificación se interpreta, empezando desde la izquierda, como el elemento de formato coincidente más largo de la tabla y continúa hasta que se encuentran coincidencias para el resto de la serie de formato. Por ejemplo, 'YYYYYYDD' se interpreta como los elementos de formato 'YYYY', 'YY' y 'DD'.

El resultado es una serie de caracteres de longitud variable. El atributo de longitud es 254. *serie-formato* determina la longitud real del resultado. Si la serie resultante supera el atributo de longitud del resultado, el resultado se trunca.

De coma flotante decimal a varchar

expresión-coma-flotante-decimal
Una expresión que devuelve un valor de cualquier tipo de datos numérico interno. Si el argumento no es un valor de coma flotante decimal, se convertirá a DECFLOAT(34) para su proceso.

serie-formato
La expresión debe devolver un valor que sea un tipo de datos incorporado CHAR, VARCHAR, numérico o de fecha y hora. Si el valor no es un tipo de datos CHAR o VARCHAR, se convierte implícitamente a VARCHAR antes de evaluar la función. En una base de datos Unicode, si el argumento proporcionado es un tipo de datos GRAPHIC o VARGRAPHIC, se convertirá a VARCHAR antes de que se evalúe la función. La longitud real no debe superar los 254 bytes (SQLSTATE 22018). El valor es una plantilla del formato que debe tener *expresión-punto-flotante-decimal*. Los elementos de formato especificados como prefijo sólo pueden utilizarse al principio de la plantilla. Los elementos de formato especificados como sufijo sólo pueden utilizarse al final de la plantilla. Los elementos de formato son sensibles a mayúsculas y minúsculas. La plantilla no debe contener más de uno de los elementos de formato MI, S o PR (SQLSTATE 22018).

Tabla 62. Elementos de formato para la función VARCHAR_FORMAT

Elemento de formato	Descripción
0	Cada 0 representa un dígito significativo. Los ceros iniciales se visualizan como ceros.
9	Cada 9 representa un dígito significativo. Los ceros iniciales se visualizan como espacios en blanco.
MI	Sufijo: si *expresión-coma-flotante-decimal* es un número negativo, se incluye un signo menos final (-) en el resultado. Si *expresión-coma-flotante-decimal* es un número positivo, se incluye un espacio en blanco final en el resultado.
S	Prefijo: si *expresión-coma-flotante-decimal* es un número negativo, se incluye un signo menos inicial (-) en el resultado. Si *expresión-coma-flotante-decimal* es un número positivo, se incluye un signo más inicial (+) en el resultado.
PR	Sufijo: si *expresión-coma-flotante-decimal* es un número negativo, se incluye un carácter de menor que inicial (<) y un carácter de mayor que final (>) en el resultado. Si *expresión-coma-flotante-decimal* es un número positivo, se incluye un espacio inicial y un espacio final en el resultado.
$	Prefijo: se incluye un signo de dólar inicial ($) en el resultado.
,	Especifica que se incluirá una coma en esa ubicación del resultado. Esta coma se utiliza como separador de grupos.
.	Especifica que se incluirá un punto en esa ubicación del resultado. Este punto se utiliza como separador decimal.

Si no se especifica serie-formato, a *expresión-coma-flotante-decimal* se le asigna un formato de constante con coma flotante decimal SQL. Si *expresión-coma-flotante-decimal* es negativa, el primer carácter del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si *expresión-coma-flotante-decimal* es cero, el resultado es 0.

El resultado es una representación de serie de caracteres de longitud variable de *expresión-coma-flotante-decimal*. El atributo de longitud es 254. La longitud real del resultado se determina mediante *serie-formato*, si se especifica; en caso contrario, la longitud real del resultado es el número menor de caracteres que pueden representar el valor de *expresión-coma-flotante-decimal*. Si la serie resultante supera el atributo de longitud del resultado, el resultado se truncará.

Si el valor de *expresión-coma-flotante-decimal* es el valor especial Infinity, sNaN o NaN, se devuelven las series ″INFINITY″, ″SNAN″ y ″NAN″, respectivamente. Si el valor especial es negativo, el primer carácter del resultado es un signo menos. El valor especial de coma flotante decimal sNaN no genera una excepción cuando se convierte en una serie.

Si *serie-formato* no incluye ninguno de los elementos de formato MI, S o PR y el valor de *expresión-coma-flotante-decimal* es negativo, se incluirá un signo menos (-) en el resultado; si no, se incluirá un espacio en blanco en el resultado.

Si el número de dígitos a la izquierda de la coma decimal de *expresión-coma-flotante-decimal* es mayor que el número de dígitos a la izquierda de la coma decimal de *serie-formato*, el resultado es una serie de caracteres de signo de número (#). Si el número de dígitos a la derecha de la coma decimal de *expresión-coma-flotante-decimal* es mayor que el número de dígitos a la derecha de la coma decimal de *serie-formato*, el resultado es una *expresión-coma-flotante-decimal* redondeada al número de dígitos a la derecha de la coma decimal de *serie-formato*. La modalidad

de redondeo DECFLOAT no se utilizará. El comportamiento de redondeo de VARCHAR_FORMAT corresponde a un valor de ROUND_HALF_UP.

La página de códigos del resultado es la página de códigos de la sección.

Notas:

- ***Calendario juliano y gregoriano***: en el caso de conversión de indicación de fecha y hora a varchar, esta función tiene en cuenta la transición del calendario juliano al calendario gregoriano el 15 de octubre de 1582.
- ***Determinismo:*** VARCHAR_FORMAT es una función determinista. Sin embargo, las invocaciones siguientes de la función dependen del valor del registro especial CURRENT LOCALE LC_TIME.
  - En el caso de conversión de indicación de fecha y hora a varchar, cuando no se especifica de forma explícita serie-formato o cuando nombre-entorno-local no se especifica de forma explícita y se da una de las condiciones siguientes:
    - *serie-formato* no es una constante
    - *serie-formato* es una constante e incluye elementos de formato que son sensibles al entorno local

    Estas invocaciones que dependen del valor de un registro especial no se pueden utilizar donde no se puedan utilizar registros especiales (SQLSTATE 42621 o 428EC).
- ***Alternativas de la sintaxis:*** TO_CHAR es sinónimo de VARCHAR_FORMAT.

Ejemplos

- Visualizar los nombres de las tablas y las indicaciones de fecha y hora de creación de todas las tablas del sistema cuyo nombre empiece por 'SYSU'.

```
SELECT VARCHAR(TABNAME, 20) AS TABLE_NAME,
  VARCHAR_FORMAT(CREATE_TIME, 'YYYY-MM-DD HH24:MI:SS')
    AS CREATION_TIME
  FROM SYSCAT.TABLES
  WHERE TABNAME LIKE 'SYSU%'
```

  Este ejemplo devuelve lo siguiente:

```
TABLE_NAME           CREATION_TIME
-------------------- -------------------
SYSUSERAUTH          2000-05-19 08:18:56
SYSUSEROPTIONS       2000-05-19 08:18:56
```

- Suponer que la variable TMSTAMP se ha definido como TIMESTAMP y tiene el valor siguiente: 2007-03-09-14.07.38.123456. Los ejemplos siguientes muestran varias invocaciones de la función y los valores de serie resultantes. El tipo de datos de resultado en cada caso es VARCHAR (254).

```
Invocación de función                        Resultado
-------------------                          ------
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMISSFF3')  20070309020738123
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHH24MISS')   20070309140738
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMI')       200703090207
VARCHAR_FORMAT(TMSTAMP,'DD/MM/YY')           09/03/07
VARCHAR_FORMAT(TMSTAMP,'MM-DD-YYYY')         03-09-2007
VARCHAR_FORMAT(TMSTAMP,'J')                  2454169
VARCHAR_FORMAT(TMSTAMP,'Q')                  1
VARCHAR_FORMAT(TMSTAMP,'W')                  2
VARCHAR_FORMAT(TMSTAMP,'IW')                 10
VARCHAR_FORMAT(TMSTAMP,'WW')                 10
VARCHAR_FORMAT(TMSTAMP,'Month','en_US')      March
VARCHAR_FORMAT(TMSTAMP,'MONTH','en_US')      MARCH
VARCHAR_FORMAT(TMSTAMP,'MON','en_US')        MAR
VARCHAR_FORMAT(TMSTAMP,'Day','en_US')        Friday
```

```
VARCHAR_FORMAT(TMSTAMP,'DAY','en_US')        FRIDAY
VARCHAR_FORMAT(TMSTAMP,'Dy','en_US')         Fri
VARCHAR_FORMAT(TMSTAMP,'Month','de_DE')      März
VARCHAR_FORMAT(TMSTAMP,'MONTH','de_DE')      MÄRZ
VARCHAR_FORMAT(TMSTAMP,'MON','de_DE')        MRZ
VARCHAR_FORMAT(TMSTAMP,'Day','de_DE')        Freitag
VARCHAR_FORMAT(TMSTAMP,'DAY','de_DE')        FREITAG
VARCHAR_FORMAT(TMSTAMP,'Dy','de_DE')         Fr
```

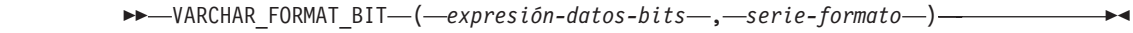
- Suponer que la variable DTE se ha definido como DATE y tiene el valor siguiente: 2007-03-09. Los ejemplos siguientes muestran varias invocaciones de la función y los valores de serie resultantes. El tipo de datos de resultado en cada caso es VARCHAR (254).

```
Invocación de función                        Resultado
-------------------                          ------
VARCHAR_FORMAT(DTE,'YYYYMMDD')               20070309
VARCHAR_FORMAT(DTE,'YYYYMMDDHH24MISS')       20070309000000
```

- Suponer que las variables POSNUM y NEGNUM se han definido como DECFLOAT(34) y tienen los valores siguientes: 1234.56 y -1234.56, respectivamente. Los ejemplos siguientes muestran varias invocaciones de la función y los valores de serie resultantes. El tipo de datos de resultado en cada caso es VARCHAR (254).

```
Invocación de función                        Resultado
-------------------                          ------
VARCHAR_FORMAT(POSNUM)                       '1234.56'
VARCHAR_FORMAT(NEGNUM)                       '-1234.56'
VARCHAR_FORMAT(POSNUM,'9999.99')             '1234.56'
VARCHAR_FORMAT(NEGNUM,'9999.99')             '1234.56'
VARCHAR_FORMAT(POSNUM,'99999.99')            ' 1234.56'
VARCHAR_FORMAT(NEGNUM,'99999.99')            ' 1234.56'
VARCHAR_FORMAT(POSNUM,'00000.00')            '01234.56'
VARCHAR_FORMAT(NEGNUM,'00000.00')            '01234.56'
VARCHAR_FORMAT(POSNUM,'9999.99MI')           '1234.56 '
VARCHAR_FORMAT(NEGNUM,'9999.99MI')           '1234.56-'
VARCHAR_FORMAT(POSNUM,'S9999.99')            '+1234.56'
VARCHAR_FORMAT(NEGNUM,'S9999.99')            '-1234.56'
VARCHAR_FORMAT(POSNUM,'9999.99PR')           ' 1234.56 '
VARCHAR_FORMAT(NEGNUM,'9999.99PR')           '<1234.56>'
VARCHAR_FORMAT(POSNUM,'S$9,999.99')          '+$1,234.56'
VARCHAR_FORMAT(NEGNUM,'S$9,999.99')          '-$1,234.56'
```

VARCHAR_FORMAT_BIT



El esquema es SYSIBM.

La función VARCHAR_FORMAT_BIT devuelve una representación de caracteres de una serie de bits a la que se ha dado formato utilizando una plantilla de caracteres.

expresión-datos-bits
Expresión que devuelve un valor que es un tipo de datos FOR BIT DATA de serie de caracteres incorporada (SQLSTATE 42815). La longitud necesaria la determinan la serie de formato especificada y el modo en que se interpreta el valor.

serie-formato
Una constante de caracteres que contiene una plantilla para el formato que deben darse al resultado.

Las series de formato válidas incluyen: `'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'` y `'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'` (SQLSTATE 42815) donde cada 'x' o 'X' corresponde a un dígito hexadecimal de *expresión-datos-bit*.

El resultado de la función es una serie de caracteres de longitud variable con el atributo de longitud y la longitud real basada en la serie de formato. Para las dos series de formato válidas listadas más arriba, el atributo de longitud es 36 y la longitud real es 36 bytes. Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

Ejemplos

- Representar un identificador exclusivo universal en su forma con formato:

```
VARCHAR_FORMAT_BIT(CAST(x'd83d6360181811db9804b622a1ef5492'
                        AS VARCHAR(16) FOR BIT DATA),
                   'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  Resultado devuelto:

```
'd83d6360-1818-11db-9804-b622a1ef5492'
```

- Representar un identificador exclusivo universal en su forma con formato:

```
VARCHAR_FORMAT_BIT(CAST(x'd83d6360181811db9804b622a1ef5492'
                        AS CHAR(16) FOR BIT DATA),
                   'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

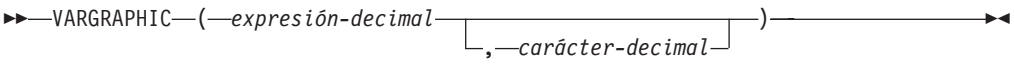
  Resultado devuelto:

```
'D83D6360-1818-11DB-9804-B622A1EF5492'
```

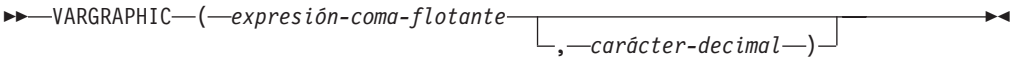
VARGRAPHIC

De entero a vargraphic:

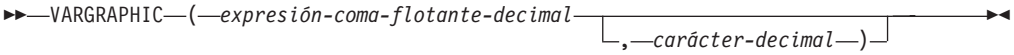
De decimal a vargraphic:



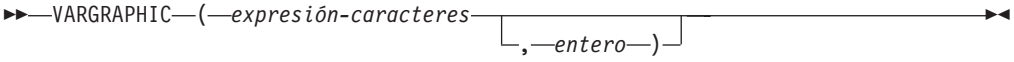
De coma flotante a vargraphic:



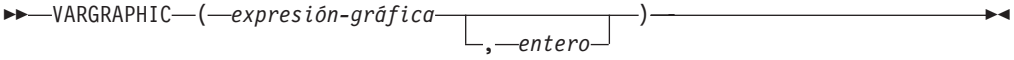
De coma flotante decimal a vargraphic:



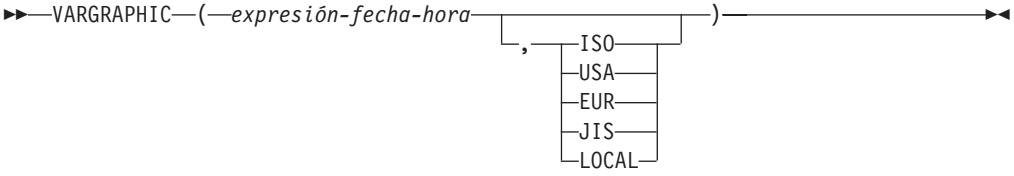
De carácter a vargraphic:



De gráfico a vargraphic:



De fecha y hora a vargraphic:



El esquema es SYSIBM. El nombre de la función no puede especificarse como nombre calificado si se utilizan palabras clave en la signatura de la función.

La función VARGRAPHIC devuelve una representación de serie gráfica de longitud variable de:

- Un número entero (sólo base de datos Unicode), si el primer argumento es SMALLINT, INTEGER o BIGINT
- Un número decimal (sólo base de datos Unicode), si el primer argumento es un número decimal

- Un número de coma flotante de precisión doble (sólo bases de datos Unicode), si el primer argumento es DOUBLE o REAL
- Un número de coma flotante decimal (sólo base de datos Unicode), si el primer argumento es un número de coma flotante decimal (DECFLOAT)
- Una serie de caracteres, convirtiendo los caracteres de un solo byte en caracteres de doble byte, si el primer argumento es cualquier tipo de serie de caracteres
- Una serie gráfica, si el primer argumento es cualquier tipo de serie gráfica
- Un valor de fecha y hora (sólo para bases de datos Unicode), si el primer argumento es un valor DATE, TIME o TIMESTAMP

En una base de datos Unicode, si un argumento proporcionado es una serie de caracteres, se convertirá a una serie gráfica antes de que se ejecute la función. Cuando la serie de salida se trunca, de forma que el último carácter es un carácter de sustitución elevado, dicho carácter se convierte en un carácter en blanco (X'0020'). No confíe en este comportamiento, porque podría cambiar en los releases futuros.

El resultado de la función es una serie gráfica de longitud variable (tipo de datos VARGRAPHIC). Si el primer argumento puede ser nulo, el resultado puede ser nulo; si el primer argumento es nulo, el resultado es el valor nulo.

De entero a vargraphic

expresión-entero
Una expresión que devuelve un valor que es un tipo de datos de entero (SMALLINT, INTEGER o BIGINT).

El resultado es la representación de serie gráfica de longitud variable de la *expresión-entero* con el formato de una constante de entero SQL. El atributo de longitud del resultado depende de si *expresión-entero* es un entero pequeño, grande o superior, como se indica a continuación:

- Si el primer argumento es un entero pequeño, la longitud máxima del resultado es de 6.
- Si el primer argumento es un entero grande, la longitud máxima del resultado es de 11.
- Si el primer argumento es un entero superior, la longitud máxima del resultado es de 20.

La longitud real del resultado es el número menor de caracteres de doble byte que se puede utilizar para representar el valor del argumento. Los ceros iniciales no se incluyen. Si el argumento es negativo, el primer carácter de doble byte del resultado es un signo menos; si no, el primer carácter de doble byte es un dígito.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De decimal a vargraphic

expresión-decimal
Una expresión que devuelve un valor que es de un tipo de datos decimal. La función escalar DECIMAL puede utilizarse para cambiar la precisión y la escala.

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo

más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud variable de la *expresión-decimal* con el formato de una constante decimal SQL. El atributo de longitud del resultado es 2+*p*, donde *p* es la precisión de la *expresión-decimal*. La longitud real del resultado es el número menor de caracteres de doble byte que se puede utilizar para representar el resultado, salvo que los ceros finales se incluyen. Los ceros iniciales no se incluyen. Si *expresión-decimal* es negativa, el primer carácter de doble byte del resultado es un signo menos; en caso contrario, el primer carácter es un dígito. Si la escala de *expresión-decimal* es cero, el carácter decimal no se devuelve.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De coma flotante a vargraphic

expresión-coma-flotante
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante (DOUBLE o REAL).

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud variable de la *expresión-coma-flotante* con el formato de una constante de coma flotante SQL.

La longitud máxima del resultado es 24. La longitud real del resultado es el número menor de caracteres de doble byte que puede representar el valor de *expresión-coma-flotante*, de manera que la mantisa conste de un solo dígito que no sea cero seguido del *carácter-decimal* y una secuencia de dígitos. Si *expresión-coma-flotante* es negativa, el primer carácter de doble byte del resultado es un signo menos; si no, el primer carácter de doble byte es un dígito. Si *expresión-coma-flotante* es cero, el resultado es 0E0.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De coma flotante decimal a vargraphic

expresión-coma-flotante-decimal
Una expresión que devuelve un valor que es de un tipo de datos de coma flotante decimal (DECFLOAT).

carácter-decimal
Especifica la constante de caracteres de doble byte que se utiliza para delimitar los dígitos decimales en la serie gráfica del resultado. La constante de caracteres de doble byte no puede ser un dígito, el signo más (+), el signo menos (-) ni un espacio en blanco (SQLSTATE 42815). El valor por omisión es el carácter de punto ('.').

El resultado es una representación de serie gráfica de longitud variable de la *expresión-coma-flotante-decimal* con el formato de una constante de coma

flotante decimal SQL. La longitud máxima del resultado es 42. La longitud real del resultado es el número menor de caracteres de doble byte que puede representar el valor de *expresión-coma-flotante-decimal*. Si *expresión-coma-flotante-decimal* es negativa, el primer carácter de doble byte del resultado es un signo menos; si no, el primer carácter de doble byte es un dígito. Si *expresión-coma-flotante-decimal* es cero, el resultado es 0.

Si el valor de *expresión-coma-flotante-decimal* es el valor especial Infinity, sNaN o NaN, se devuelven las series G'INFINITY', G'SNAN' y G'NAN', respectivamente. Si el valor especial es negativo, el primer carácter de doble byte del resultado es un signo menos. El valor especial de coma flotante decimal sNaN no genera un aviso cuando se convierte en una serie.

La página de códigos del resultado es la página de códigos DBCS de la sección.

De carácter a vargraphic

expresión-caracteres
Una expresión que devuelve un valor que es una serie de caracteres incorporados (CHAR, VARCHAR o CLOB).

entero
El atributo de longitud de la serie gráfica de longitud variable resultante. El valor debe estar entre 0 y 16.336. Si el segundo argumento no se especifica:

- Si la *expresión-caracteres* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento.

La longitud real del resultado es el atributo de longitud mínimo del resultado y la longitud real de *expresión-caracteres*. Si la longitud de la *expresión-caracteres* es superior al atributo de longitud del resultado, éste se trunca sin que se devuelva ningún aviso.

En el resultado, todos los caracteres de un solo byte de la *expresión-caracteres* se convierten en su representación de doble byte equivalente o en el carácter de sustitución de doble byte. Todos los caracteres de doble byte de la *expresión-caracteres* se correlacionan sin ninguna conversión adicional. Si el primer byte de un carácter de doble byte aparece como el último byte de la *expresión-caracteres*, éste se convierte en el carácter de sustitución de doble byte. El orden secuencial de los caracteres de la *expresión-caracteres* se conserva.

En el caso de una base de datos Unicode, esta función convierte la serie de caracteres de la página de códigos del argumento a UCS-2. Cada carácter del argumento, incluidos los caracteres de doble byte, se convierte. Si se proporciona un valor para el segundo argumento, éste especifica la longitud necesaria de la serie resultante (en caracteres UCS-2).

La conversión a elementos de código mediante la función VARGRAPHIC se basa en la página de códigos del argumento.

Los caracteres de doble byte del argumento no se convierten. El resto de caracteres se convierten a sus equivalentes de doble byte correspondientes. Si no existe un equivalente de doble byte correspondiente, se utiliza el carácter de sustitución de doble byte para la página de códigos.

No se genera ningún aviso ni código de error si se devuelven uno o varios caracteres de sustitución de doble byte en el resultado.

De gráfico a vargraphic

expresión-gráfica
Una expresión que devuelve un valor que es un tipo de datos de serie gráfica incorporada (GRAPHIC, VARGRAPHIC o DBCLOB).

entero
El atributo de longitud de la serie gráfica de longitud variable resultante. El valor debe estar entre 0 y 16.336. Si el segundo argumento no se especifica:

- Si la *expresión-gráfica* es la constante de serie vacía, el atributo de longitud del resultado es 0.
- Si no, el atributo de longitud del resultado es el mismo que el atributo de longitud del primer argumento.

La longitud real del resultado es el atributo de longitud mínimo del resultado y la longitud real de *expresión-gráfica*. Si la longitud de la *expresión-gráfica* es superior al atributo de longitud del resultado, éste se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y la *expresión-gráfica* no sea DBCLOB.

Si la longitud de la expresión gráfica es mayor que el atributo de longitud del resultado, el resultado se trunca. Se devuelve un aviso (SQLSTATE 01004), a menos que los caracteres truncados sean todo espacios en blanco y que la expresión gráfica no sea DBCLOB.

De fecha y hora a vargraphic

expresión-fecha-hora
Una expresión que sea de uno de los tipos de datos siguientes:

DATE El resultado es la representación de serie gráfica de la fecha en el formato especificado por el segundo argumento. La longitud del resultado es 10. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIME El resultado es la representación de serie gráfica de la hora en el formato especificado por el segundo argumento. La longitud del resultado es 8. Se devuelve un error si se especifica el segundo argumento y no es un valor válido (SQLSTATE 42703).

TIMESTAMP
El resultado es la representación de serie gráfica de la indicación de fecha y hora. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(0), la longitud del resultado es 19. Si el tipo de datos de *expresión-fecha-hora* es TIMESTAMP(*n*), donde *n* es un número entre 1 y 12, la longitud del resultado es 20+*n*. De lo contrario, la longitud del resultado es 26. El segundo argumento no se debe especificar (SQLSTATE 42815).

La página de códigos del resultado es la página de códigos DBCS de la sección.

Nota: La especificación CAST debe utilizarse para aumentar la portabilidad de las aplicaciones cuando el primer argumento es numérico o si el primer argumento es una serie y se especifica el argumento de longitud. Para obtener más información, consulte la "especificación CAST".

Ejemplos

- La columna EDLEVEL está definida como SMALLINT. Lo siguiente devuelve el valor como serie gráfica de longitud variable.

```
SELECT VARGRAPHIC(EDLEVEL)
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  Genera el valor G'18 '.

- Las columnas SALARY y COMM se definen como DECIMAL con una precisión de 9 y una escala de 2. Se devuelven los ingresos totales del empleado Haas con el carácter de coma decimal.

```
SELECT VARGRAPHIC(SALARY + COMM, ',')
  FROM EMPLOYEE
  WHERE LASTNAME = 'HAAS'
```

  Genera el valor G'56970,00 '.

WEEK

El esquema es SYSFUN.

La función escalar WEEK devuelve la semana del año del argumento como un valor entero comprendido entre 1 y 54. La semana empieza en domingo.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

WEEK_ISO

El esquema es SYSFUN.

La función WEEK devuelve la semana del año del argumento como un valor entero comprendido entre 1 y 53. La semana empieza en lunes e incluye siempre 7 días. La semana 1 es la primera semana del año que contenga un jueves, que equivale a la primera semana que contenga el 4 de enero. Por consiguiente, es posible hacer que aparezca un máximo de 3 días del principio de un año en la última semana del año anterior. Y, a la inversa, pueden aparecer un máximo de 3 días del final de un año en la primera semana del año siguiente.

El argumento debe ser un valor DATE, TIMESTAMP o una representación de serie de caracteres válida de una fecha o una indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

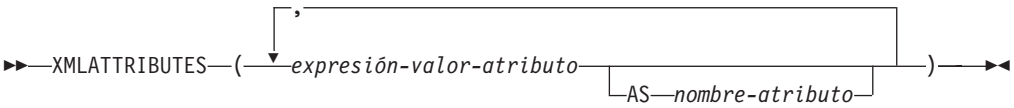
El resultado de la función es INTEGER. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo

La lista siguiente muestra ejemplos del resultado de WEEK_ISO y DAYOFWEEK_ISO.

```
DATE       WEEK_ISO    DAYOFWEEK_ISO
---------- ----------- -------------
1997-12-28          52             7
1997-12-31           1             3
1998-01-01           1             4
1999-01-01          53             5
1999-01-04           1             1
1999-12-31          52             5
2000-01-01          52             6
2000-01-03           1             1
```

XMLATTRIBUTES



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLATTRIBUTES construye los atributos XML a partir de los argumentos. Esta función sólo se puede utilizar como argumento de la función XMLELEMENT. El resultado es una secuencia XML que contiene un nodo de atributo XQuery para cada valor de entrada que no sea nulo.

expresión-valor-atributo
Expresión cuyo resultado es el valor de atributo. El tipo de datos de *expresión-valor-atributo* no puede ser un tipo estructurado (SQLSTATE 42884). La expresión puede ser cualquier expresión SQL. Si la expresión no es una referencia de columna simple, debe especificarse un nombre de atributo.

nombre-atributo
Especifica un nombre de atributo. El nombre es un identificador de SQL cuyo formato debe corresponder al de un nombre calificado XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. El nombre del atributo no puede ser `xmlns` ni llevar el prefijo `xmlns:`. Para declarar un espacio de nombres, debe utilizarse la función XMLNAMESPACES. No se permiten los nombres de atributos duplicados, tanto si son implícitos como explícitos (SQLSTATE 42713).

Si no se especifica *nombre-atributo*, *expresión-valor-atributo* debe ser un nombre de columna (SQLSTATE 42703). El nombre de atributo se crea a partir del nombre de columna, utilizando la correlación con elusión de caracteres ("fully escaped") desde un nombre de columna a un nombre de atributo XML.

El tipo de datos del resultado es XML. Si el resultado de la *expresión-valor-atributo* puede ser nulo, el resultado puede ser nulo; si el resultado de cada *expresión-valor-atributo* es nulo, el resultado es el valor nulo.

Ejemplos:

Nota: XMLATTRIBUTES no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Generar un elemento con atributos.

```
SELECT E.EMPNO, XMLELEMENT(
  NAME "Emp",
  XMLATTRIBUTES(
    E.EMPNO, E.FIRSTNME ||' '|| E.LASTNAME AS "name"
  )
)
AS "Result"
FROM EMPLOYEE E WHERE E.EDLEVEL = 12
```

Esta consulta genera el resultado siguiente:

```
EMPNO      Result
000290 <Emp EMPNO="000290" name="JOHN PARKER"></Emp>
000310 <Emp EMPNO="000310" name="MAUDE SETRIGHT"></Emp>
200310 <Emp EMPNO="200310" name="MICHELLE SPRINGER"></Emp>
```

- Generar un elemento con una declaración de espacio de nombres que no se utiliza en ningún QName. El prefijo se utiliza en un valor de atributo.

```
VALUES XMLELEMENT(
  NAME "size",
  XMLNAMESPACES(
    'http://www.w3.org/2001/XMLSchema-instance' AS "xsi",
    'http://www.w3.org/2001/XMLSchema' AS "xsd"
  ),
  XMLATTRIBUTES(
    'xsd:string' AS "xsi:type"
  ), '1'
)
```

Esta consulta genera el resultado siguiente:

```
<size xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">1</size>
```

XMLCOMMENT

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLCOMMENT devuelve un valor XML con un único nodo de comentario XQuery con el argumento de entrada como contenido.

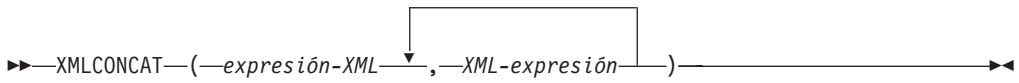
expresión-serie
Una expresión cuyo valor tiene un tipo de serie de caracteres: CHAR, VARCHAR o CLOB. El resultado de la *expresión-serie* se analiza para comprobar si se ajusta a los requisitos de los comentarios XML, como se especifica en las normas de XML 1.0. El resultado de la *expresión-serie* debe ajustarse a la expresión regular siguiente:

```
((Char - '-') | ('-' (Char - '-')))*
```

donde `Char` se define como cualquier carácter Unicode excepto los bloques de sustitución X'FFFE' y X'FFFF'. Básicamente, el comentario XML no puede contener dos guiones adyacentes y no puede acabar con un guión (SQLSTATE 2200S).

El tipo de datos del resultado es XML. Si el resultado de la *expresión-serie* puede ser nulo, el resultado puede ser nulo; si el valor de entrada es nulo, el resultado es el valor nulo.

XMLCONCAT



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLCONCAT devuelve una secuencia que contiene la concatenación de un número variable de argumentos de entrada de XML.

expresión-XML
Especifica una expresión de tipo de datos XML.

El tipo de datos del resultado es XML. El resultado es una secuencia XML que contiene la concatenación de los valores XML de entrada que no son nulos. Los valores nulos de la entrada se pasan por alto. Si el resultado de cualquier *expresión-XML* puede ser nulo, el resultado puede ser nulo; si el resultado de cada valor de entrada es nulo, el resultado es el valor nulo.

Ejemplo:

Nota: XMLCONCAT no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Construir un elemento de departamento para los departamentos A00 y B01, con una lista de empleados ordenados por nombre. Incluir un comentario preliminar inmediatamente antes del nombre del departamento.

```
SELECT XMLCONCAT(
  XMLCOMMENT(
    'Confirm these employees are on track for their product schedule'
  ),
  XMLELEMENT(
    NAME "Department",
    XMLATTRIBUTES(
      E.WORKDEPT AS "name"
    ),
    XMLAGG(
      XMLELEMENT(
        NAME "emp", E.FIRSTNME
      )
    ORDER BY E.FIRSTNME
    )
  )
)
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY E.WORKDEPT
```

Esta consulta genera el resultado siguiente:

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>DIAN</emp>
<emp>GREG</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
```

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLDOCUMENT devuelve un valor XML con un único nodo de documento XQuery con ninguno o varios nodos hijo.

expresión-XML
Una expresión que devuelve un valor XML. Un elemento de secuencia en el valor XML no debe ser un nodo de atributo (SQLSTATE 10507).

El tipo de datos del resultado es XML. Si el resultado de la *expresión-XML* puede ser nulo, el resultado puede ser nulo; si el valor de entrada es nulo, el resultado es el valor nulo.

Los hijos del nodo de documento resultante se construyen como se describe en los pasos siguientes. La expresión de entrada es una secuencia de nodos o valores atómicos, a la que se hace referencia en estos pasos como secuencia de contenido.

1. Si la secuencia de contenido contiene un nodo de documento, este nodo se sustituye en la secuencia de contenido por sus hijos.
2. Cada secuencia adyacente de uno o más valores atómicos en la secuencia de contenido se sustituye por un nodo de texto que contiene el resultado de la conversión de cada valor atómico en una serie, insertándose un único carácter en blanco entre los valores adyacentes.
3. Para cada nodo de la secuencia de contenido, se construye una nueva copia en profundidad del nodo. Una copia en profundidad de un nodo es una copia de todo el subárbol cuya raíz se encuentra en ese nodo, e incluye el propio nodo y sus descendientes. Cada nodo copiado tiene una nueva identidad de nodo.
4. Los nodos de la secuencia de contenido se convierten en hijos del nuevo nodo de documento.

La función XMLDOCUMENT ejecuta el constructor de documentos calculados XQuery. El resultado de

```
XMLQUERY('document {$E}' PASSING BY REF expresión-XML AS "E")
```

es equivalente a

```
XMLDOCUMENT( expresión-XML )
```

con la excepción del caso en el que *expresión-XML* es nula y XMLQUERY devuelve la secuencia vacía, en comparación con XMLDOCUMENT, que devuelve el valor nulo.

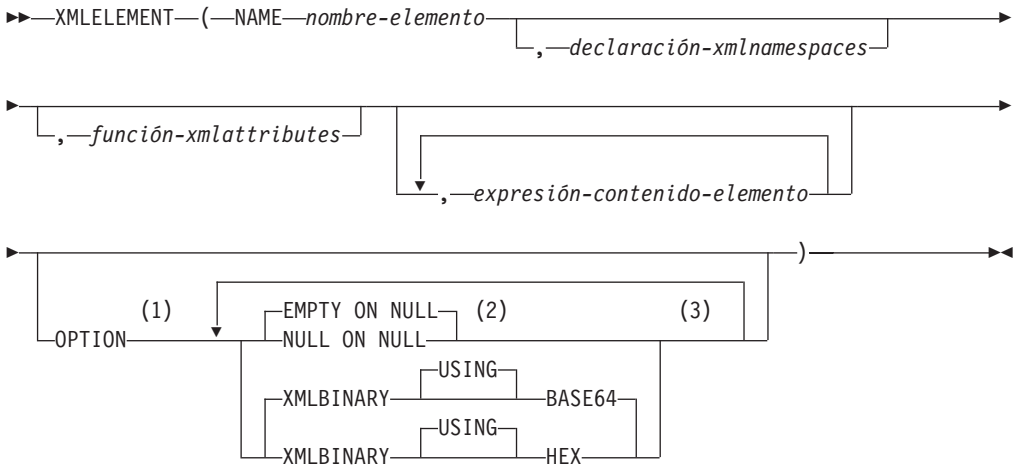
Ejemplo:

- Insertar un documento construido en una columna XML.

```
INSERT INTO T1 VALUES(
  123, (
    SELECT XMLDOCUMENT(
      XMLELEMENT(
        NAME "Emp", E.FIRSTNME || ' ' || E.LASTNAME, XMLCOMMENT(
          'This is just a simple example'
        )
      )
    )
```

```
    FROM EMPLOYEE E
    WHERE E.EMPNO = '000120'
  )
)
```

XMLELEMENT



Notas:

1 La cláusula OPTION sólo puede especificarse si, como mínimo, se ha especificado una *función-xmlattributes* o una *expresión-contenido-elemento*.

2 NULL ON NULL o EMPTY ON NULL sólo puede especificarse si, como mínimo, se ha especificado una *expresión-contenido-elemento*.

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

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLELEMENT devuelve un valor XML que es un nodo de elemento XQuery.

NAME *nombre-elemento*
Especifica el nombre de un elemento XML. El nombre es un identificador de SQL cuyo formato debe corresponder al de un nombre calificado XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. Si el nombre está calificado, el prefijo de espacio de nombres deberá declararse dentro del ámbito (SQLSTATE 42635).

declaración-xmlnamespaces
Especifica las declaraciones de espacios de nombres XML que son el resultado de la declaración XMLNAMESPACES. Los espacios de nombres que se declaran se encuentran en el ámbito de la función XMLELEMENT. Los nombres de espacios se aplican a cualquiera de las funciones XML anidadas que se encuentran dentro de la función XMLELEMENT, con independencia de si aparecen o no dentro de otra subselección.

Si no se especifica *declaración-xmlnamespaces*, las declaraciones de espacio de nombres no se asocian al elemento construido.

función-xmlattributes
Especifica los atributos XML del elemento. Los atributos son el resultado de la función XMLATTRIBUTES.

expresión-contenido-elemento
El contenido del nodo de elemento XML generado se especifica mediante una

expresión o una lista de expresiones. El tipo de datos de *expresión-contenido-elemento* no puede ser un tipo estructurado (SQLSTATE 42884). La expresión puede ser cualquier expresión SQL.

Si no se especifica *expresión-contenido-elemento*, se utiliza una serie vacía como contenido del elemento y no debe especificarse OPTION NULL ON NULL o EMPTY ON NULL.

OPTION
Especifica opciones adicionales para construir el elemento XML. Si no se especifica ninguna cláusula OPTION, el valor por omisión es EMPTY ON NULL XMLBINARY USING BASE64. Esta cláusula no afectará a las invocaciones de XMLELEMENT anidadas que se han especificado en *expresión-contenido-elemento*.

EMPTY ON NULL o NULL ON NULL
Especifica si ha de devolverse un valor nulo o un elemento nulo en caso de que los valores de cada *expresión-contenido-elemento* sean un valor nulo. Esta opción sólo afecta al manejo de nulos del contenido del elemento, no a los valores de atributo. El valor por omisión es EMPTY ON NULL.

EMPTY ON NULL
Si el valor de cada *expresión-contenido-elemento* es nulo, se devuelve un elemento vacío.

NULL ON NULL
Si el valor de cada elemento *expresión-contenido-elemento* es nulo, se devuelve un valor nulo.

XMLBINARY USING BASE64 o XMLBINARY USING HEX
Especifica la codificación de datos binarios de entrada que se da por supuesta, los datos de serie de caracteres con el atributo FOR BIT DATA o un tipo diferenciado que se basa en uno de estos tipos. La codificación se aplica a los valores de atributo o contenido del elemento. El valor por omisión es XMLBINARY USING BASE64.

XMLBINARY USING BASE64
Especifica que se da por supuesta la codificación de caracteres base64, tal como define la codificación xs:base64Binary para el tipo de esquema XML. La codificación base64 utiliza un subconjunto de US-ASCII de 65 caracteres (10 dígitos, 26 caracteres en minúsculas, 26 caracteres en mayúsculas, '+' y '/') para representar cada seis bits de los datos binarios o de bits con un carácter imprimible del subconjunto. Estos caracteres son una selección concebida especialmente para permitir su representación universal. Mediante la utilización de este método, el tamaño de los datos codificados es un 33% más grande que los datos binarios o de bits originales.

XMLBINARY USING HEX
Especifica que se da por supuesta la codificación de caracteres hexadecimales, tal como define la codificación xs:hexBinary para el tipo de esquema XML. La codificación hexadecimal representa cada byte (8 bits) con dos caracteres hexadecimales. Mediante la utilización de este método, los datos codificados doblan el tamaño de los datos binarios o de bits originales.

Esta función toma un nombre de elemento, un conjunto opcional de declaraciones de espacios de nombres, un conjunto opcional de atributos y cero o más argumentos que forman parte del contenido del elemento XML. El resultado es una secuencia XML que contiene un nodo de elemento XML o el valor nulo.

El tipo de datos del resultado es XML. Si cualquiera de los argumentos *expresión-contenido-elemento* puede ser nulo, el resultado puede ser nulo; si todos los valores del argumento *expresión-contenido-elemento* son nulos y la opción NULL ON NULL está en vigor, el resultado es el valor nulo.

Nota:

1. Cuando se construyen elementos que se copiarán como contenido de otro elemento que define espacios de nombres por omisión, los espacios de nombres por omisión deben estar no declarados explícitamente en el elemento copiado para evitar posibles errores que podrían resultar del hecho de heredar el espacio de nombres por omisión del nuevo elemento padre. Los prefijos de espacio de nombres predefinidos ('xs', 'xsi', 'xml' y 'sqlxml') también se deben declarar explícitamente cuando se utilizan.
2. **Construcción de un nodo de elemento**: El nodo de elemento resultante se construye como se indica a continuación:
   a. La *declaración-xmlnamespaces* añade un conjunto de espacios de nombres con ámbito para el elemento construido. Cada espacio de nombres con ámbito asocia un prefijo de espacio de nombres (o el espacio de nombres por omisión) a un URI de espacio de nombres. Los espacios de nombres con ámbito definen el conjunto de prefijos de espacio de nombres que están disponibles para interpretar los QName que se encuentran dentro del ámbito del elemento.
   b. Si se especifica la función-xmlattributes, se evalúa, y el resultado es una secuencia de nodos de atributos.
   c. Cada *expresión-contenido-elemento* se evalúa y el resultado se convierte en una secuencia de nodos, tal como se indica a continuación:
      - Si el tipo de resultado no es XML, se convierte en un nodo de texto XML cuyo contenido es el resultado de *expresión-contenido-elemento* correlacionado con XML en función de las normas de correlación de valores de datos SQL con valores de datos XML (vea la tabla en la que se describen las conversiones soportadas de valores que no son XML en valores XML en "Conversiones entre tipos de datos").
      - Si el tipo de resultado es XML, el resultado es, en general, una secuencia de elementos. Puede que algunos de los elementos de esa secuencia sean nodos de documentos. Cada nodo de documento de la secuencia se sustituye por la secuencia de sus hijos de nivel superior. A continuación, para cada nodo de la secuencia resultante, se construye una nueva copia en profundidad del nodo, incluidos sus hijos y atributos. Cada nodo copiado tiene una nueva identidad de nodo. Los nodos de atributos y elementos copiados conservan su anotación de tipo. Para cada secuencia adyacente de uno o más valores atómicos devueltos en la secuencia se construye un nuevo nodo de texto, que contiene el resultado de la conversión de cada valor atómico en una serie, insertándose un único carácter en blanco entre los valores adyacentes. Los nodos de texto adyacentes de la secuencia de contenido se fusionan en un único nodo de texto mediante la concatenación de su contenido, sin la intervención de caracteres en blanco. Después de la concatenación, cualquier nodo de texto cuyo contenido sea una serie de longitud cero se suprimirá de la secuencia de contenido.
   d. La secuencia resultante de atributos XML y las secuencias resultantes de todas las especificaciones *expresión-contenido-elemento* se concatenan en una única secuencia, que se denomina secuencia de contenido. Cualquier secuencia de nodos de texto adyacentes de la secuencia de contenido se fusiona en un único nodo de texto. Si todos los argumentos

expresión-contenido-elemento son series vacías, o si no se ha especificado un argumento *expresión-contenido-elemento*, se devuelve un elemento vacío.

e. La secuencia de contenido no debe contener un nodo de atributo a continuación de un nodo que no sea un nodo de atributo (SQLSTATE 10507). Los nodos de atributos que se producen en la secuencia de contenido se convierten en atributos del nuevo nodo de elemento. No deben tener el mismo nombre dos o más de estos nodos de atributos (SQLSTATE 10503). Se crea una declaración de espacio de nombres correspondiente a cualquier espacio de nombres utilizado en los nombres de los nodos de atributos si el URI de espacio de nombres no se encuentra en los espacios de nombres con ámbito del elemento construido.

f. Los nodos de elementos, texto, comentarios e instrucciones de proceso de la secuencia de contenido se convierten en los hijos del nodo de elemento construido.

g. El nodo de elemento construido recibe una anotación de tipo `xs:anyType` y cada uno de sus atributos recibe una anotación de tipo `xdt:untypedAtomic`. El nombre de nodo del nodo de elemento construido es el nombre-elemento que se ha especificado después de la palabra clave NAME.

3. **Normas para utilizar espacios de nombres dentro de XMLELEMENT**: Examine las siguientes normas relacionadas con el ámbito de los espacios de nombres:

   - Los espacios de nombres declarados en la declaración XMLNAMESPACES son los espacios de nombres con ámbito del nodo de elemento que ha construido la función XMLELEMENT. Si el nodo de elemento está serializado, cada uno de sus espacios de nombres con ámbito se serializará como un atributo de espacio de nombres a menos que sea un espacio de nombres con ámbito del padre del nodo de elemento y el elemento padre también esté serializado.
   - Si existe una expresión XMLQUERY o XMLEXISTS es una *expresión-contenido-elemento*, los espacios de nombres se convierten en los nombres de espacios estáticamente conocidos de la expresión XQuery de XMLQUERY o XMLEXISTS. Los espacios de nombres estáticamente conocidos se utilizan para resolver los QName de la expresión XQuery. Si el prólogo XQuery declara un espacio de nombres con el mismo prefijo, dentro del ámbito de la expresión XQuery, el espacio de nombres declarado en el prólogo alterará temporalmente los espacios de nombres declarados en la declaración XMLNAMESPACES.
   - Si un atributo del elemento construido procede de una *expresión-contenido-elemento*, puede que su espacio de nombres todavía no se haya declarado como un espacio de nombres con ámbito del elemento construido y, en este caso, se crea un nuevo espacio de nombres para éste. Si esto puede generar un conflicto, lo que significa que el prefijo del nombre de atributo ya se han vinculado a un URI distinto mediante un espacio de nombres con ámbito, DB2 genera un prefijo que no causa ningún conflicto y el prefijo utilizado en el nombre de atributo se cambia por el nuevo prefijo y se crea un espacio de nombres para este nuevo prefijo. El nuevo prefijo generado sigue el patrón siguiente: "db2ns-xx", donde "x" es un carácter seleccionado en el juego de caracteres [A a Z, a a z, 0 a 9]. Por ejemplo:

```
VALUES XMLELEMENT(
  NAME "c", XMLQUERY(
    'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b'
      PASSING XMLPARSE(
```

```
          DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
        ) AS "m"
    )
  )
```

devuelve:

```
<c xmlns:tst="www.ipo.com" tst:b="2"/>
```

Un segundo ejemplo:

```
VALUES XMLELEMENT(
  NAME "tst:c", XMLNAMESPACES(
    'www.tst.com' AS "tst"
  ),
  XMLQUERY(
    'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b'
      PASSING XMLPARSE(
        DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
      ) AS "m"
  )
)
```

devuelve:

```
<tst:c xmlns:tst="www.tst.com" xmlns:db2ns-a1="www.ipo.com"
  db2ns-a1:b="2"/>
```

Ejemplos:

Nota: XMLELEMENT no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Construir un elemento con la opción NULL ON NULL.

```
SELECT E.FIRSTNME, E.LASTNAME, XMLELEMENT(
  NAME "Emp", XMLELEMENT(
    NAME "firstname", E.FIRSTNME
  ),
  XMLELEMENT(
    NAME "lastname", E.LASTNAME
  )
  OPTION NULL ON NULL
)
AS "Result"
FROM EMPLOYEE E
WHERE E.EDLEVEL = 12
```

Esta consulta genera el resultado siguiente:

```
FIRSTNME     LASTNAME        Emp
JOHN         PARKER          <Emp><firstname>JOHN</firstname>
                               <lastname>PARKER</lastname></Emp>
MAUDE        SETRIGHT        <Emp><firstname>MAUDE</firstname>
                               <lastname>SETRIGHT</lastname></Emp>
MICHELLE     SPRINGER        <Emp><firstname>MICHELLE</firstname>
                               <lastname>SPRINGER</lastname></Emp>
```

- Producir un elemento con una lista de elementos anidados como elementos hijos.

```
SELECT XMLELEMENT(
  NAME "Department", XMLATTRIBUTES(
    E.WORKDEPT AS "name"
  ),
  XMLAGG(
    XMLELEMENT(
      NAME "emp", E.FIRSTNME
```

```
      )
      ORDER BY E.FIRSTNME
    )
  )
  AS "dept_list"
  FROM EMPLOYEE E
  WHERE E.WORKDEPT IN ('A00', 'B01')
  GROUP BY WORKDEPT
```

Esta consulta genera el resultado siguiente:

```
dept_list
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

- Creación de elementos XML anidados que especifican un espacio de nombres de elemento XML por omisión y que utilizan una subselección

```
SELECT XMLELEMENT(
         NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
         XMLATTRIBUTES(cid),
            (SELECT
               XMLAGG(
                 XMLELEMENT(
                   NAME "poid", poid
                 )
               )
             FROM purchaseorder
             WHERE purchaseorder.custid = customer.cid
            )
       )
FROM customer
WHERE cid = '1002'
```

La sentencia devuelve el documento XML siguiente con el espacio de nombres de elemento por omisión declarado en el elemento root:

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid>5000</poid>
  <poid>5003</poid>
  <poid>5006</poid>
</root>
```

- Utilización de una expresión de tabla común con espacios de nombres XML

  Cuando se construye un elemento XML con una expresión de tabla común y el elemento se utiliza en otro ubicación de la misma sentencia de SQL, las declaraciones de espacio de nombres deben especificarse como parte de la construcción del elemento. La sentencia siguiente especifica el espacio de nombres XML por omisión tanto en la expresión de tabla común que utiliza la tabla PURCHASEORDER para crear los elementos poid como en la sentencia SELECT que utiliza la tabla CUSTOMER para crear el elemento root.

```
WITH tempid(id, elem) AS
  (SELECT custid, XMLELEMENT(NAME "poid",
     XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
       poid)
   FROM purchaseorder )
SELECT XMLELEMENT(NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
            XMLATTRIBUTES(cid),
            (SELECT XMLAGG(elem)
```

```
            FROM tempid
            WHERE tempid.id = customer.cid )
       )
FROM customer
WHERE cid = '1002'
```

La sentencia devuelve el documento XML siguiente con un espacio de nombres de elemento por omisión declarado en el elemento root.

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid>5000</poid>
  <poid>5003</poid>
  <poid>5006</poid>
</root>
```

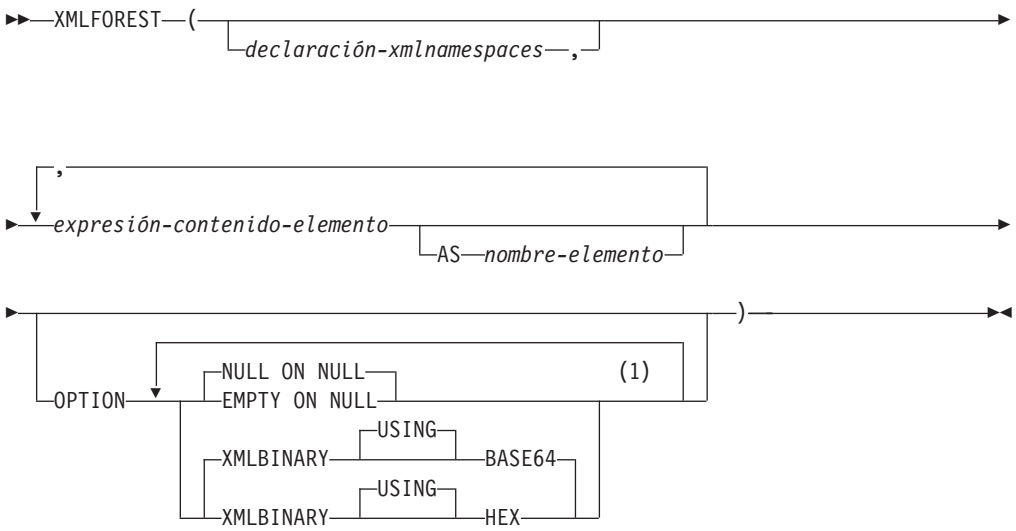
En la sentencia siguiente, el espacio de nombres del elemento default se declara sólo en la sentencia SELECT que utiliza la tabla CUSTOMER para crear el elemento root:

```
WITH tempid(id, elem) AS
    (SELECT custid, XMLELEMENT(NAME "poid", poid)
     FROM purchaseorder )
SELECT XMLELEMENT(NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
            XMLATTRIBUTES(cid),
            (SELECT XMLAGG(elem)
             FROM tempid
             WHERE tempid.id = customer.cid )
       )
FROM customer
WHERE cid = '1002'
```

La sentencia devuelve el documento XML siguiente con el espacio de nombres de elemento por omisión declarado en el elemento root. Como los elementos poid se crean en la expresión de tabla común sin una declaración de espacio de nombres del elemento default, no se define el espacio de nombres del elemento default para los elementos poid. En el documento XML, el espacio de nombres del elemento default para los elementos poid se establece en una serie vacía "", porque el espacio de nombres del elemento default para los elementos poid no se ha definido y los elementos poid no pertenecen al espacio de nombres del elemento default del elemento root xmlns="http://mytest.uri".

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid xmlns="">5000</poid>
  <poid xmlns="">5003</poid>
  <poid xmlns="">5006</poid>
</root>
```

XMLFOREST



Notas:

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

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLFOREST devuelve un valor XML que es una secuencia de nodos de elemento XQuery.

declaración-xmlnamespaces
Especifica las declaraciones de espacios de nombres XML que son el resultado de la declaración XMLNAMESPACES. Los espacios de nombres que se declaran se encuentran en el ámbito de la función XMLFOREST. Los nombres de espacios se aplican a cualquiera de las funciones XML anidadas que se encuentran dentro de la función XMLFOREST, con independencia de si aparecen o no dentro de otra subselección.

Si no se especifica *declaración-xmlnamespaces*, las declaraciones de espacio de nombres no se asocian a los elementos construidos.

expresión-contenido-elemento
El contenido del nodo de elemento XML generado se especifica mediante una expresión. El tipo de datos de *expresión-contenido-elemento* no puede ser un tipo estructurado (SQLSTATE 42884). La expresión puede ser cualquier expresión SQL. Si la expresión no es una referencia de columna simple, debe especificarse un nombre de elemento.

AS *nombre-elemento*
Especifica el nombre de elemento XML como identificador SQL. El nombre del elemento debe tener el formato de un nombre calificado XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. Si el nombre está calificado, el prefijo de espacio de nombres deberá declararse dentro del ámbito (SQLSTATE 42635). Si no se especifica el *nombre-elemento*, *expresión-contenido-elemento* debe ser un nombre de columna (SQLSTATE 42703).

El nombre de elemento se crea a partir del nombre de columna, utilizando la correlación con elusión de caracteres (″fully escaped″) desde un nombre de columna a un QName.

OPTION
Especifica opciones adicionales para construir el elemento XML. Si no se especifica ninguna cláusula OPTION, el valor por omisión es NULL ON NULL XMLBINARY USING BASE64. Esta cláusula no afectará a las invocaciones de XMLELEMENT anidadas que se han especificado en *expresión-contenido-elemento*.

EMPTY ON NULL o NULL ON NULL
Especifica si ha de devolverse un valor nulo o un elemento nulo en caso de que los valores de cada *expresión-contenido-elemento* sean un valor nulo. Esta opción sólo afecta al manejo de nulos del contenido del elemento, no a los valores de atributo. El valor por omisión es NULL ON NULL.

EMPTY ON NULL
Si el valor de cada *expresión-contenido-elemento* es nulo, se devuelve un elemento vacío.

NULL ON NULL
Si el valor de cada elemento *expresión-contenido-elemento* es nulo, se devuelve un valor nulo.

XMLBINARY USING BASE64 o XMLBINARY USING HEX
Especifica la codificación de datos binarios de entrada que se da por supuesta, los datos de serie de caracteres con el atributo FOR BIT DATA o un tipo diferenciado que se basa en uno de estos tipos. La codificación se aplica a los valores de atributo o contenido del elemento. El valor por omisión es XMLBINARY USING BASE64.

XMLBINARY USING BASE64
Especifica que se da por supuesta la codificación de caracteres base64, tal como define la codificación xs:base64Binary para el tipo de esquema XML. La codificación base64 utiliza un subconjunto de US-ASCII de 65 caracteres (10 dígitos, 26 caracteres en minúsculas, 26 caracteres en mayúsculas, '+' y '/') para representar cada seis bits de los datos binarios o de bits con un carácter imprimible del subconjunto. Estos caracteres son una selección concebida especialmente para permitir su representación universal. Mediante la utilización de este método, el tamaño de los datos codificados es un 33% más grande que los datos binarios o de bits originales.

XMLBINARY USING HEX
Especifica que se da por supuesta la codificación de caracteres hexadecimales, tal como define la codificación xs:hexBinary para el tipo de esquema XML. La codificación hexadecimal representa cada byte (8 bits) con dos caracteres hexadecimales. Mediante la utilización de este método, los datos codificados doblan el tamaño de los datos binarios o de bits originales.

Esta función toma un conjunto opcional de declaraciones de espacios de nombres y uno o varios argumentos que forman el contenido de nombre y elemento para uno o varios nodos de elemento. El resultado es una secuencia XML que contiene una secuencia de nodos de elemento XQuery o el valor nulo.

El tipo de datos del resultado es XML. Si cualquiera de los argumentos *expresión-contenido-elemento* puede ser nulo, el resultado puede ser nulo; si todos los

valores del argumento *expresión-contenido-elemento* son nulos y la opción NULL ON NULL está en vigor, el resultado es el valor nulo.

La función XMLFOREST se puede expresar mediante XMLCONCAT y XMLELEMENT. Por ejemplo, las dos expresiones siguientes son equivalentes desde el punto de vista semántico.

```
XMLFOREST(declaración-xmlnamespaces, arg1 AS name1, arg2 AS name2 ...)
```

```
XMLCONCAT(
  XMLELEMENT(
    NAME name1, declaración-xmlnamespaces, arg1
  ),
  XMLELEMENT(
    NAME name2, declaración-xmlnamespaces, arg2
  )
  ...
)
```

Nota:

1. Cuando se construyen elementos que se copiarán como contenido de otro elemento que define espacios de nombres por omisión, los espacios de nombres por omisión deben estar no declarados explícitamente en el elemento copiado para evitar posibles errores que podrían resultar del hecho de heredar el espacio de nombres por omisión del nuevo elemento padre. Los prefijos de espacio de nombres predefinidos ('xs', 'xsi', 'xml' y 'sqlxml') también se deben declarar explícitamente cuando se utilizan.

Ejemplo:

Nota: XMLFOREST no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Construir un bosque de elementos con un espacio de nombres por omisión.

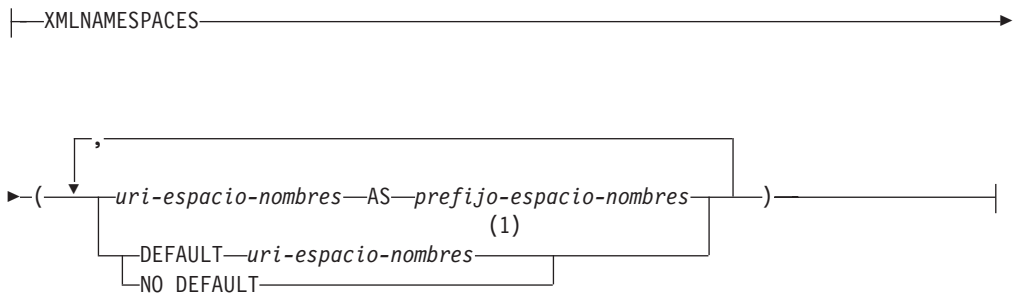
```
SELECT EMPNO,
  XMLFOREST(
    XMLNAMESPACES(
      DEFAULT 'http://hr.org', 'http://fed.gov' AS "d"
    ),
    LASTNAME, JOB AS "d:job"
  )
AS "Result"
FROM EMPLOYEE
WHERE EDLEVEL = 12
```

Esta consulta genera el resultado siguiente:

```
EMPNO      Result
000290 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>

000310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>

200310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SPRINGER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
```

XMLNAMESPACES

declaración-xmlnamespaces:



Notas:

1 Sólo puede especificarse DEFAULT o NO DEFAULT una vez en los argumentos de XMLNAMESPACES.

El esquema es SYSIBM. El nombre de la declaración no puede especificarse como un nombre calificado.

La declaración XMLNAMESPACES construye las declaraciones de espacios de nombres a partir de los argumentos. Esta declaración únicamente puede emplearse como argumento de funciones específicas como XMLELEMENT, XMLFOREST y XMLTABLE. El resultado es una o varias declaraciones de espacios de nombres XML que contienen espacios de nombres con ámbito para cada uno de los valores de entrada no nulos.

uri-espacio-nombres
Especifica el identificador de recursos universal (URI) de espacio de nombres como una constante de serie de caracteres de SQL. Esta constante de serie de caracteres no puede estar vacía si se utiliza con un *prefijo-espacio-nombres* (SQLSTATE 42815).

prefijo-espacio-nombres
Especifica un prefijo de espacio de nombres. El prefijo es un identificador de SQL que debe tener el formato de un nombre XML NCName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. El prefijo no puede ser `xml` ni `xmlns` y el prefijo debe ser exclusivo dentro de la lista de declaraciones de espacios de nombres (SQLSTATE 42635).

DEFAULT *uri-espacio-nombres*
Especifica el espacio de nombres por omisión que debe utilizarse en el ámbito de esta declaración de espacio de nombres. El *uri-espacio-nombres* es válido para los nombres no calificados del ámbito salvo que otra declaración DEFAULT o una declaración NO DEFAULT altere temporalmente este valor en un ámbito anidado.

NO DEFAULT
Especifica que no se utilizará ningún espacio de nombres por omisión en el ámbito de esta declaración de espacio de nombres. No existe ningún espacio de nombres por omisión en el ámbito salvo que una declaración DEFAULT altere temporalmente este valor en un ámbito anidado.

El tipo de datos del resultado es XML. El resultado es una declaración de espacio de nombres XML para cada uno de los espacios de nombres indicados. El resultado no puede ser nulo.

Ejemplos:

Nota: XMLNAMESPACES no inserta espacios en blanco ni caracteres de nueva línea en la salida. Todas las salidas de los ejemplos se han formateado para mejorar la legibilidad.

- Genere un elemento XML denominado adm:employee y un atributo XML adm:department, ambos asociados con un espacio de nombres cuyo prefijo es adm.

```
SELECT EMPNO, XMLELEMENT(
  NAME "adm:employee", XMLNAMESPACES(
    'http://www.adm.com' AS "adm"
  ),
  XMLATTRIBUTES(
    WORKDEPT AS "adm:department"
  ),
LASTNAME
)
FROM EMPLOYEE
WHERE JOB = 'ANALYST'
```

  Esta consulta genera el resultado siguiente:

```
000130 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  QUINTANA</adm:employee>
000140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  NICHOLLS</adm:employee>
200140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  NATZ</adm:employee>
```

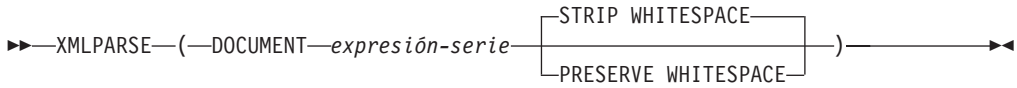
- Genere un elemento XML denominado 'employee' asociado con un espacio de nombres por omisión y un subelemento denominado 'job' que no utilice un espacio de nombres por omisión y cuyo subelemento 'department' utilice un espacio de nombres por omisión.

```
SELECT EMP.EMPNO, XMLELEMENT(
  NAME "employee", XMLNAMESPACES(
    DEFAULT 'http://hr.org'
  ),
  EMP.LASTNAME, XMLELEMENT(
    NAME "job", XMLNAMESPACES(
      NO DEFAULT
    ),
    EMP.JOB, XMLELEMENT(
      NAME "department", XMLNAMESPACES(
        DEFAULT 'http://adm.org'
      ),
      EMP.WORKDEPT
    )
  )
)
FROM EMPLOYEE EMP
WHERE EMP.EDLEVEL = 12
```

  Esta consulta genera el resultado siguiente:

```
000290 <employee xmlns="http://hr.org">PARKER<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
000310 <employee xmlns="http://hr.org">SETRIGHT<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
200310 <employee xmlns="http://hr.org">SPRINGER<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
```

XMLPARSE



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLPARSE analiza el argumento como un documento XML y devuelve un valor XML.

DOCUMENT
Especifica que la expresión de serie de caracteres que se analizará debe evaluarse como un documento XML con el formato correcto conforme a la especificación XML 1.0, según la modificación de la recomendación de espacios de nombres XML (SQLSTATE 2200M).

expresión-serie
Especifica una expresión que devuelve una serie de caracteres o un valor BLOB. Si se utiliza un marcador de parámetro, debe convertirse explícitamente en uno de los tipos de datos soportados.

STRIP WHITESPACE o PRESERVE WHITESPACE
Especifica si deben conservarse o no los espacios en blanco del argumento de entrada. Si no se especifica ningún valor, el valor por omisión es STRIP WHITESPACE.

STRIP WHITESPACE
Especifica que los nodos de texto que contengan únicamente caracteres de espacio en blanco hasta 1000 bytes de longitud se eliminarán, salvo que el elemento continente más próximo tenga el atributo xml:space='preserve'. Si algún nodo de texto empieza con más de 1000 bytes de espacios en blanco, se devuelve un error (SQLSTATE 54059).

Esta opción también afecta a los caracteres de espacio en blanco de la sección CDATA. Las definiciones DTD pueden tener declaraciones DOCTYPE de elementos, pero los modelos de contenido de los elementos no se utilizan para determinar si se eliminan o no los espacios en blanco.

PRESERVE WHITESPACE
Especifica que deben conservarse todos los espacios en blanco, aun cuando el elemento continente más próximo tenga el atributo xml:space='default'.

El tipo de datos del resultado es XML. Si el resultado de la *expresión-serie* puede ser nulo, el resultado puede ser nulo; si el resultado de la *expresión-serie* es nulo, el resultado es el valor nulo.

Nota:

1. **Codificación de la serie de entrada**: La serie de entrada puede contener una declaración XML que identifica la codificación de los caracteres del documento XML. Si se pasa la serie a la función XMLPARSE como una serie de caracteres, se convertirá en la página de códigos en el servidor de bases de datos. Esta página de códigos puede ser distinta de la página de códigos de origen y la codificación identificada en la declaración XML.

   Por consiguiente, las aplicaciones deben evitar la utilización directa de XMLPARSE con entrada de serie de caracteres y deben enviar series con

documentos XML directamente utilizando variable del lenguaje principal para mantener la coincidencia entre la página de códigos externa y la codificación de la declaración XML. Si debe utilizarse XMLPARSE en esta situación, debe especificarse un tipo BLOB como argumento para evitar la conversión de la página de códigos.

2. **Manejo de las DTD**: Las definiciones de tipo de documento externas (DTD) y las entidades externas deben estar registradas en una base de datos. Se comprueba que tanto las DTD internas como las externas tienen una sintaxis válida. Durante el proceso de análisis también se realizan las acciones siguientes:
   - Se aplican los valores por omisión definidos por las DTD internas y externas.
   - Las referencias de entidad y las entidades de parámetro se sustituyen por sus formas expandidas.
   - Si una DTD interna y una DTD externa definen el mismo elemento, se devuelve un error (SQLSTATE 2200M).
   - Si una DTD interna y una DTD externa definen la misma entidad o el mismo atributo, se elige la definición interna.

   Después del análisis, las DTD y las entidades internas, así como las referencias a las DTD y entidades externas, no se conservan en la representación del valor almacenada.
3. **Conversión de caracteres en bases de datos no UTF-8**: La conversión de página de códigos se produce cuando se analiza un documento XML en un servidor de bases de datos no Unicode, si el documento se pasa desde una variable del lenguaje principal o marcador de parámetro de un tipo de datos de caracteres o desde un literal de serie de caracteres. La acción de analizar un documento XML utilizando un marcador de parámetro o variable del lenguaje principal del tipo XML, BLOB o FOR BIT DATA (CHAR FOR BIT DATA o VARCHAR FOR BIT DATA) impedirá la conversión de página de códigos. Cuando se utiliza un tipo de datos de caracteres, deben tomarse precauciones para asegurarse de que todos los caracteres del documento XML tengan un elemento de código coincidente en la página de códigos de la base de datos destino, en caso contrario es posible que se introduzcan caracteres de sustitución. Puede utilizarse el parámetro de configuración **enable_xmlchar** para ayudar a asegurar la integridad de los datos XML almacenados en una base de datos no Unicode. Establecer este parámetro en ″NO″ bloquea la inserción de documentos XML de tipos de datos de caracteres. Los tipos de datos BLOB y FOR BIT DATA se siguen permitiendo, porque los documentos pasados a una base de datos utilizando estos tipos de datos evitan la conversión de página de códigos.

Ejemplo

Si se utiliza la opción PRESERVE WHITESPACE se conservan los caracteres de espacio en blanco del documento XML insertado en la tabla, incluidos los existentes en el elemento de descripción.

```
INSERT INTO PRODUCT VALUES ('100-103-99','Bolsa de herramientas',14.95,NULL,NULL,NULL,
XMLPARSE( DOCUMENT
  '<produce xmlns="http://posample.org" pid="100-103-99">
    <description>
     <name>Bolsa de herramientas</name>
     <details>
      Bolsa de herramientas Super Deluxe:
      - 26 pulgadas de longitud, 12 pulgadas de ancho
      - Mango almohadillado curvo
      - Cerradura de bloqueo
```

```
     - Bolsillos exteriores reforzados
    </details>
    <price>14,95</price>
    <weight>3 kg</weight>
   </description>
  </product>' PRESERVE WHITESPACE ));
```

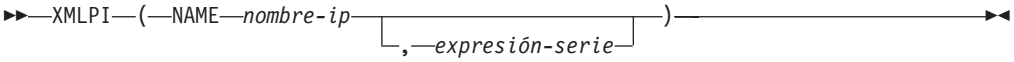
Al ejecutar la sentencia de selección siguiente:

```
SELECT XMLQUERY ('$d/*:product/*:description/*:details' PASSING DESCRIPTION as "d" )
FROM PRODUCT WHERE PID = '100-103-99' ;
```

se devuelve el elemento details con los caracteres de espacio en blanco:

```
<details xmlns="http://posample.org">
  Bolsa de herramientas Super Deluxe:
  - 26 pulgadas de longitud, 12 pulgadas de ancho
  - Mango almohadillado curvo
  - Cerradura de bloqueo
  - Bolsillos exteriores reforzados
</details>
```

XMLPI



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLPI devuelve un valor XML con un nodo de instrucción de proceso XQuery.

NAME *nombre-ip*
Especifica el nombre de una instrucción de proceso. El nombre es un identificador de SQL que debe tener el formato de un nombre XML NCName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. El nombre no puede ser 'xml' en ninguna combinación de mayúsculas y minúsculas (SQLSTATE 42634).

expresión-serie
Expresión que devuelve un valor que es una serie de caracteres. La serie obtenida se convierte a UTF-8 y debe ajustarse al contenido de una instrucción de proceso XML conforme a la especificación de las normas de XML 1.0 (SQLSTATE 2200T):

- La serie no puede contener la subserie '?>' ya que esta subserie termina una instrucción de proceso.
- Cada uno de los caracteres de la serie puede ser cualquier carácter Unicode excepto los bloques de sustitución, X'FFFE' y X'FFFF'.

La serie obtenida pasa a ser el contenido del nodo de instrucción de proceso construido.

El tipo de datos del resultado es XML. Si el resultado de la *expresión-serie* puede ser nulo, el resultado puede ser nulo; si el resultado de la *expresión-serie* es nulo, el resultado es el valor nulo. Si la *expresión-serie* es una serie vacía o no se especifica, se devuelve un nodo de instrucción de proceso vacío.

Ejemplos:

- Generar un nodo de instrucción de proceso XML.

```
SELECT XMLPI(
  NAME "Instruction", 'Push the red button'
)
FROM SYSIBM.SYSDUMMY1
```

Esta consulta genera el resultado siguiente:

```
<?Instruction Push the red button?>
```

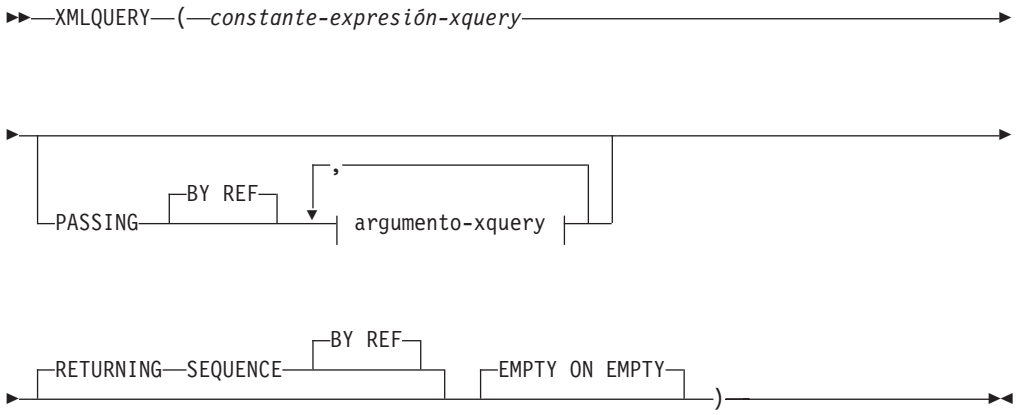
- Generar un nodo de instrucción de proceso XML vacío.

```
SELECT XMLPI(
  NAME "Warning"
)
FROM SYSIBM.SYSDUMMY1
```

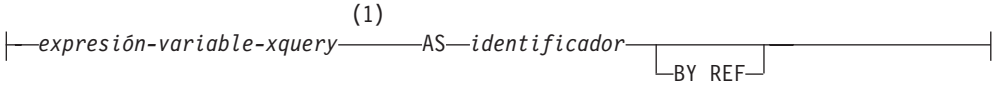
Esta consulta genera el resultado siguiente:

```
<?Warning ?>
```

XMLQUERY



argumento-xquery:



Notas:

1 El tipo de datos de la expresión no puede ser DECFLOAT.

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLQUERY devuelve un valor XML a partir de la evaluación de una expresión XQuery utilizando posiblemente los argumentos de entrada especificados como variables XQuery.

constante-expresión-xquery
Especifica una constante de serie de caracteres de SQL que se interpreta como una expresión XQuery mediante la sintaxis del lenguaje XQuery soportada. La serie de la constante se convierte a UTF-8 antes de analizarse como sentencia XQuery. La expresión XQuery se ejecuta utilizando un conjunto opcional de valores XML de entrada y devuelve una secuencia de salida que también se devuelve como valor de la expresión XMLQUERY. El valor para la *constante-expresión-xquery* no debe ser una serie vacía o una serie de caracteres en blanco (SQLSTATE 10505).

PASSING
Especifica los valores de entrada y el modo en que pasan a la expresión XQuery especificada por la *constante-expresión-xquery*. Por omisión, cada nombre de columna exclusivo en el ámbito en el que se invoca la función se pasa implícitamente a la expresión XQuery utilizando el nombre de la columna como nombre de variable. Si un *identificador* de un argumento-xquery especificado coincide con el nombre de columna con ámbito, entonces el argumento-xquery explícito se pasa a la expresión XQuery alterando temporalmente dicha columna implícita.

BY REF
Especifica que el mecanismo de pase por omisión es por referencia para cualquier *expresión-variable-xquery* de tipo de datos XML y para el valor

devuelto. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML.

Esta cláusula no afectará al modo en que se pasan los valores que no son XML. Los valores que no son XML crean una copia nueva del valor durante la conversión a XML.

argumento-xquery
Especifica un argumento que se pasará a la expresión XQuery especificada por *constante-expresión-xquery*. Un argumento especifica un valor y la forma en que ese valor se debe pasar. El argumento contiene una expresión SQL que se evalúa.

- Si el valor del resultado es del tipo XML, pasa a ser un *valor-xml-entrada*. Un valor XML nulo se convierte en una secuencia XML vacía.
- Si el valor del resultado no es del tipo XML, debe ser convertible al tipo de datos XML. Un valor nulo se convierte en una secuencia XML vacía. El valor convertido se transforma en un *valor-xml-entrada*.

Cuando se evalúa la *constante-expresión-xquery*, se presenta un valor igual a *valor-xml-entrada* a una variable XQuery y la cláusula AS especifica un nombre.

expresión-variable-xquery
Especifica una expresión SQL cuyo valor está disponible para la expresión XQuery especificada por *constante-expresión-xquery* durante la ejecución. La expresión no puede contener una referencia de secuencia (SQLSTATE 428F9) ni una función OLAP (SQLSTATE 42903). El tipo de datos de la expresión no puede ser DECFLOAT.

AS *identificador*
Especifica que el valor generado por *expresión-variable-xquery* se pasará a *constante-expresión-xquery* como una variable XQuery. El nombre de la variable será *identificador*. El signo de dólar inicial ($) que precede a los nombres de variable en el lenguaje XQuery no se incluye en el *identificador*. El identificador debe ser un nombre de variable XQuery válido y debe tener el formato de un nombre XML NCName (SQLSTATE 42634). El identificador no debe tener más de 128 bytes de longitud. Dos argumentos de la misma cláusula PASSING no pueden emplear el mismo identificador (SQLSTATE 42711).

BY REF
Indica que un valor de entrada XML se debe pasar por referencia. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML. Si no se especifica BY REF a continuación de una *expresión-variable-xquery*, los argumentos XML se pasan mediante el

mecanismo de pase por omisión que se proporciona mediante la sintaxis situada tras la palabra clave PASSING. Esta opción no puede especificarse para valores que no son XML. Cuando se pasa un valor que no es XML, el valor se convierte a XML; este proceso crea una copia.

RETURNING SEQUENCE
Indica que la expresión XMLQUERY devuelve una secuencia.

BY REF
Indica que el resultado de la expresión XQuery se devuelve por referencia. Si este valor contiene nodos, toda expresión que utilice el valor de retorno de la expresión XQuery recibirá directamente las referencias de nodo, con lo que se conservan todas las propiedades de los nodos, como las identidades de nodo originales y el orden del documento. Los nodos a los que se haga referencia seguirán conectados dentro de sus árboles de nodos. Si no se especifica la cláusula BY REF y se especifica PASSING, se utiliza el mecanismo de pase por omisión. Si no se especifica BY REF y no se especifica PASSING, el mecanismo de retorno por omisión es BY REF.

EMPTY ON EMPTY
Especifica un resultado de secuencia vacía obtenido del proceso de la expresión XQuery se devuelve como una secuencia vacía.

El tipo de datos del resultado es XML; no puede ser nulo.

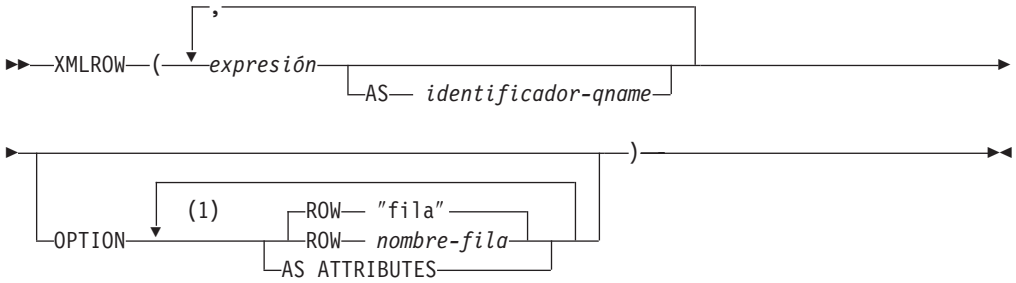
Si la evaluación de la expresión XQuery genera un error, la función XMLQUERY devuelve el error de XQuery (clase de SQLSTATE '10').

Nota:

1. **Restricciones de uso de XMLQUERY**: La función XMLQUERY no puede formar parte de los elementos siguientes:
   - Parte de la cláusula ON asociada a un operador JOIN o una sentencia MERGE (SQLSTATE 42972)
   - Parte de la cláusula GENERATE KEY USING o RANGE THROUGH de la sentencia CREATE INDEX EXTENSION (SQLSTATE 428E3)
   - Parte de la cláusula FILTER USING de la sentencia CREATE FUNCTION (escalar externa) o la cláusula FILTER USING de la sentencia CREATE INDEX EXTENSION (SQLSTATE 428E4)
   - Parte de una restricción de comprobación o de una expresión de generación de columnas (SQLSTATE 42621)
   - Una cláusula-group-by (SQLSTATE 42822)
   - Un argumento de una función-columna (SQLSTATE 42607)
2. **XMLQUERY como subconsulta**: Una expresión XMLQUERY que actúa como subconsulta puede estar limitada por sentencias que restringen las subconsultas.

XMLROW

La función XMLROW devuelve un valor XML con un único nodo de documento XQuery que contiene un nodo de elemento de nivel superior.



Notas:

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

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

expresión
El contenido de cada nodo de elemento XML generado se especifica mediante una expresión. El tipo de datos de la expresión no puede ser un tipo estructurado (SQLSTATE 42884). La expresión puede ser cualquier expresión SQL. Si la expresión no es una referencia de columna simple, debe especificarse un nombre de elemento.

AS *identificador-qname*
Especifica el nombre de elemento XML o nombre de atributo como identificador SQL. El *identificador-qname* debe tener el formato de un nombre de calificador XML o QName (SQLSTATE 42634). Para obtener más información sobre los nombres válidos, consulte las especificaciones sobre espacios de nombres W3C XML. Si el nombre está calificado, el prefijo de espacio de nombres deberá declararse dentro del ámbito (SQLSTATE 42635). Si no se especifica *identificador-qname*, *expresión* debe ser un nombre de columna (SQLSTATE 42703).). El nombre de elemento o nombre de atributo se crea a partir del nombre de columna, utilizando la correlación con elusión de caracteres ("fully escaped") desde un nombre de columna a un QName.

OPTION
Especifica opciones adicionales para construir el valor XML. Si no se especifica ninguna cláusula OPTION, se aplica el comportamiento por omisión.

AS ATTRIBUTES
Especifica que cada expresión está correlacionada a un valor de atributo que tenga nombre de columna o *identificador-qname* que sirva como nombre de atributo.

ROW *nombre-fila*
Especifica el nombre del elemento al que está correlacionado cada fila. Si no se especifica esta opción, el nombre de elemento por omisión es "fila".

Notas

Por omisión, cada fila del conjunto de resultados se correlaciona con un valor XML del siguiente modo:

- Cada fila se transforma en un elemento XML denominado "fila" y cada columna se transforma en un elemento anidado con el nombre de columna como nombre de elemento.
- El comportamiento de manejo de nulos por omisión es NULL ON NULL. Un valor de NULL de una columna se correlaciona con la ausencia del subelemento. Si todos los valores de columna son NULL, la función devolverá un valor NULL.
- El esquema de codificación binario para los tipos de datos BLOB y FOR BIT DATA es la codificación base64Binary.
- Un nodo de documento se añadirá implícitamente al elemento de fila para hacer que el resultado de XML sea un documento XML con una sola raíz bien formado.

Ejemplos

Suponga que existe la siguiente tabla T1 en la que las columnas C1 y C2 contienen datos numéricos almacenados en un formato relacional:

```
C1          C2
----------- -----------
          1           2
          -           2
          1           -
          -           -

  4 registro(s) seleccionado(s).
```

- En el siguiente ejemplo se muestra una consulta XMLRow y un fragmento de salida con el comportamiento por omisión, mediante la utilización de una secuencia de los elementos de fila para representar la tabla:

```
SELECT XMLROW(C1, C2) FROM T1
```

```
<row><C1>1</C1><C2>2</C2></row>
<row><C2>2</C2></row>
<row><C1>1</C1></row>

  4 registro(s) seleccionado(s).
```

- El siguiente ejemplo muestra un fragmento de salida y consulta XMLRow con correlación céntrica de atributos. En vez de aparecer como elementos anidados como en el ejemplo anterior, los datos relacionales se correlacionan a los atributos de elementos:

```
SELECT XMLROW(C1, C2 OPTION AS ATTRIBUTES) FROM T1
```

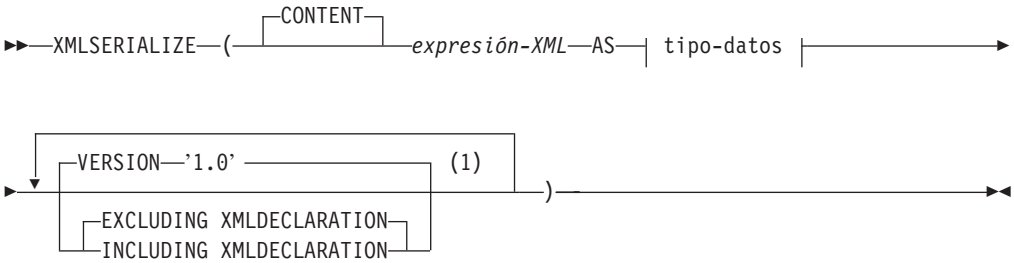
```
<row C1="1" C2="2"/>
<row C2="2"/>
<row C1="1"/>

  4 registro(s) seleccionado(s).
```

- El ejemplo siguiente muestra una consulta XMLRow y un fragmento de salida en el que el elemento <row> por omisión se sustituye por <entry>. Las columnas C1 y C2 se devuelven como los elementos <column1> y <column2> y el total de C1 y C2 se devuelve dentro de un elemento <total>:

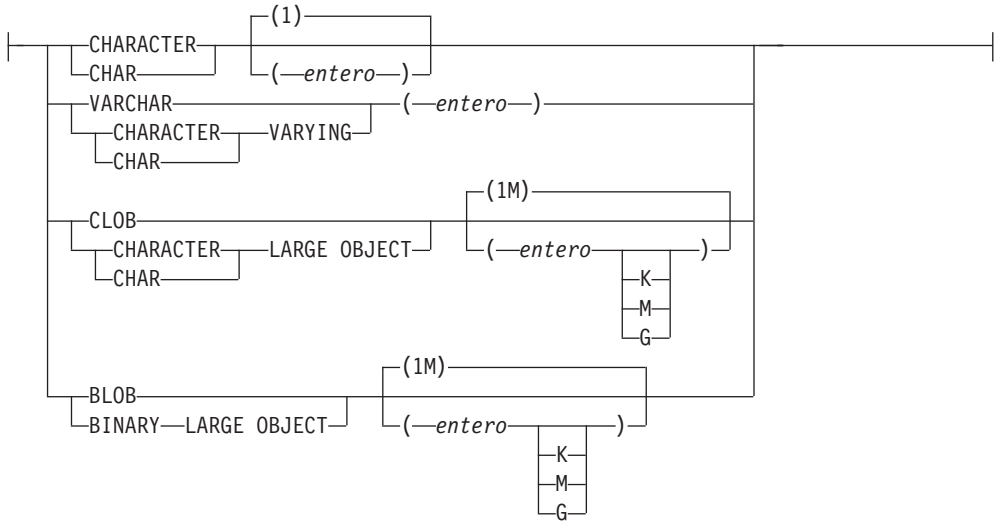
```
SELECT XMLROW(
   C1 AS "column1", C2 AS "column2",
   C1+C2 AS "total" OPTION ROW "entry")
FROM T1
```

```
<entry><column1>1</column1><column2>2</column2><total>3</total></entry>
<entry><column2>2</column2></entry>
<entry><column1>1</column1></entry>

  4 registro(s) seleccionado(s).
```

XMLSERIALIZE



tipo-datos:



Notas:

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

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLSERIALIZE devuelve un valor XML serializado de los tipos de datos especificados, generados a partir del argumento *expresión-XML*.

CONTENT
Indica que se puede especificar cualquier valor XML y el resultado de la serialización se basa en este valor de entrada.

expresión-XML
Especifica una expresión que devuelve un valor del tipo de datos XML. El valor de la secuencia XML no debe contener un elemento que sea un nodo de atributo (SQLSTATE 2200W). Ésta es la entrada al proceso de serialización.

AS *tipo-datos*
Especifica el tipo de resultado. El atributo de longitud implícito o explícito del tipo especificado de datos de resultados debe ser suficiente para contener la salida serializada (SQLSTATE 22001).

VERSION '1.0'
Especifica la versión XML del valor serializado. La única versión soportada es '1.0', que se debe especificar como constante de tipo serie (SQLSTATE 42815).

EXCLUDING XMLDECLARATION o INCLUDING XMLDECLARATION
Especifica si se incluye una declaración XML en el resultado. El valor por omisión es EXCLUDING XMLDECLARATION.

EXCLUDING XMLDECLARATION
Especifica que una declaración XML no está incluida en el resultado.

INCLUDING XMLDECLARATION
Especifica que una declaración XML está incluida en el resultado. La declaración XML es la serie '<?xml version="1.0" encoding="UTF-8"?>'.

El resultado tiene el tipo de datos especificado por el usuario. Una secuencia XML se convierte efectivamente para tener un único nodo de documento, aplicando XMLDOCUMENT a la *expresión-XML* antes de serializar los nodos XML resultantes. Si el resultado de la *expresión-XML* puede ser nulo, el resultado puede ser nulo; si el resultado de la *expresión-XML* es nulo, el resultado es el valor nulo.

Nota:

1. **Codificación en el resultado serializado**: El resultado serializado está codificado con UTF-8. Si se utiliza XMLSERIALIZE con un tipo de datos de caracteres, y se especifica la cláusula INCLUDING XMLDECLARATION, es posible que la serie de caracteres resultante que contiene XML serializado tenga una declaración de codificación XML que no coincida con la página de códigos de la serie de caracteres. Después de la serialización, que utiliza codificación UTF-8, la serie de caracteres que se devuelve del servidor al cliente se convierte en la página de códigos del cliente, y es posible que esa página de códigos sea diferente de UTF-8.

   Por consiguiente, las aplicaciones deben evitar la utilización directa de XMLSERIALIZE INCLUDING XMLDECLARATION que devuelven tipos de series caracteres y deben recuperar valores XML directamente en variables del lenguaje principal para mantener la coincidencia entre la página de códigos externa y la codificación de la declaración XML. Si debe utilizarse XMLSERIALIZE en esta situación, debe especificarse un tipo BLOB para evitar la conversión de la página de códigos.
2. **Sintaxis alternativa**: se puede especificar XMLCLOB(*expresión-XML*) en lugar de XMLSERIALIZE(*expresión-XML* AS CLOB(2G)). Recibe soporte solamente para la compatibilidad con versiones anteriores de DB2.

XMLTEXT

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLTEXT devuelve un valor XML con un único nodo de texto XQuery cuyo contenido es el argumento de entrada.

expresión-serie
Una expresión cuyo valor tiene un tipo de serie de caracteres: CHAR, VARCHAR o CLOB.

El tipo de datos del resultado es XML. Si el resultado de la *expresión-serie* puede ser nulo, el resultado puede ser nulo; si el valor de entrada es nulo, el resultado es el valor nulo. Si el resultado de la *expresión-serie* es una serie vacía, el valor del resultado es un nodo de texto vacío.

Ejemplos:

- Crear una consulta XMLTEXT simple.

```
VALUES (
  XMLTEXT(
    'El símbolo para las acciones de Johnson&Johnson es JNJ.'
  )
)
```

  Esta consulta genera el siguiente resultado serializado:

```
1
--------------------------------------------------
El símbolo para las acciones de Johnson&amp;Johnson es JNJ.
```

  Tenga en cuenta que el signo '&' se correlaciona con '&' cuando se serializa un nodo de texto.

- Utilice XMLTEXT con XMLAGG para construir contenido mixto. Supongamos que el contenido de la tabla T es como sigue:

```
seqno  plaintext                                                  emphtext
------ ---------------------------------------------------------- -------------
1      Esta consulta muestra cómo construir                       contenido mixto
2       utilizando XMLAGG y XMLTEXT. Sin                             XMLTEXT
3      XMLAGG no tendrá nodos de texto para agrupar con otros nodos, por lo tanto
       no puede generar contenido mixto
```

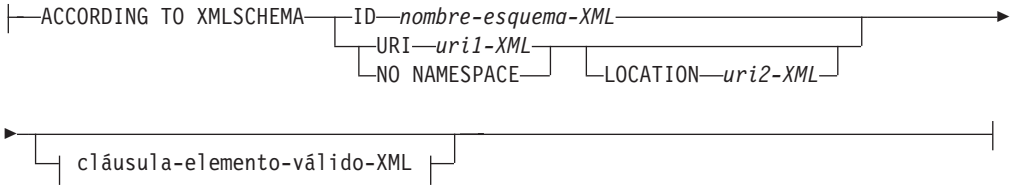
```
SELECT XMLELEMENT(
  NAME "para", XMLAGG(
    XMLCONCAT(
      XMLTEXT(
        PLAINTEXT
      ),
      XMLELEMENT(
        NAME "emphasis", EMPHTEXT
      )
    )
    ORDER BY SEQNO
  ), '.'
) AS "result"
FROM T
```

  Esta consulta genera el resultado siguiente:

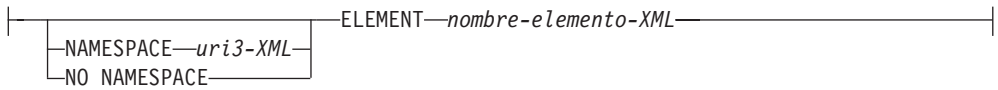
```
resultado
--------------------------------------------------------------------------------
<para>Esta consulta muestra cómo construir <emphasis>contenido mixto</emphasis>
utilizando XMLAGG y XMLTEXT. Sin <emphasis>XMLTEXT</emphasis> , XMLAGG
no tendrá nodos de texto para agrupar con otros nodos, por lo tanto, no puede
generar <emphasis>contenido mixto</emphasis>.</para>
```

XMLVALIDATE

validar-XML-según-cláusula:



cláusula-elemento-válido-XML:



El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLVALIDATE devuelve una copia del valor de entrada XML aumentado con la información obtenida a partir de la validación de esquema XML, incluidos los valores por omisión.

DOCUMENT
Especifica que el valor XML resultante de la *expresión-XML* debe ser un documento XML con el formato correcto conforme a XML Versión 1.0 (SQLSTATE 2200M).

expresión-XML
Una expresión que devuelve un valor del tipo de datos XML. Si la *expresión-XML* es una variable del lenguaje principal XML o un marcador de parámetro escrito implícita o explícitamente, la función ejecuta un análisis de validación que elimina espacios en blanco que se pueden ignorar y el valor CURRENT IMPLICIT XMLPARSE OPTION no se tiene en cuenta.

validar-XML-según-cláusula
Especifica la información que se usará al validar el valor de entrada XML.

ACCORDING TO XMLSCHEMA
Indica que la información del esquema XML para la validación se especifica de forma explícita. Si no se incluye esta cláusula, se debe proporcionar la información del esquema XML en el contenido del valor *expresión-XML*.

ID *nombre-esquema-XML*
Especifica un identificador de SQL para el esquema XML que se utilizará para la validación. El nombre (incluido el calificador de esquema de SQL implícito o explícito) debe designar de forma exclusiva un esquema XML existente en el depósito de esquema XML en el servidor actual. Si no existe un esquema XML con este nombre en el esquema de SQL especificado explícita o implícitamente, se devuelve un error (SQLSTATE 42704).

URI *uri1-XML*
Especifica el URI del espacio de nombres de destino del esquema XML que se utilizará para la validación. El valor de *XML-uri1* especifica un URI como constante de serie de caracteres que no está vacía. El URI debe ser el espacio de nombres de destino de un esquema XML registrado (SQLSTATE 4274A) y, si no se ha especificado una cláusula LOCATION, debe identificar exclusivamente el esquema XML registrado (SQLSTATE 4274B).

NO NAMESPACE
Especifica que el esquema XML para la validación no tenga espacio de nombres de destino. El URI del espacio de nombres de destino es equivalente a una serie de caracteres vacía que no se puede especificar como URI de espacio de nombres de destino explícito.

LOCATION *uri2-XML*
Especifica el URI de ubicación del esquema XML que se utilizará para la validación. El valor de *XML-uri2* especifica un URI como constante de serie de caracteres que no está vacía. El URI de ubicación del esquema XML, combinado con el URI del espacio de nombres de destino, debe identificar un esquema XML registrado (SQLSTATE 4274A), y sólo debe existir ese esquema XML registrado (SQLSTATE 4274B).

cláusula-elemento-válido-XML
Especifica que el valor XML en la *expresión-XML* debe tener el nombre del elemento especificado como elemento raíz del documento XML.

NAMESPACE *uri3-XML* **o NO NAMESPACE**
Especifica el espacio de nombres de destino para el elemento que se deba validar. Si no se especifica cláusula alguna, se presupone que el elemento especificado se encuentra en el mismo espacio de nombres que el espacio de nombres de destino del esquema XML registrado que se utilizará para la validación.

NAMESPACE *uri3-XML*
Especifica el URI del espacio de nombres para el elemento que se debe validar. El valor de *XML-uri3* especifica un URI como constante de serie de caracteres que no está vacía. Éste se puede utilizar cuando el esquema XML registrado que se utilizará para la validación tiene más de un espacio de nombres.

NO NAMESPACE
Especifica que el elemento para la validación no tiene espacio de nombres de destino. El URI del espacio de nombres de destino es equivalente a una serie de caracteres vacía que no se puede especificar como URI de espacio de nombres de destino explícito.

ELEMENT *nombre-elemento-xml*
Especifica el nombre de un elemento global en el esquema XML que se utilizará para la validación. El elemento especificado, con espacio de nombres implícito o explícito, debe coincidir con el elemento raíz del valor de *expresión-XML* (SQLSTATE 22535 o 22536).

El tipo de datos del resultado es XML. Si el valor de la *expresión-XML* puede ser nulo, el resultado puede ser nulo; si el valor de la *expresión-XML* es nulo, el resultado es el valor nulo.

El proceso de validación XML se lleva a cabo en un valor XML serializado. Debido a que XMLVALIDATE se invoca como argumento de tipo XML, este valor se serializa automáticamente antes del proceso de la validación con las dos excepciones siguientes.

- Si el argumento para XMLVALIDATE es una variable del lenguaje principal XML o un marcador de parámetro escrito implícita o explícitamente, se ejecuta una operación de análisis de validación en el valor de entrada (no se lleva a cabo ningún análisis implícito que no sea de validación y no se tiene en cuenta el valor CURRENT IMPLICIT XMLPARSE OPTION).
- Si el argumento para XMLVALIDATE es una invocación XMLPARSE que utiliza la opción PRESERVE WHITESPACE, el análisis XML y la validación XML del documento se pueden combinar en una única operación de análisis de validación.

Si un valor XML se ha validado anteriormente, el proceso de serialización elimina la información del tipo anotado de la validación anterior. Sin embargo, los valores por omisión y las expansiones de entidades de la validación anterior permanecen sin cambios. Si la validación es satisfactoria, todos los caracteres de espacio en blanco que se puedan ignorar se eliminarán del resultado.

Para validar un documento cuyo elemento raíz no tiene un espacio de nombres, debe existir un atributo xsi:noNamespaceSchemaLocation en el elemento raíz.

Nota:

1. **Determinación del esquema XML**: se puede especificar el esquema XML explícitamente como parte de la invocación XMLVALIDATE, o bien se puede determinar a partir de la información del esquema XML en el valor de entrada XML. Si no se especifica durante la invocación la información del esquema XML, el espacio de nombres de destino y la ubicación del esquema en el valor de entrada XML se utilizan para identificar el esquema registrado para la validación. Si no se especifica un esquema XML explícito, el valor de entrada XML debe contener una sugerencia de información del esquema XML (SQLSTATE 2200M). La información del esquema XML explícita o implícita debe identificar un esquema XML registrado (SQLSTATE 42704, 4274A o 22532), y sólo debe existir ese esquema XML registrado (SQLSTATE 4274B o 22533).
2. **Autorización del esquema XML**: el esquema XML que se utiliza para la validación se debe registrar en el depósito del esquema XML antes de utilizarlo. El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
   - Privilegio USAGE en el esquema XML que se debe utilizar durante la validación
   - Autorización DBADM

Ejemplos:

- Validar utilizando el esquema XML identificado por la sugerencia del esquema XML en el documento de instancia XML.

  ```
  INSERT INTO T1(XMLCOL)
    VALUES (XMLVALIDATE(?))
  ```

  Se presupone que el marcador de parámetro de entrada está vinculado a un valor XML que contiene la información del esquema XML.

```
<po:order
        xmlns:po='http://my.world.com'
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://my.world.com/world.xsd" >
...
</po:order>
```

Asimismo, se presupone que el esquema XML que está asociado con el espacio de nombres de destino ″http://my.world.com″ y por la sugerencia schemaLocation ″http://my.world.com/world.xsd″ se encuentra en el depósito del esquema XML.

Basándose en estas suposiciones, se validará el valor de entrada XML de acuerdo con ese esquema XML.

- Validar utilizando el esquema XML identificador por el nombre de SQL PODOCS.WORLDPO.

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID PODOCS.WORLDPO
    )
  )
```

Asumiendo que el esquema XML que esté asociado con el nombre de SQL FOO.WORLDPO se encuentre en el depósito XML, se validará el valor de entrada XML de acuerdo con ese esquema XML.

- Validar un elemento especificado del valor XML.

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID FOO.WORLDPO
      NAMESPACE 'http://my.world.com/Mary'
      ELEMENT "po"
    )
  )
```

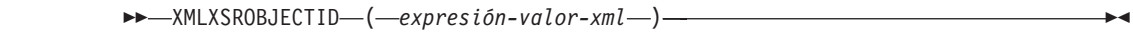
Asumiendo que el esquema XML que esté asociado con el nombre de SQL FOO.WORLDPO se encuentre en el depósito XML, se validará el esquema XML en relación al elemento ″po″, cuyo espacio de nombres es ’http://my.world.com/Mary’.

- El esquema XML se identifica por el espacio de nombres de destino y la ubicación del esquema.

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA URI 'http://my.world.com'
      LOCATION 'http://my.world.com/world.xsd'
    )
  )
```

Asumiendo que un esquema XML asociado con el espacio de nombres de destino ″http://my.world.com″ y por la sugerencia schemaLocation ″http://my.world.com/world.xsd″ se encuentre en el depósito del esquema XML, el valor de entrada XML se validará de acuerdo con ese esquema.

XMLXSROBJECTID



El esquema es SYSIBM.

La función XMLXSROBJECTID devuelve el identificador de objeto XSR del esquema XML utilizado para validar el documento XML especificado en el argumento. El identificador de objeto XSR se devuelve como un valor BIGINT y proporciona la clave de una fila única de SYSCAT.XSROBJECTS.

expresión-valor-xml
Especifica una expresión que da como resultado un valor con un tipo de datos XML. El valor XML obtenido debe ser una secuencia XML con un único elemento que es un documento XML o el valor nulo (SQLSTATE 42815). Si el argumento es nulo, la función devolverá nulo. Si *expresión-valor-xml* no especifica un documento XML validado, la función devolverá 0.

Nota:

1. Es posible que el esquema XML correspondiente a un ID de objeto XSR devuelto por la función ya no exista, ya que un esquema XML se puede eliminar sin que ello repercuta sobre los valores XML validados mediante dicho esquema. Por consiguiente, las consultas que utilicen el ID de objeto XSR para obtener más información del esquema XML a partir de las vistas de catálogo, pueden devolver un conjunto vacío.
2. Las aplicaciones pueden utilizar el identificador de objeto XSR para recuperar información adicional acerca del esquema XML. Por ejemplo, el identificador de objeto XSR puede utilizarse para devolver los documentos de esquema XML individuales que forman el esquema XML registrado de SYSCAT.SYSXSROBJECTCOMPONENTS y la jerarquía de documentos del esquema XML de SYSCAT.XSROBJECTHIERARCHIES.

Ejemplos:

- Recuperar el identificador de esquema XML para el documento XML XMLDOC almacenado en la tabla MYTABLE.

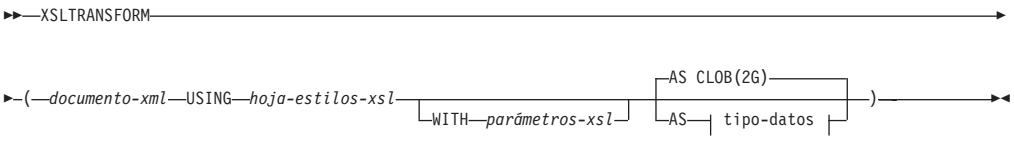
```
SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
```

- Recuperar los documentos del esquema XML asociados al documento XML que tiene un ID específico (en este caso donde DOCKEY=1) en la tabla MYTABLE, incluida la jerarquía de los documentos de esquema XML que forman el esquema XML.

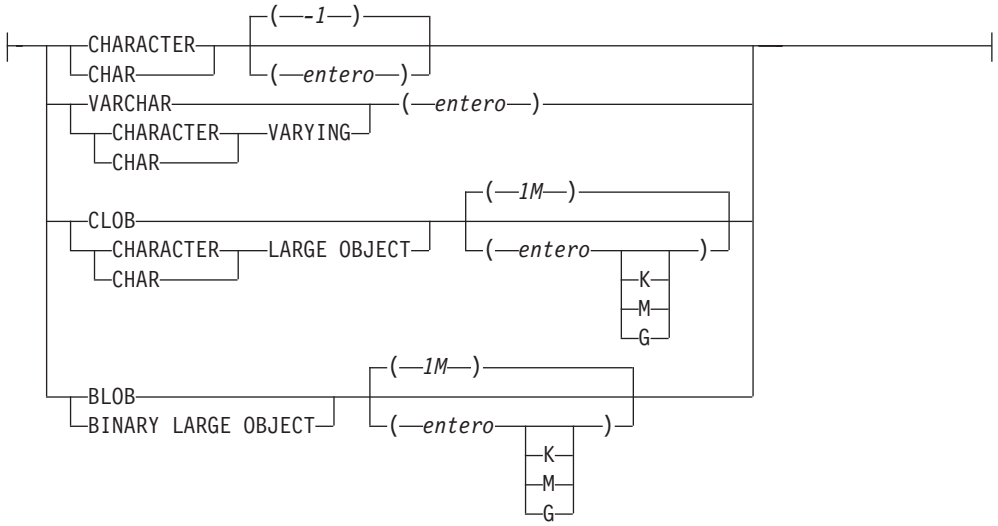
```
SELECT H.HTYPE, C.TARGETNAMESPACE, C.COMPONENT
FROM SYSCAT.XSROBJECTCOMPONENTS C, SYSCAT.XSROBJECTHIERARCHIES H
WHERE C.OBJECTID =
  (SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
   WHERE DOCKEY = 1)
AND C.OBJECTID = H.OBJECTID
```

XSLTRANSFORM

Utilice XSLTRANSFORM para convertir datos XML a otros formatos, incluyendo la conversión de documentos XML que se ajustan a un esquema XML en documentos que se ajustan a otro esquema.



tipo-datos:



El esquema es SYSIBM. Esta función no puede especificarse como nombre calificado.

La función XSLTRANSFORM transforma un documento XML en un formato de datos diferente. Los datos pueden transformarse a cualquier formato posible para el procesador XSLT, incluido XML, HTML o texto plano (pero sin limitarse a éstos).

Todas las vías de acceso que utiliza XSLTRANSFORM son internas para el sistema de base de datos. Este mandato no puede utilizarse directamente en la actualidad con archivos u hojas de estilos que residan en un sistema de archivos externos.

documento-xml
Expresión que devuelve un documento XML bien formado con un tipo de datos de XML, CHAR, VARCHAR, CLOB o BLOB. Este es el documento que se transforma utilizando la hoja de estilos XSL especificada en *hoja-estilos-xsl*.

Nota:

EL documento XML debe como mínimo tener una única raíz y estar bien formado.

hoja-estilos-xsl
Expresión que devuelve un documento XML bien formado con un tipo de datos de XML, CHAR, VARCHAR, CLOB o BLOB. El documento es una hoja de estilos XSL que se adecua a la Recomendación W3C XSLT Versión 1.0. No se

soportan las hojas de estilo que incorporan sentencias XQUERY o la declaración `xsl:include`. Esta hoja de estilos se aplica al objeto de transformar el valor especificado en *documento-xml*.

parámetros-xsl
Expresión que devuelve un nulo o un documento XML bien formado con un tipo de datos de XML, CHAR, VARCHAR, CLOB o BLOB. Este es un documento que proporciona valores de parámetros a la hoja de estilos XSL especificada en *hoja-estilos-xsl*. El valor del parámetro puede especificarse como atributo o como nodo de texto.

La sintaxis del documento del parámetro es la siguiente:

```
<params xmlns="http://www.ibm.com/XSLTransformParameters">
<param name="..." value="..."/>
<param name="...">enter value here</param>
  ...
</params>
```

Nota:

El documento de hoja de estilos debe tener `xsl:param` elemento(s) en el mismo con valores de atributo de nombres que coincidan con los especificados en el documento del parámetro.

AS *tipo-datos*
Especifica el tipo de datos del resultado. El atributo de longitud implícito o explícito del tipo especificado de datos de resultados debe ser suficiente para contener la salida transformada (SQLSTATE 22001). El tipo de datos de resultados por omisión es CLOB(2G).

Nota:

Si el argumento *documento-xml* o el argumento *hoja-estilos-xsl* es nulo, el resultado será nulo.

La conversión de página de códigos puede producirse al almacenar cualquiera de los documentos anteriores en una columna CHAR, VARCHAR o CLOB, lo cual podría dar como resultado una pérdida de caracteres.

Ejemplo

Este ejemplo ilustra el modo de utilizar XSLT como motor de formato. Para obtener la configuración, inserte en primer lugar los dos documentos de ejemplo que hay a continuación en la base de datos.

```
INSERT INTO XML_TAB VALUES
(1,
          '<?xml version="1.0"?>
<students xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "/home/steffen/xsd/xslt.xsd">
<student studentID="1" firstName="Steffen" lastName="Siegmund"
    age=â€23â€ university=â€Rostockâ€/>
</students>',
    '<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="headline"/>
<xsl:param name="showUniversity"/>
<xsl:template match="students">
                <html>
                   <head/>
```

```
                        <body>
                               <h1><xsl:value-of select="$headline"/></h1>
                               <table border="1">
                                       <th>
                                           <tr>
                                 <td width="80">IDestudiante</td>
                                 <td width="200">Nombre</td>
                                 <td width="200">Apellido</td>
                                 <td width="50">Edad</td>
                                 <xsl:choose>
       <xsl:when test="$showUniversity ='true'">
                                                <td width="200">Universidad</td>
                                     </xsl:when>
       </xsl:choose>
                                </tr>
                                   </th>
                                   <xsl:apply-templates/>
                                 </table>
                         </body>
                  </html>
                 </xsl:template>
                      <xsl:template match="student">
                      <tr>
                    <td><xsl:value-of select="@studentID"/></td>
                    <td><xsl:value-of select="@firstName"/></td>
                    <td><xsl:value-of select="@lastName"/></td>
                    <td><xsl:value-of select="@age"/></td>
                    <xsl:choose>
                        <xsl:when test="$showUniversity = 'true' ">
                          <td><xsl:value-of select="@university"/></td>
                        </xsl:when>
                    </xsl:choose>
                           </tr>
                      </xsl:template>
</xsl:stylesheet>'
);
```

A continuación, llame la función XSLTRANSFORM para convertir los datos XML a HTML y visualícelos.

```
SELECT XSLTRANSFORM (XML_DOC USING XSL_DOC AS CLOB(1M)) FROM XML_TAB;
```

El resultado es este documento:

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1></h1>
<table border="1">
<th>
<tr>
<td width="80">IDestudiante</td>
<td width="200">Nombre</td>
<td width="200">Apellido</td>
<td width="50">Edad</td>
</tr>
</th>
 <tr>
<td>1</td>
<td>Steffen</td><td>Siegmund</td>
<td>23</td>
</tr>
             </table>
</body>
</html>
```

En este ejemplo, la salida es HTML y los parámetros sólo influyen el HTML que se produce y los datos convertidos en el mismo. Como tal, ilustra la utilización de XSLT como motor de formato para la salida de usuario final.

Nota sobre uso:

Esta función **no** se destina a aplicaciones de alto rendimiento y **no puede** sustituir a funciones similares del servidor de aplicaciones.

YEAR

El esquema es SYSIBM.

La función YEAR devuelve la parte correspondiente al año de un valor.

El argumento debe ser un valor DATE, TIMESTAMP, una duración de fecha, una duración de indicación de fecha y hora o una representación de serie de caracteres válida de una fecha o indicación de fecha y hora que no sea un CLOB. En una base de datos Unicode, si un argumento proporcionado es una serie gráfica, se convertirá a una serie de caracteres antes de que se ejecute la función.

El resultado de la función es un entero grande. Si el argumento puede ser nulo, el resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

Las demás normas dependen del tipo de datos del argumento especificado:

- Si el argumento es un valor DATE, TIMESTAMP o una representación de serie válida de una fecha o indicación de fecha y hora:
  - El resultado es la parte correspondiente al año del valor, que es un entero entre 1 y 9999.
- Si el argumento es una duración de fecha o duración de la indicación de fecha y hora:
  - El resultado es la parte correspondiente al año del valor, que es un entero entre -9999 y 9999. El resultado que no es cero tiene el mismo signo que el argumento.

Ejemplos

- Seleccione todos los proyectos de la tabla PROJECT que se han planificado para empezar (PRSTDATE) y finalizar (PRENDATE) en el mismo año.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

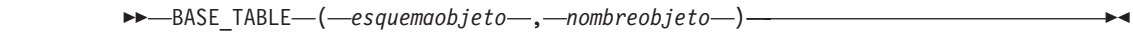
- Seleccione todos los proyectos de la tabla PROJECT que se haya planificado que finalizasen en menos de un año.

```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

Funciones de tabla

Sólo se puede utilizar una función de tabla en la cláusula FROM de una sentencia. Las funciones de tabla devuelven columnas de una tabla, de aspecto parecido a una tabla creada mediante una sentencia CREATE TABLE simple. Las funciones de tabla pueden calificarse con un nombre de esquema.

BASE_TABLE



El esquema es SYSPROC.

La función BASE_TABLE devuelve el nombre de objeto y el nombre de esquema del objeto encontrado después de que se haya resuelto cualquier cadena de alias. El ″nombreobjeto″, y el ″nombreesquema″, especificados se utilizan como punto de inicio de la resolución. Si el punto de inicio no hace referencia a un alias, se devuelven el nombre de esquema y el nombre no calificado del punto de inicio. La función devuelve una única tabla de filas que consta de las columnas siguientes:

Tabla 63. Información devuelta por la función BASE_TABLE

Nombre de columna	Tipo de datos	Descripción
BASESCHEMA	VARCHAR(128)	Nombre de esquema del objeto encontrado después de resolver cualquier cadena de alias. Coincide con ″esquemaobjeto″ si no se encuentra ningún alias coincidente.
BASENAME	VARCHAR(128)	Nombre no calificado del objeto encontrado después de resolver cualquier cadena de alias. Coincide con ″nombreobjeto″ si no se encuentra ningún alias coincidente. El nombre puede identificar una tabla, una vista o un objeto no definido.

esquemaobjeto
Una expresión de caracteres que representa el esquema utilizado para calificar el valor del *nombreobjeto* suministrado antes de la resolución. El tipo de datos de *esquemaobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

nombreobjeto
Una expresión de caracteres que representa el nombre no calificado que debe resolverse. El tipo de datos de *nombreobjeto* debe ser CHAR o VARCHAR y su longitud ser mayor que 0 y menor que 129 bytes.

Nota: La función de tabla BASE_TABLE mejora el rendimiento en configuraciones de bases de datos particionadas al evitar la comunicación innecesaria que se produce entre la partición coordinadora y la partición de catálogo cuando se utilizan las funciones escalares TABLE_SCHEMA y TABLE_NAME.

Ejemplo

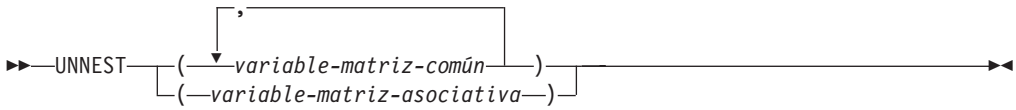
La siguiente sentencia utilizando las funciones TABLE_SCHEMA y TABLE_NAME se escribe del modo siguiente:

```
SELECT COLCOUNT INTO :H00030
FROM SYSCAT.TABLES
WHERE OWNER = TABLE_SCHEMA(:H00031 ,:H00032 )
AND TABNAME = TABLE_NAME(:H00031 ,:H00032 )
```

La sentencia equivalente utilizando la función BASE_TABLE se puede escribir del modo siguiente:

```
SELECT COLCOUNT INTO :H00030
FROM SYSCAT.TABLES A, TABLE(SYSPROC.BASE_TABLE(:H00032, :H00031)) AS B
WHERE A.OWNER = B.BASESCHEMA
AND A.TABNAME = B.BASENAME
```

UNNEST



El esquema es SYSIBM.

La función UNNEST devuelve una tabla de resultados que incluye una fila para cada elemento de la matriz especificada. Si se han especificado varios argumentos de matriz común, el número de filas coincidirá con la matriz que tenga la cardinalidad más alta.

variable-matriz-común
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz común, o especificación CAST de un marcador de parámetro a un tipo de matriz común.

variable-matriz-asociativa
Variable de SQL, parámetro de SQL o variable global de un tipo de matriz asociativa, o especificación CAST de un marcador de parámetro a un tipo de matriz asociativa.

Los nombres de las columnas de resultado generados por la función UNNEST se pueden proporcionar como parte de la *cláusula-correlación* necesaria de la cláusula *tabla-derivada-colección*.

La función UNNEST sólo puede utilizarse en una cláusula *tabla-derivada-colección* en un contexto en el que las matrices estén soportadas (SQLSTATE 42887).

La tabla de resultados depende de los argumentos de entrada.

- Si se especifica un único argumento de matriz común:
  - Si el elemento de la matriz no es del tipo de datos de fila, el resultado es una tabla de una sola columna cuyo tipo de datos de columna coincide con el tipo de datos del elemento de la matriz.
  - Si el elemento de la matriz sí es del tipo de datos de fila, el resultado es una tabla con una columna para cada campo de fila del elemento de la matriz. Los tipos de datos de columna de la tabla de resultados coinciden con los correspondientes tipos de datos de campo de fila del elemento de la matriz.
- Si se especifica más de un argumento de matriz común y ningún elemento de la matriz tiene un tipo de datos de fila, la primera matriz proporciona la primera columna de la tabla de resultados, la segunda matriz proporciona la segunda columna y así sucesivamente. El tipo de datos de cada columna coincide con el tipo de datos de los elementos de matriz del argumento de matriz correspondiente. Si las cardinalidades de las matrices no son idénticas, la cardinalidad de la tabla resultante es la misma que la matriz con la mayor cardinalidad. Los valores de columna de la tabla se establecen en el valor nulo para todas las filas cuyo valor de índice de matriz sea mayor que la cardinalidad de la matriz correspondiente. En otras palabras, si se visualiza cada matriz como una tabla con dos columnas, una para los índices de matriz y otra para los datos, UNNEST realiza una operación OUTER JOIN entre las matrices, utilizando la igualdad en los índices de matriz como predicado de unión.
- Si se especifica un único argumento de matriz asociativa:

  - Si el elemento de la matriz no es del tipo de datos de fila, el resultado es una tabla de dos columnas donde el tipo de datos de la primera columna coincide con el tipo de datos del índice de la matriz y el tipo de datos de la segunda columna coincide con el tipo de datos del elemento de la matriz.
  - Si el elemento de la matriz es del tipo de datos de fila, el resultado es una tabla con una columna más que el número de campos del tipo de datos de fila, donde el tipo de datos de la primera columna coincide con el tipo de datos del índice de la matriz y el tipo de datos de las columnas restantes coincide con los tipos de datos de fila de los elementos de la matriz.
- Se devuelve un error (SQLSTATE 42884):
  - Si se especifica más de un argumento de matriz asociativa.
  - Si se especifica más de un argumento de matriz y como mínimo una de las matrices tiene un tipo de datos de elemento que es de tipo fila.
  - Si se especifican argumentos de matriz común y asociativa.

Los nombres de las columnas de resultado generados por la función UNNEST se pueden proporcionar como parte de la *cláusula-correlación* necesaria de la *tabla-derivada-colección*.

Esta función de tabla especial solamente se utiliza en la *tabla-derivada-colección* de *referencia-tabla* en una cláusula FROM.

Si se proporciona más de una matriz y al menos uno de los argumentos es una matriz asociativa, se devuelve un error (SQLSTATE 42884).

Si se utiliza la cláusula WITH ORDINALITY al desanidar una matriz asociativa, se devuelve un error (SQLSTATE 428HT).

Ejemplos

- Supongamos que la variable de matriz común RECENT_CALLS del tipo de matriz PHONENUMBERS contiene solamente los tres valores de elemento 9055553907, 4165554213 y 4085553678. La consulta siguiente:

```
SELECT T.ID, T.NUM
FROM UNNEST(RECENT_CALLS) WITH ORDINALITY AS T(NUM, ID)
```

  devuelve una tabla formateada así:

```
ID    NUM
----------
1     9055553907
2     4165554213
3     4085553678
```

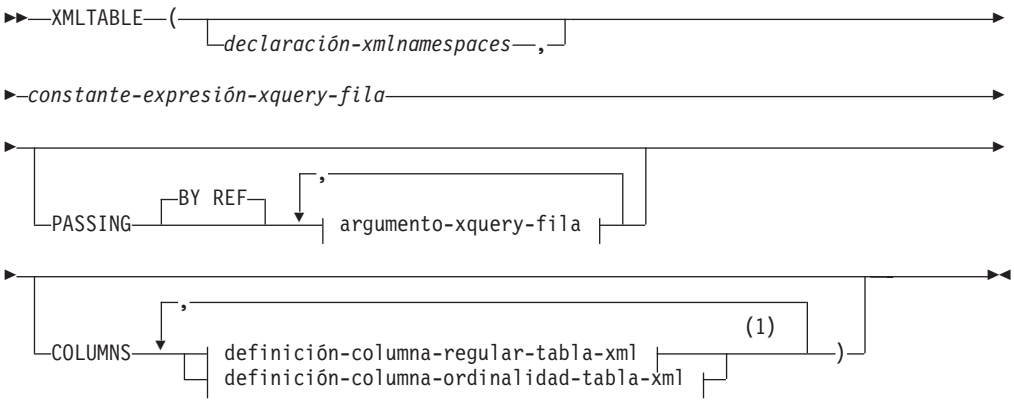
- Devolver la lista de números de teléfono particulares de la variable de matriz PHONELIST del tipo de matriz PERSONAL_PHONENUMBERS junto con los valores de serie de caracteres de índice. La consulta siguiente:

```
SELECT T.ID, T.PHONE
FROM UNNEST(PHONELIST) AS T(ID, PHONE)
```

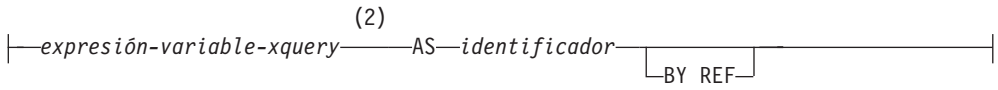
  devuelve una tabla formateada así:

```
ID    PHONE
---------------
Home  4163053745
Work  4163053746
Mom   4164789683
```

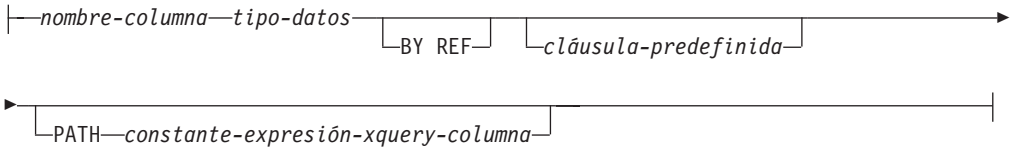
XMLTABLE



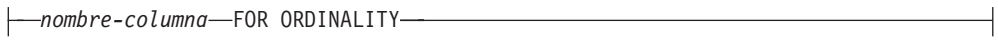
argumento-xquery-fila:



definición-columna-normal-tabla-xml:



definición-columna-ordinalidad-tabla-xml:



Notas:

1 La cláusula definición-columna-ordinalidad-tabla-xml no debe especificarse más de una vez (SQLSTATE 42614)).

2 El tipo de datos de la expresión no puede ser DECFLOAT.

El esquema es SYSIBM. El nombre de la función no puede especificarse como un nombre calificado.

La función XMLTABLE devuelve una tabla de resultados a partir de la evaluación de las expresiones de XQuery, posiblemente utilizando los argumentos de entrada especificados como variables XQuery. Cada elemento de la secuencia de resultados de la expresión XQuery de fila representa una fila de la tabla de resultados.

declaración-xmlnamespaces
Especifica una o más declaraciones de espacios de nombre XML que se convierten en parte del contexto estático de la *constante-expresión-xquery-fila* y la *constante-expresión-xquery-columna*. El conjunto de los espacios de nombres estáticamente conocidos para las expresiones XQuery que son argumentos de XMLTABLE es la combinación del conjunto establecido previamente de

espacios de nombres estáticamente conocidos y las declaraciones de espacios de nombres especificados en esta cláusula. El prólogo XQuery dentro de una expresión XQuery puede alterar temporalmente estos espacios de nombres.

Si no se ha especificado la *declaración-xmlnamespaces*, sólo el conjunto establecido previamente de espacios de nombres estáticamente conocidos se aplica a las expresiones XQuery.

constante-expresión-xquery-fila
Especifica una constante de serie de caracteres de SQL que se interpreta como una expresión XQuery mediante la sintaxis del lenguaje XQuery soportada. La serie de la constante se convierte directamente a UTF-8 sin la conversión a la base de datos o la página de códigos de la sección. La expresión XQuery se ejecuta utilizando un conjunto opcional de valores de entrada XML, y devuelve una secuencia XQuery de salida donde se genera una fila para cada elemento de la secuencia. El valor de *constante-expresión-xquery-fila* no debe ser una serie vacía o una serie de espacios en blanco (SQLSTATE 10505).

PASSING
Especifica valores de entrada y la manera en que estos valores se pasan a la expresión XQuery especificada por la *constante-expresión-xquery-fila*. Por omisión, cada nombre de columna exclusivo en el ámbito en el que se invoca la función se pasa implícitamente a la expresión XQuery utilizando el nombre de la columna como nombre de variable. Si un *identificador* de un argumento-xquery-fila especificado coincide con el nombre de columna con ámbito, entonces el argumento-xquery-fila se pasa a la expresión XQuery alterando temporalmente dicha columna implícita.

BY REF
Especifica que cualquier argumento de entrada XML pasan por referencia por omisión. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML.

Esta cláusula no afectará al modo en que se pasan los valores que no son XML. Los valores que no son XML crean una copia nueva del valor durante la conversión a XML.

argumento-xquery-fila
Especifica un argumento que se debe pasar a la expresión XQuery especificada por la *constante-expresión-xquery-fila*. Un argumento especifica un valor y la forma en que ese valor se debe pasar. El argumento incluye una expresión SQL que se evalúa antes de pasar el resultado a la expresión XQuery.

- Si el valor del resultado es del tipo XML, pasa a ser un *valor-xml-entrada*. Un valor XML nulo se convierte en una secuencia XML vacía.
- Si el valor del resultado no es del tipo XML, debe ser convertible al tipo de datos XML. Un valor nulo se convierte en una secuencia XML vacía. El valor convertido se transforma en un *valor-xml-entrada*.

Cuando se evalúa la *constante-expresión-xquery-fila*, se presenta una variable XQuery con un valor equivalente al *valor-xml-entrada* y un nombre especificado por la cláusula AS.

expresión-variable-xquery
Especifica una expresión SQL cuyo valor está disponible para la expresión XQuery especificada por la *constante-expresión-xquery-fila* durante la ejecución. La expresión no puede contener una expresión NEXT VALUE, PREVIOUS VALUE (SQLSTATE 428F9) ni una función OLAP (SQLSTATE 42903). El tipo de datos de la expresión no puede ser DECFLOAT.

AS *identificador*
Especifica que el valor generado por la *expresión-variable-xquery* pasará a la *constante-expresión-xquery-fila* como variable XQuery. El nombre de la variable será *identificador*. El signo de dólar inicial ($) que precede a los nombres de variable en el lenguaje XQuery no se incluye en el *identificador*. El identificador debe ser un nombre válido de variable XQuery y está restringido a un nombre XML NCName. El identificador no debe tener más de 128 bytes de longitud. Dos argumentos de la misma cláusula PASSING no pueden emplear el mismo identificador (SQLSTATE 42711).

BY REF
Indica que un valor de entrada XML se debe pasar por referencia. Cuando los valores XML se pasan por referencia, la evaluación de XQuery utiliza los árboles de nodos de entrada, si los hay, directamente desde las expresiones de entrada especificadas, con lo que se conservan todas las propiedades, incluyendo las identidades de nodo originales y el orden del documento. Si dos argumentos pasan el mismo valor XML, las comparaciones de identidad de nodo y orden de documento en que intervienen algunos nodos incluidos entre los dos argumentos de entrada pueden hacer referencia a nodos del mismo árbol de nodos XML. Si no se especifica BY REF a continuación de una *variable-expresión-xquery*, los argumentos XML se pasan mediante el mecanismo de pase por omisión que se ofrece mediante la sintaxis situada tras la palabra clave PASSING. Esta opción no se puede especificar para valores que no sean XML (SQLSTATE 42636). Cuando se pasa un valor que no es XML, el valor se convierte a XML; este proceso crea una copia.

COLUMNS
Especifica las columnas de salida de la tabla de resultados.. Si no se especifica esta cláusula, se devuelve por referencia una columna sin nombre exclusivo de tipo de datos XML, con el valor basado en el elemento de secuencia de la evaluación de la expresión XQuery en la *constante-expresión-xquery-fila* (equivalente a especificar PATH '.'). Para hacer referencia a la columna de resultados, se debe especificar un *nombre-columna* en la *cláusula-correlación* que sigue a la función.

definición-columna-regular-tabla-xml
Especifica las columnas de salida de la tabla de resultados, incluido el nombre de la columna, el tipo de datos, el mecanismo de pase XML y una expresión XQuery para extraer el valor del elemento de la secuencia para la fila

nombre-columna
Especifica el nombre de la columna en la tabla de resultados. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

tipo-datos
Especifica el tipo de datos de la columna. Consulte CREATE TABLE

para la sintaxis y una descripción de los tipos disponibles. Se puede utilizar un *tipo-datos* en XMLTable si hay un XMLCAST soportado del tipo de datos XML al *tipo-datos* especificado.

BY REF
Especifica que los valores XML se devuelven por referencia para columnas del tipo de datos XML. Por omisión, BY REF devuelve los valores XML. Cuando los valores XML se devuelven por referencia, el valor XML incluye los árboles de nodos de entrada, si los hay, directamente desde los valores de resultados, y conserva todas las propiedades, incluidas las identidades de nodo originales y el orden del documento. Esta opción no se puede especificar para las columnas que no sean XML (SQLSTATE 42636). Cuando se procesa una columna que no es XML, el valor se convierte desde XML; este proceso crea una copia.

cláusula-predefinida
Especifica un valor por omisión para la columna. Consulte CREATE TABLE para la sintaxis y una descripción de la *cláusula-predefinida*. Para las columnas resultado XMLTABLE, se aplica el valor por omisión cuando el proceso de la expresión XQuery incluida en la *constante-expresión-xquery-columna* devuelve una secuencia vacía.

PATH *constante-expresión-xquery-columna*
Especifica una constante de serie de caracteres de SQL que se interpreta como una expresión XQuery mediante la sintaxis del lenguaje XQuery soportada. La serie de la constante se convierte directamente a UTF-8 sin la conversión a la base de datos o la página de códigos de la sección. La *constante-expresión-xquery-columna* especifica una expresión XQuery que determina el valor de la columna en relación a un elemento que es el resultado de evaluar la expresión XQuery en la *constante-expresión-xquery-fila*. Dado un elemento del resultado de procesar la *constante-expresión-xquery-fila* como elemento de contexto proporcionado exteriormente, se evalúa la *constante-expresión-xquery-columna*, y se devuelve una secuencia de salida. El valor de la columna está determinado por esta secuencia de salida, como sigue.

- Si la secuencia de salida no contiene elementos, la *cláusula-predefinida* proporciona el valor de la columna.
- Si se devuelve una secuencia vacía y no se especifica ninguna *cláusula-predefinida*, se asignará un valor nulo a la columna.
- Si se devuelve una secuencia que no está vacía, el valor es XMLCAST para el *tipo-datos* especificado para la columna. El proceso de este XMLCAST podría devolver un error.

El valor de *constante-expresión-xquery-columna* no debe ser una serie vacía o una serie de espacios en blanco (SQLSTATE 10505). Si no se especifica esta cláusula, la expresión XQuery por omisión es simplemente *nombre-columna*.

definición-columna-ordinalidad-tabla-xml
Especifica la columna de ordinalidad de la tabla de resultados.

nombre-columna
Especifica el nombre de la columna en la tabla de resultados. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

FOR ORDINALITY
Especifica que *nombre-columna* es la columna de ordinalidad de la tabla de resultados. El tipo de datos de esta columna es BIGINT. El valor de esta columna en la tabla de resultados es el número secuencial del elemento para la fila en la secuencia resultante de evaluar la expresión XQuery en *constante-expresión-xquery-fila*.

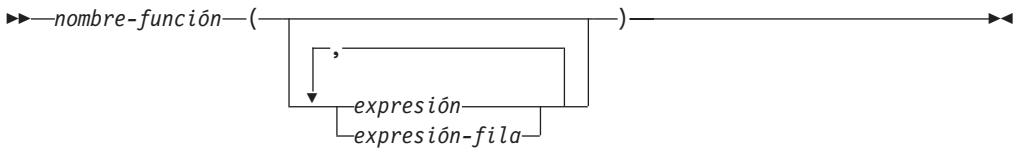
Si la evaluación de cualquiera de las expresiones XQuery produce un error, la función XMLTABLE devuelve un error de XQuery (clase de SQLSTATE '10').

Ejemplos:

- Listar como un resultado de tabla los elementos de la orden de compra para las órdenes con un estado de 'NEW'.

```
SELECT U."PO ID", U."Part #", U."Product Name",
    U."Quantity", U."Price", U."Order Date"
  FROM PURCHASEORDER P,
    XMLTABLE(XMLNAMESPACES('http://podemo.org' AS "pod"),
             '$po/PurchaseOrder/itemlist/item' PASSING P.PORDER AS "po"
             COLUMNS "PO ID"         INTEGER       PATH '../../@POid',
                     "Part #"        CHAR(6)       PATH 'product/@pid',
                     "Product Name"  CHAR(50)      PATH 'product/pod:name',
                     "Quantity"      INTEGER       PATH 'quantity',
                     "Price"         DECIMAL(9,2)  PATH 'product/pod:price',
                     "Order Date"    TIMESTAMP     PATH '../../dateTime'
             ) AS U
 WHERE P.STATUS = 'NEW'
```

Funciones definidas por el usuario



Las *funciones definidas por el usuario (UDF)* son extensiones o adiciones a las funciones incorporadas existentes del lenguaje SQL. Una función definida por el usuario puede ser una función escalar, que devuelve un solo valor cada vez que se invoca; una función agregada, a la que se pasa un conjunto de valores similares y devuelve un solo valor para el conjunto; una función de fila, que devuelve una fila o una función de tabla, que devuelve una tabla.

En los esquemas SYSFUN y SYSPROC se proporcionan varias funciones definidas por el usuario.

Una función definida por el usuario (UDF) sólo puede ser una función agregada si su fuente es una función agregada existente. Se hace referencia a una UDF mediante un nombre de función calificado o no calificado, seguido de paréntesis que encierran los argumentos de la función (si los hay). Una función de columna escalar definida por el usuario registrada con la base de datos puede aludirse en los mismos contextos en que pueda aparecer cualquier función incorporada. Una función de fila definida por el usuario sólo puede aludirse implícitamente cuando está registrada como función de transformación para un tipo definido por el usuario. Una función de tabla definida por el usuario registrada en la base de datos sólo puede aludirse en la cláusula FROM de una sentencia SELECT.

Los argumentos de la función deben corresponderse en número y posición con los parámetros especificados para la función definida por el usuario cuando se registró con la base de datos. Además, los argumentos deben ser de tipos de datos que sean promocionables a los tipos de datos de los parámetros definidos correspondientes.

El resultado de la función es el especificado en la cláusula RETURNS. La cláusula RETURNS, definida cuando se registró la UDF, determina si una función es una función de tabla o no lo es. Si se especifica (o se toma como valor por omisión) la cláusula RETURNS NULL ON NULL INPUT al registrar la función, el resultado es nulo si algún argumento es nulo. En el caso de las funciones de tabla, esto se interpreta como una tabla de retorno sin filas (es decir, una tabla vacía).

Consulte la sección sobre las expresiones de fila para obtener más información sobre los tipos de datos de fila y las normas.

A continuación se muestran algunos ejemplos de funciones definidas por el usuario:

- Una UDF escalar denominada ADDRESS extrae la dirección de inicio de los resúmenes almacenados en formato script. La función ADDRESS espera un argumento CLOB y devuelve VARCHAR(4000):

```
SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
  WHERE RESUME_FORMAT = 'SCRIPT'
```

- La tabla T2 tiene una columna numérica A. Al invocar la UDF escalar denominada ADDRESS del ejemplo anterior:

```
SELECT ADDRESS(A) FROM T2
```

  se genera un error (SQLSTATE 42884), ya que no existe ninguna función con un nombre que coincida y con un parámetro promocionable del argumento.

- Una UDF de tabla denominada WHO devuelve información acerca de las sesiones de la máquina servidora que estaban activas en el momento de ejecutar la sentencia. La función WHO se invoca desde una cláusula FROM que incluye la palabra clave TABLE y una variable de correlación obligatoria. Los nombres de columna de la tabla WHO() se han definido en la sentencia CREATE FUNCTION.

```
SELECT ID, START_DATE, ORIG_MACHINE
  FROM TABLE( WHO() ) AS QQ
  WHERE START_DATE LIKE 'MAY%'
```

Capítulo 4. Procedimientos

Visión general de los procedimientos

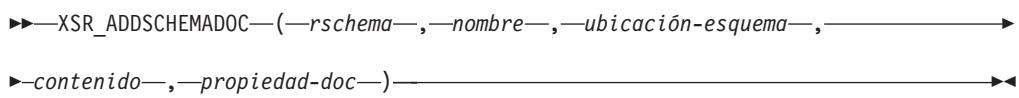
Un procedimiento es un programa de aplicación que se puede iniciar mediante la sentencia CALL de SQL. El procedimiento se especifica mediante un nombre de procedimiento, que puede ir seguido por argumentos, incluidos entre paréntesis.

El argumento o los argumentos de un procedimiento son valores escalares individuales, que pueden ser de tipos diferentes y pueden tener significados diferentes. Los argumentos pueden utilizarse para pasar valores en el procedimiento y recibir y/o devolver valores del procedimiento.

Los procedimientos definidos por el usuario son procedimientos que se registran en una base de datos en SYSCAT.ROUTINES, utilizando la sentencia CREATE PROCEDURE. Con el gestor de bases de datos se proporciona un conjunto de funciones de este tipo, en un esquema llamado SYSFUN y otro en un esquema llamado SYSPROC.

Los procedimientos pueden calificarse con el nombre de esquema.

XSR_ADDSCHEMADOC



El esquema es SYSPROC.

Cada esquema XML del depósito de esquemas XML (XSR) puede constar de uno o más documentos de esquema XML. Cuando un esquema XML consta de múltiples documentos, se utiliza el procedimiento almacenado XSR_ADDSCHEMADOC para añadir cada esquema XML en lugar del documento de esquema XML primario.

Autorización

El ID de autorización de quien realiza la llamada del procedimiento debe ser el propietario del objeto de XSR, como aparece en la vista de catálogo SYSCAT.XSROBJECTS.

rschema
Argumento de entrada de tipo VARCHAR (128) que especifica el esquema de SQL para el esquema XML. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este esquema XML en el XSR, el cual debe moverse al estado completo. (La otra parte del identificador de SQL viene suministrada por el argumento de nombre.) Este argumento puede tener un valor NULL, que indica que el esquema de SQL por omisión se utiliza, tal como se define en el registro especial CURRENT SCHEMA. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento. Los objetos de XSR no experimentarán colisiones de nombres con objetos de base de datos que existan

fuera del XSR, ya que los objetos de XSR aparecen en un espacio de nombres diferente de los objetos que están fuera del depósito de esquemas XML.

nombre
Argumento de entrada de tipo VARCHAR (128) que especifica el nombre del esquema XML. El identificador de SQL completo para el esquema XML es *rschema.name*. El esquema XML ya debe existir como resultado de llamar el procedimiento almacenado XSR_REGISTER y el registro del esquema XML no puede se puede completar todavía. Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

ubicación-esquema
Argumento de entrada de tipo VARCHAR (1000), que puede tener un valor NULL, que indica la ubicación de esquema del documento de esquema XML primario al que se añade el documento de esquema XML. Este argumento es el nombre externo del esquema XML, es decir, el documento primario se puede identificar con el atributo xsi:schemaLocation en los documentos de instancia XML.

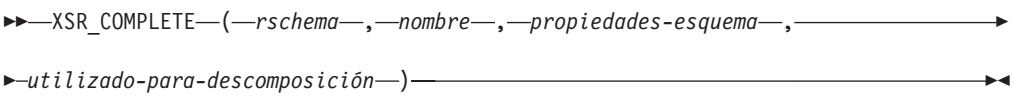
contenido
Parámetro de entrada de tipo BLOB (30M) que contiene el contenido del documento de esquema XML que se añade. Este argumento no puede tener un valor NULL; se debe proporcionar un documento de esquema XML.

propiedad-doc
Parámetro de entrada de tipo BLOB (5M) que indica las propiedades para el documento de esquema XML que se añade. Este parámetro puede tener un valor NULL; en caso contrario, el valor es un documento XML.

Ejemplo:

```
CALL SYSPROC.XSR_ADDSCHEMADOC(
  'user1',
  'POschema',
  'http://myPOschema/address.xsd',
  :content_host_var,
  0)
```

XSR_COMPLETE



El esquema es SYSPROC.

El procedimiento XSR_COMPLETE es el último procedimiento almacenado al que se llama como parte del proceso de registro del esquema XML, que registra esquemas XML con el depósito de esquemas XML (XSR). Un esquema XML no está disponible para ser validado hasta que se completa el registro del esquema mediante una llamada a este procedimiento almacenado.

Autorización:

El ID de autorización de quien realiza la llamada del procedimiento debe ser el propietario del objeto de XSR, como aparece en la vista de catálogo SYSCAT.XSROBJECTS.

rschema
Argumento de entrada de tipo VARCHAR (128) que especifica el esquema de SQL para el esquema XML. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este esquema XML en el XSR, el cual debe moverse al estado completo. (La otra parte del identificador de SQL viene suministrada por el argumento de nombre.) Este argumento puede tener un valor NULL, que indica que el esquema de SQL por omisión se utiliza, tal como se define en el registro especial CURRENT SCHEMA. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento. Los objetos de XSR no experimentarán colisiones de nombres con objetos de base de datos que existan fuera del XSR, ya que los objetos de XSR aparecen en un espacio de nombres diferente de los objetos que están fuera del depósito de esquemas XML.

nombre
Argumento de entrada de tipo VARCHAR (128) que especifica el nombre del esquema XML. El identificador de SQL completo para el esquema XML, para el que se debe realizar una comprobación de finalización, es *rschema.name*. El esquema XML ya debe existir como resultado de llamar el procedimiento almacenado XSR_REGISTER y el registro del esquema XML no puede se puede completar todavía. Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

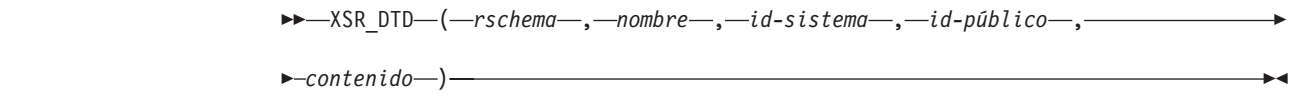
propiedades-esquema
Argumento de entrada de tipo BLOB (5M) que especifica las propiedades, si las hay, asociadas al esquema XML. El valor para este argumento puede ser NULL, si no hay propiedades asociadas, o bien un documento XML que represente las propiedades para el esquema XML.

utilizado-para-descomposición
Parámetro de entrada de tipo entero que indica si un esquema XML se va a utilizar para la descomposición. Si un esquema XML se va a utilizar para descomposición, este valor debe establecerse en 1; de lo contrario, debe establecerse en cero.

Ejemplo:

```
CALL SYSPROC.XSR_COMPLETE(
  'user1',
  'POschema',
  :schemaproperty_host_var,
  0)
```

XSR_DTD



El esquema es SYSPROC.

El procedimiento XSR_DTD registra una declaración de tipo de documento (DTD) con el depósito de esquemas XML (XSR).

Autorización

El ID de autorización de quien realiza la llamada del procedimiento debe tener como mínimo uno de los siguientes:

- Autorización DBADM.
- Autorización de base de datos IMPLICIT_SCHEMA si no existe el esquema de SQL.
- Privilegio CREATEIN si existe el esquema de SQL.

rschema
Argumento de entrada y salida de tipo VARCHAR (128) que especifica el esquema de SQL para la DTD. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este DTD en el XSR. (La otra parte del identificador de SQL viene suministrada por el argumento de *nombre*.) Este argumento puede tener un valor NULL, que indica que el esquema de SQL por omisión se utiliza, tal como se define en el registro especial CURRENT SCHEMA. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento. Los esquemas relacionales que comienzan con la serie 'SYS' no se deben utilizar para este valor. Los objetos de XSR no experimentarán colisiones de nombres con objetos de base de datos que existan fuera del XSR, ya que los objetos de XSR aparecen en un espacio de nombres diferente de los objetos que están fuera del depósito de esquemas XML.

nombre
Argumento de entrada y de salida de tipo VARCHAR (128) que especifica el nombre del DTD. El identificador de SQL completo para el DTD es *rschema.name* y debe ser exclusivo entre todos los objetos del XSR. Este argumento acepta un valor NULL. Cuando se proporciona un valor NULL para este argumento, se genera un valor exclusivo y se almacena dentro del XSR. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

id-sistema
Parámetro de entrada de tipo VARCHAR (1000) que especifica el identificador del sistema del DTD. El ID del sistema del DTD debería corresponderse con el identificador de recursos uniforme del DTD de la declaración de DOCTYPE del documento de instancia XML o de una declaración de ENTITY (con la palabra clave SYSTEM como prefijo, en el caso de que se utilice). Este argumento no puede tener un valor NULL. El ID de sistema se puede especificar junto con un ID público.

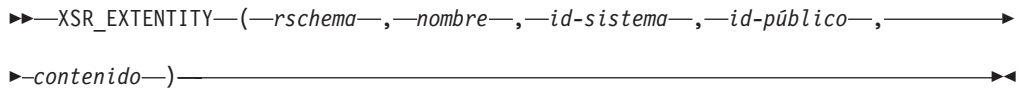
id-público
Parámetro de entrada de tipo VARCHAR (1000) que especifica el identificador público del DTD. El ID público de un DTD debería corresponderse con el identificador de recursos uniforme del DTD de la declaración de DOCTYPE del documento de instancia XML o de una declaración de ENTITY (con la palabra clave PUBLIC como prefijo, en el caso de que se utilice). Este argumento acepta un valor NULL y sólo debería utilizarse si también se especifica en la declaración de DOCTYPE del documento de instancia XML o en una declaración de ENTITY.

contenido
Parámetro de entrada de tipo BLOB (30M) que contiene el contenido del documento DTD. Este argumento no puede tener un valor NULL.

Ejemplo: Registre el DTD identificado por medio del ID de sistema *http://www.test.com/person.dtd* y del ID público *http://www.test.com/person*:

```
CALL SYSPROC.XSR_DTD ( 'MYDEPT' ,
   'PERSONDTD' ,
   'http://www.test.com/person.dtd' ,
   'http://www.test.com/person',
   :content_host_variable
   )
```

XSR_EXTENTITY



El esquema es SYSPROC.

El procedimiento XSR_EXTENTITY registra una entidad externa con el depósito de esquemas XML (XSR).

Autorización

El ID de autorización de quien realiza la llamada del procedimiento debe tener como mínimo uno de los siguientes:

- Autorización DBADM.
- Autorización de base de datos IMPLICIT_SCHEMA si no existe el esquema de SQL.
- Privilegio CREATEIN si existe el esquema de SQL.

rschema
Argumento de entrada y salida de tipo VARCHAR (128) que especifica el esquema de SQL para la entidad externa. El esquema de SQL es una parte del identificador de SQL utilizado para identificar esta entidad externa en el XSR. (La otra parte del identificador de SQL viene suministrada por el argumento de *nombre*.) Este argumento puede tener un valor NULL, que indica que el esquema de SQL por omisión se utiliza, tal como se define en el registro especial CURRENT SCHEMA. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento. Los esquemas relacionales que comienzan con la serie 'SYS' no se deben utilizar para este valor. Los objetos de XSR no experimentarán colisiones de nombres con objetos de base de datos que existan fuera del XSR, ya que los objetos de XSR aparecen en un espacio de nombres diferente de los objetos que están fuera del depósito de esquemas XML.

nombre
Argumento de entrada y de salida de tipo VARCHAR (128) que especifica el nombre de la entidad externa. El identificador de SQL completo para la entidad externa es *rschema.name* y debe ser exclusivo entre todos los objetos del XSR. Este argumento acepta un valor NULL. Cuando se proporciona un valor NULL para este argumento, se genera un valor exclusivo y se almacena dentro del XSR. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

id-sistema
Parámetro de entrada de tipo VARCHAR (1000) que especifica el identificador de la entidad externa. El ID del sistema de la entidad externa debería corresponderse con el identificador de recursos uniforme de la entidad externa

de la declaración de ENTITY (con la palabra clave SYSTEM como prefijo, en el caso de que se utilice). Este argumento no puede tener un valor NULL. El ID de sistema se puede especificar junto con un ID público.

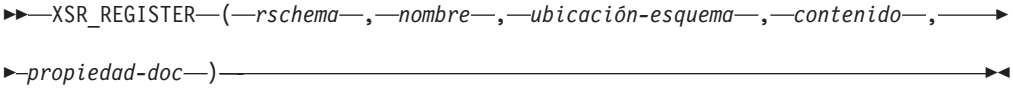
id-público
Parámetro de entrada de tipo VARCHAR (1000) que especifica el identificador público de la entidad externa. El ID público de una entidad externa debería corresponderse con el identificador de recursos uniforme de la entidad externa de la declaración de ENTITY (con la palabra clave PUBLIC como prefijo, en el caso de que se utilice). Este argumento acepta un valor NULL y sólo debería utilizarse si también se especifica en la declaración de DOCTYPE del documento de instancia XML o en una declaración de ENTITY.

contenido
Parámetro de entrada de tipo BLOB (30M) que contiene el contenido del documento de la entidad externa. Este argumento no puede tener un valor NULL.

Ejemplo: Registre las entidades externas identificadas mediante los identificadores del sistema *http://www.test.com/food/chocolate.txt* y *http://www.test.com/food/cookie.txt*:

```
CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
   'CHOCLATE' ,
   'http://www.test.com/food/chocolate.txt' ,
   NULL ,
   :content_of_chocolate.txt_as_a_host_variable
   )

CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
   'COOKIE' ,
   'http://www.test.com/food/cookie.txt' ,
   NULL ,
   :content_of_cookie.txt_as_a_host_variable
   )
```

XSR_REGISTER



El esquema es SYSPROC.

El procedimiento XSR_REGISTER es el primer procedimiento almacenado al que se llama como parte del proceso de registro del esquema XML, que registra esquemas XML con el depósito de esquemas XML (XSR).

Autorización

El ID de autorización de quien realiza la llamada del procedimiento debe tener como mínimo uno de los siguientes:

- Autorización DBADM.
- Autorización de base de datos IMPLICIT_SCHEMA si no existe el esquema de SQL.
- Privilegio CREATEIN si existe el esquema de SQL.

rschema
Argumento de entrada y salida de tipo VARCHAR (128) que especifica el esquema de SQL para el esquema XML. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este esquema XML en el XSR. (La otra parte del identificador de SQL viene suministrada por el argumento de nombre.) Este argumento puede tener un valor NULL, que indica que el esquema de SQL por omisión se utiliza, tal como se define en el registro especial CURRENT SCHEMA. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento. Los esquemas relacionales que comienzan con la serie 'SYS' no se deben utilizar para este valor. Los objetos de XSR no experimentarán colisiones de nombres con objetos de base de datos que existan fuera del XSR, ya que los objetos de XSR aparecen en un espacio de nombres diferente de los objetos que están fuera del depósito de esquemas XML.

nombre
Argumento de entrada y de salida de tipo VARCHAR (128) que especifica el nombre del esquema XML. El identificador de SQL completo para el esquema XML es *rschema.name* y debe ser exclusivo entre todos los objetos del XSR. Este argumento acepta un valor NULL. Cuando se proporciona un valor NULL para este argumento, se genera un valor exclusivo y se almacena dentro del XSR. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

ubicación-esquema
Argumento de entrada de tipo VARCHAR (1000), que puede tener un valor NULL, que indica la ubicación de esquema del documento de esquema XML primario. Este argumento es el nombre externo del esquema XML, es decir, el documento primario se puede identificar con el atributo xsi:schemaLocation en los documentos de instancia XML.

contenido
Parámetro de entrada de tipo BLOB (30M) que contiene el contenido del documento de esquema XML primario. Este argumento no puede tener un valor NULL; se debe proporcionar un documento de esquema XML.

propiedad-doc
Parámetro de entrada de tipo BLOB (5M) que indica las propiedades para el documento de esquema XML primario. Este parámetro puede tener un valor NULL; en caso contrario, el valor es un documento XML.

Ejemplo: El ejemplo siguiente muestra cómo llamar al procedimiento XSR_REGISTER desde la línea de mandatos:

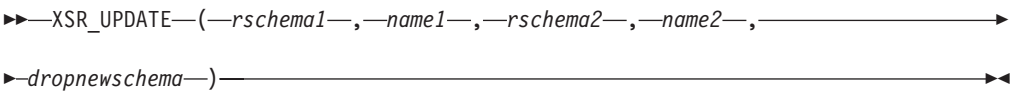
```
CALL SYSPROC.XSR_REGISTER(
  'user1',
  'POschema',
  'http://myPOschema/PO.xsd',
  :content_host_var,
  :docproperty_host_var)
```

Ejemplo: El ejemplo siguiente muestra cómo llamar al procedimiento XSR_REGISTER desde un programa de aplicación Java:

```
stmt = con.prepareCall("CALL SYSPROC.XSR_REGISTER (?, ?, ?, ?, ?)");
String xsrObjectName = "myschema1";
String xmlSchemaLocation = "po.xsd";
stmt.setNull(1, java.sql.Types.VARCHAR);
stmt.setString(2, xsrObjectName);
stmt.setString(3, xmlSchemaLocation);
stmt.setBinaryStream(4, buffer, (int)length);
```

```
stmt.setNull(5, java.sql.Types.BLOB);
stmt.registerOutParameter(1, java.sql.Types.VARCHAR);
stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
stmt.execute();
```

XSR_UPDATE



El esquema es SYSPROC.

El procedimiento almacenado XSR_UPDATE se utiliza para desarrollar un esquema XML existente en el depósito de esquemas XML (XSR). Esto le permite modificar o ampliar un esquema XML existente de modo que pueda utilizarse para validar tanto los documentos XML ya existentes como los recién insertados.

El esquema XML original y el nuevo esquema XML especificados como argumentos en XSR_UPDATE deben registrarse y completarse en el XSR antes de que se llame el procedimiento. Estos esquemas XML también deben ser compatibles. Para obtener más detalles sobre los requisitos de compatibilidad consulte *Requisitos de compatibilidad para desarrollar un esquema XML*.

Autorización

El ID de autorización de la sentencia del que llama debe tener al menos uno de los privilegios siguientes:

- Autorización DBADM.
- Privilegio SELECT en las vistas de catálogo SYSCAT.XSROBJECTS y SYSCAT.XSROBJECTCOMPONENTS y uno de los siguientes conjuntos de privilegios:
  - OWNER del esquema XML especificado por medio del esquema de SQL *rschema1* y el nombre del objeto *name1*
  - El privilegio ALTERIN del esquema SQL especificado por medio del argumento *rschema1* y, si el argumento *dropnewschema* no es igual a cero, el privilegio DROPIN del esquema de SQL especificado por medio del argumento *rschema2*.

rschema1
Argumento de entrada de tipo VARCHAR (128) que especifica el esquema de SQL para el esquema XML original que ha de actualizarse. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este esquema XML en el XSR. (La otra parte del identificador de SQL viene suministrada por el argumento de *name1*.) Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

name1
Argumento de entrada de tipo VARCHAR (128) que especifica el nombre del esquema XML original que ha de actualizarse. El identificador de SQL completo para el esquema XML es *rschema1.name1*. Este esquema XML ya debe estar registrado y finalizado en el XSR. Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

rschema2
Argumento de entrada de tipo VARCHAR (128) que especifica el esquema de SQL para el esquema XML nuevo que se utilizará para actualizar el esquema XML original. El esquema de SQL es una parte del identificador de SQL utilizado para identificar este esquema XML en el XSR. (La otra parte del identificador de SQL viene suministrada por el argumento de *name2*.) Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

name2
Argumento de entrada de tipo VARCHAR (128) que especifica el nombre del esquema XML nuevo que se utilizará para actualizar el esquema XML original. El identificador de SQL completo para el esquema XML es *rschema2.name2*. Este esquema XML ya debe estar registrado y finalizado en el XSR. Este argumento no puede tener un valor NULL. Las normas para caracteres y delimitadores válidos que se aplican a cualquier identificador de SQL también se aplican a este argumento.

dropnewschema
Un parámetro de entrada de tipo entero que indica si el nuevo esquema XML nuevo debe descartarse después de utilizarlo para actualizar el esquema XML original. Si este parámetro se establece en cualquier valor distinto de cero, el nuevo esquema XML se descartará. Este argumento no puede tener un valor nulo.

Ejemplo:

```
CALL SYSPROC.XSR_UPDATE(
  'STORE',
  'PROD',
  'STORE',
  'NEWPROD',
  1)
```

El contenido del esquema XML STORE.PROD se actualiza con el contenido de STORE.NEWPROD y se descarta el esquema XML STORE.NEWPROD.

Capítulo 5. Consultas de SQL

Una *consulta* especifica una tabla resultante. Una consulta es un componente de algunas sentencias de SQL. Las tres formas de una consulta son:

- subselección
- selección completa
- sentencia-select.

Autorización

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Para cada tabla o vista identificada en la consulta, uno de los siguientes:
  - Privilegio SELECT para la tabla o vista
  - Privilegio CONTROL sobre la tabla o vista
- Autorización DATAACCESS

Para cada variable global utilizada como expresión en la consulta, los privilegios con los que cuenta el ID de autorización de la sentencia deben incluir uno de los 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 la consulta contiene una sentencia de cambio de datos de SQL, los requisitos de autorización de la sentencia también se aplican a la consulta.

Los privilegios de grupo, con excepción de PUBLIC, no se comprueban para las consultas contenidas en sentencias de SQL estático o en sentencias DDL.

Para los apodos, los requisitos de autorización de la fuente de datos para el objeto al que el apodo hace referencia se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización diferente en la fuente de datos.

Consultas y expresiones de tabla

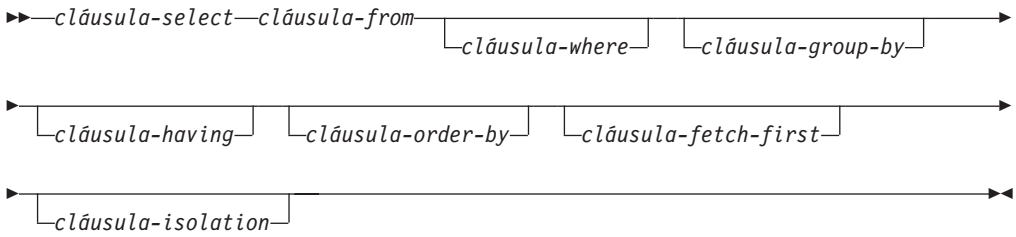
Una *consulta* es un componente de determinadas sentencias SQL; especifica una tabla de resultados (temporal).

Una *expresión de tabla* crea una tabla de resultados temporal a partir de una consulta sencilla. Las cláusulas definen mejor la tabla de resultados. Por ejemplo, puede utilizar una expresión de tabla como una consulta para seleccionar todos los directores de varios departamentos, que deben tener una experiencia laboral de más de 15 años y que deben estar ubicados en la oficina de Nueva York.

Una *expresión de tabla común* es como una vista temporal dentro de una consulta compleja. Se puede hacer referencia a la misma en otros puntos de la consulta y se puede utilizar en lugar de una vista. Cada uso de una expresión de tabla común dentro de una consulta compleja comparte la misma vista temporal.

El uso recurrente de una expresión de tabla común dentro de una consulta se puede utilizar para dar soporte a aplicaciones como sistemas de reservar de líneas aéreas, generadores de material de envío (BOM) y planificación de red.

subselección



La *subselección* es un componente de la selección completa.

Una subselección especifica una tabla resultante que deriva de las tablas, vistas o apodos identificados en la cláusula FROM. La derivación puede describirse como una secuencia de operaciones en las que el resultado de cada operación es la entrada de la siguiente. (Es la única manera de describir la subselección. El método utilizado para realizar la derivación puede ser bastante distinto del que aquí se describe. Si existe alguna parte de la subselección que en realidad no es necesario ejecutar para obtener el resultado correcto, es posible que no se ejecute.)

La autorización para una *subselección* se describe en la sección de autorización en ″Consultas de SQL″.

Las cláusulas de la subselección se procesan en el orden siguiente:

1. FROM, cláusula
2. WHERE, cláusula
3. GROUP BY, cláusula
4. HAVING, cláusula
5. SELECT, cláusula
6. ORDER BY, cláusula
7. Cláusula FETCH FIRST

Una subselección que contenga una cláusula ORDER BY o FETCH FIRST no puede especificarse:

- En la selección completa más exterior de una vista.
- En una tabla de consulta materializada.
- A menos que la subselección esté entre paréntesis.

Por ejemplo, lo siguiente no es válido (SQLSTATE 428FJ):

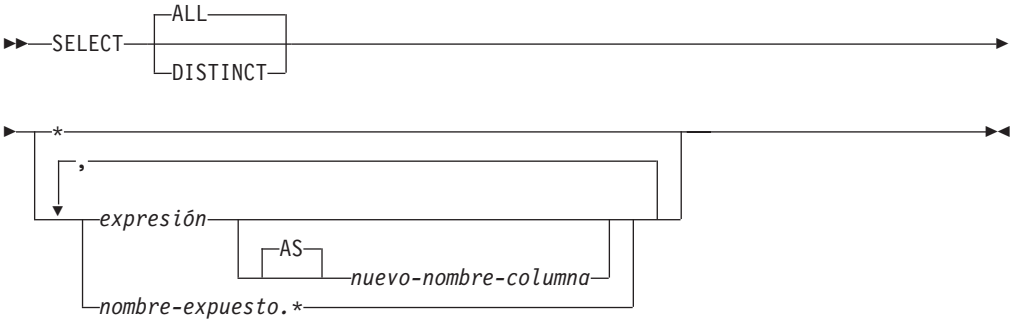
```
SELECT * FROM T1
   ORDER BY C1
UNION
SELECT * FROM T2
   ORDER BY C1
```

El ejemplo siguiente *sí* es válido:

```
(SELECT * FROM T1
   ORDER BY C1)
UNION
(SELECT * FROM T2
   ORDER BY C1)
```

Nota: Una cláusula ORDER BY en una subselección no afecta el orden de las filas que una consulta devuelve. Una cláusula ORDER BY sólo afecta el orden de las filas devueltas si se especifica en la selección completa más externa.

cláusula-select



La cláusula SELECT especifica las columnas de la tabla de resultados final, R. La aplicación produce los valores de columna de la *lista de selección* a R. La lista de selección son los nombres o expresiones especificados en la cláusula SELECT y R es el resultado de la operación anterior de la subselección. Por ejemplo, si las únicas cláusulas especificadas con SELECT, FROM y WHERE, R es el resultado de la cláusula WHERE.

ALL
Retiene todas las filas de la tabla resultante final y no elimina los duplicados redundantes. Es el valor por omisión.

DISTINCT
Elimina todas las filas excepto una de los juegos de filas duplicadas de la tabla resultante final. Si se utiliza DISTINCT, ninguna columna de tipo serie de la tabla resultante puede ser un tipo LOB, un tipo diferenciado basado en LOB ni un tipo estructurado. DISTINCT puede utilizarse más de una vez en una subselección. Esto incluye SELECT DISTINCT, la utilización de DISTINCT en una función agregada de la lista de selección o una cláusula HAVING y las subconsultas de la subselección.

Dos filas sólo son duplicadas una de la otra si cada valor de la primera es igual al valor correspondiente de la segunda. Para la determinación de duplicados, dos valores nulos se consideran iguales y dos representaciones de coma flotante decimal diferentes del mismo número se consideran iguales. Por ejemplo, -0 es igual a +0 y 2,0 es igual a 2,00. También se consideran iguales a cada uno de los valores especiales de coma flotante decimal: -NAN equivale a -NAN, -SNAN equivale s -SNAN, -INFINITY equivale a -INFINITY, INFINITY equivale a INFINITY, SNAN equivale a SNAN y NAN equivale a NAN.

Cuando el tipo de datos de una columna es de coma flotante decimal y hay varias representaciones del mismo número en la columna, el valor de partición que se devuelve para SELECT DISTINCT puede ser cualquiera de las representaciones de la columna. Para obtener más información, consulte el apartado “Comparaciones numéricas” en la página 133.

Por compatibilidad con otras implementaciones de SQL, se puede especificar UNIQUE como sinónimo de DISTINCT.

Notación de lista de selección

* Representa una lista de nombres que identifican las columnas de la tabla R, excluidas las columnas definidas como IMPLICITLY HIDDEN. El primer nombre de la lista identifica la primera columna de R, el segundo nombre identifica la segunda columna de R y así sucesivamente.

La lista de nombres se establece cuando se vincula el programa que contiene la cláusula SELECT. Por lo tanto, * (el asterisco) no identifica ninguna columna que se haya añadido a la tabla después de vincular la sentencia que contiene la referencia a la tabla.

expresión
Especifica los valores de una columna del resultado. Puede ser cualquier expresión que sea un elemento válido en el lenguaje SQL pero normalmente incluye nombres de columnas. Cada nombre de columna utilizado en la lista de selección debe identificar sin ambigüedades una columna R. El tipo de resultado de la expresión no puede ser un tipo de fila (SQLSTATE 428H2).

nuevo-nombre-columna **o AS** *nuevo-nombre-columna*
Nombra o cambia el nombre de la columna del resultado. El nombre no debe estar calificado y no tiene que ser exclusivo. El uso subsiguiente del nombre-columna está limitado en lo siguiente:

- Un *nuevo-nombre-columna* especificado en la cláusula AS se puede utilizar en la cláusula-order-by, siempre que sea exclusivo.
- Un *nuevo-nombre-columna* especificado en la cláusula AS de la lista de selección no se puede utilizar en ninguna otra cláusula de la subselección (cláusula-where, cláusula-group-by o cláusula-having).
- Un *nuevo-nombre-columna* especificado en la cláusula AS no se puede utilizar en la cláusula-update.
- Un *nuevo-nombre-columna* especificado en la cláusula AS se conoce fuera de la selección completa de las expresiones de tabla anidadas, las expresiones de tablas comunes y CREATE VIEW.

*nombre.**
Representa la lista de nombres que identifican las columnas de la tabla resultante identificada por *nombre-expuesto*, excluidas las columnas definidas como IMPLICITLY HIDDEN. El *nombre-expuesto* puede ser un nombre de tabla, un nombre de vista, un apodo o un nombre de correlación, y debe designar una tabla, una vista o un apodo especificado en la cláusula FROM. El primer nombre de la lista identifica la primera columna de la tabla, vista o apodo, el segundo nombre de la lista identifica la segunda columna de la tabla, vista o apodo, y así sucesivamente.

La lista de nombres se establece cuando se vincula la sentencia que contiene la cláusula SELECT. Por lo tanto, * no identifica ninguna columna que se haya añadido a la tabla después de vincular la sentencia.

El número de columnas del resultado de SELECT es igual al número de expresiones de la forma operativa de la lista de selección (es decir, la lista establecida cuando se ha preparado la sentencia) y no puede exceder de 500 para una página de 4K de tamaño o de 1012 para una página de 8K, 16K o 32K de tamaño.

Límites en las columnas de una serie

Para ver las limitaciones en la lista de selección, consulte “Restricciones al utilizar series de caracteres de longitud variable”.

Aplicación de la lista de selección

Algunos resultados de aplicar la lista de selección en R dependen de si se utiliza GROUP BY o HAVING o no. Los resultados se describen en dos listas separadas.

Si se utiliza GROUP BY o HAVING

- Una expresión *X* (no una función agregada) que se utilice en la lista de selección debe tener una cláusula GROUP BY con:
  - una *expresión-agrupación* en la cual cada expresión o nombre-columna identifique sin ambigüedades una columna de R (consulte el apartado "cláusula-group-by" en la página 742) o
  - cada columna de R a la que *X* haga referencia como una *expresión-agrupación* separada.
- La lista de selección se aplica a cada grupo de R y el resultado contiene tantas filas como grupos haya en R. Cuando la lista de selección se aplica a un grupo de R, este grupo es la fuente de los argumentos de las funciones agregadas de la lista de selección.

Si no se utiliza GROUP BY ni HAVING

- La lista de selección no debe incluir ninguna función agregada o cada *nombre-columna* de la lista de selección debe estar especificado en una función agregada o debe ser una referencia de columna correlacionada.
- Si la selección no incluye funciones agregadas, la lista de selección se aplica a cada fila de R y el resultado contiene tantas filas como el número de filas en R.
- Si la lista de selección es una lista de funciones agregadas, R es la fuente de los argumentos de las funciones y el resultado de aplicar la lista de selección es una fila.

En cualquier caso la columna *n* del resultado contiene los valores especificados por la aplicación de la expresión *n* en la forma operativa de la lista de selección.

Atributos nulos de columnas del resultado

Las columnas del resultado no permiten valores nulos si se derivan de:

- Una columna que no permite valores nulos
- Una constante
- La función COUNT o COUNT_BIG
- Una variable del lenguaje principal que no tiene una variable de indicador
- Una función o expresión escalar no incluye un operando que permita nulos

Las columnas del resultado permiten valores nulos si se derivan de:

- Cualquier función agregada excepto COUNT o COUNT_BIG
- Una columna que permite valores nulos
- Una función o expresión escalar que incluye un operando que permite nulos
- Una función NULLIF con argumentos que contienen valores iguales
- Una variable del lenguaje principal que tiene una variable de indicador, un parámetro de SQL, una variable de SQL o una variable global
- Un resultado de una operación de conjunto si como mínimo uno de los elementos correspondientes de la lista de selección puede ser nulo

- Una expresión aritmética o una columna de vista que se deriva de una expresión aritmética y la base de datos se ha configurado con **dft_sqlmathwarn** establecido en Yes
- Una subselección escalar
- Una operación de desreferencia
- A GROUPING SETS *expresión-agrupación*

Nombres de las columnas del resultado

- Si se especifica la cláusula AS, el nombre de la columna del resultado es el nombre especificado en esta cláusula.
- Si no se especifica la cláusula AS y se especifica una lista de columnas en la cláusula de correlaciones, el nombre de la columna de resultados es el nombre correspondiente en la lista de columnas de correlación.
- Si no se especifica ni una cláusula AS ni una lista de columnas en la cláusula de correlación y la columna del resultado se deriva únicamente de una única columna (sin ninguna función u operador), el nombre de columna del resultado es el nombre no calificado de dicha columna.
- Si no se especifica ni una cláusula AS ni una lista de columnas en la cláusula de correlación y la columna del resultado se deriva únicamente de una única variable SQL o parámetro SQL (sin ninguna función u operador), el nombre de columna del resultado es el nombre no calificado de dicha variable SQL o parámetro SQL.
- Si no se especifica ni una cláusula AS ni una lista de columnas en la cláusula de correlación y la columna del resultado se deriva mediante una operación de desreferencia, el nombre de columna del resultado es el nombre no calificado de la columna de destino de la operación de desreferencia.
- Todos los demás nombres de columnas del resultado carecen de nombre. El sistema asigna números temporales (como series de caracteres) a estas columnas.

Tipos de datos de las columnas del resultado

Cada columna del resultado de SELECT adquiere un tipo de datos de la expresión de la que se deriva.

Cuando la expresión es ...	**El tipo de datos de la columna del resultado es ...**
el nombre de cualquier columna numérica	el mismo que el tipo de datos de la columna, con la misma precisión y escala que para las columnas DECIMAL, o la misma precisión que para las columnas DECFLOAT.
Una constante	El mismo tipo de datos de la constante.
el nombre de cualquier variable numérica	el mismo que el tipo de datos de la variable, con la misma precisión y escala que para las variables DECIMAL, o la misma precisión que para las variables DECFLOAT.
el nombre de cualquier columna de serie	el mismo que el tipo de datos de la columna, con el mismo atributo de longitud.
el nombre de cualquier variable de serie.	el mismo que el tipo de datos de la variable, con el mismo atributo de longitud; si el tipo de datos de la variable no es idéntico a un tipo de datos SQL (por ejemplo, una serie terminada en NUL en C), la columna del resultado es una serie de longitud variable.

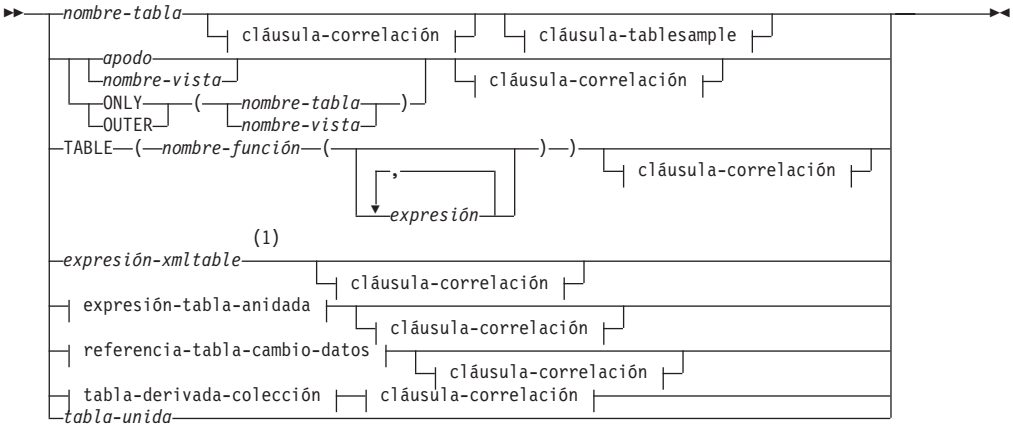
Cuando la expresión es ...	El tipo de datos de la columna del resultado es ...
el nombre de una columna de indicación de fecha y hora	el mismo tipo de datos de la columna.
el nombre de una columna de tipo definido por el usuario	el mismo tipo de datos de la columna.
el nombre de una columna de tipo de referencia	el mismo tipo de datos de la columna.

cláusula-from

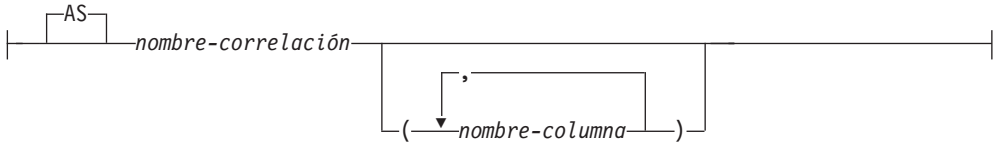
La cláusula FROM especifica una tabla resultante intermedia.

Si se especifica sólo una *referencia-tabla*, la tala resultante intermedia es simplemente el resultado de dicha *referencia-tabla*. Si se especifican más de una *referencia-tabla*, la tabla resultante intermedia consta de todas las combinaciones posibles de las filas de la *referencia-tabla* especificadas (el producto cartesiano). Cada fila del resultado es una fila de la primera *referencia-tabla* concatenada con una fila de la segunda *referencia-tabla*, concatenada a su vez con una fila de la tercera, y así sucesivamente. El número de filas del resultado es el producto del número de filas de todas las referencias a tabla individuales. Para obtener una descripción de *referencia-tabla*, consulte "referencia-tabla".

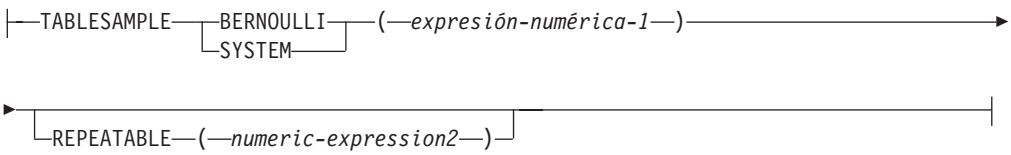
referencia-tabla



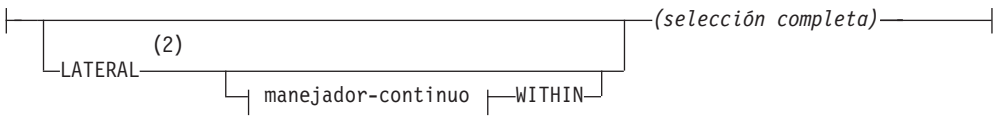
cláusula-correlación:



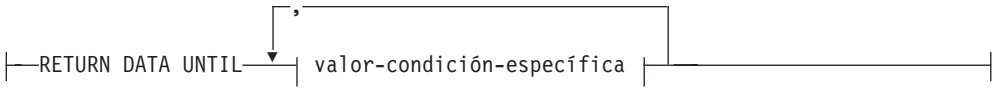
cláusula-tablesample:



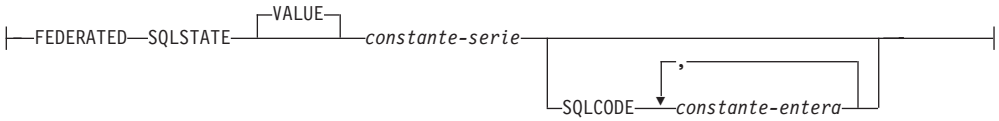
expresión-tabla-anidada:



manejador-continuo:



valor-condición-específica:



referencia-tabla-cambio-datos:

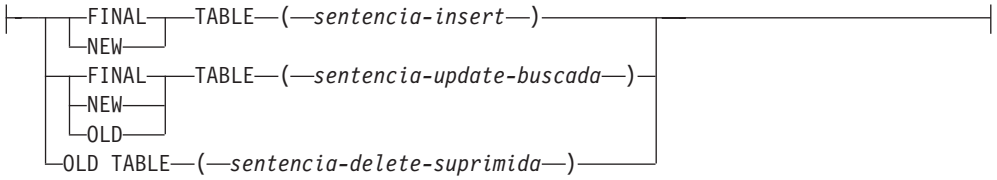
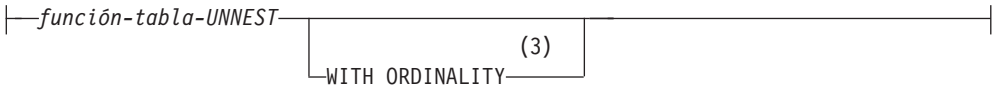


tabla-derivada-colección:



Notas:

1 Una expresión XMLTABLE puede formar parte de una referencia de tabla. En este caso, las subexpresiones de la expresión XMLTABLE se encuentran en el ámbito de variables de un rango anterior de la cláusula FROM. Para obtener más información, consulte la descripción de "XMLTABLE".

2 Puede especificarse TABLE en lugar de LATERAL.

3 WITH ORDINALITY solo se puede especificar si el argumento de la función de tabla UNNEST es una o más variables de matriz ordinarias; no se puede especificar una variable de matriz asociativa(SQLSTATE 428HT).

Cada *nombre-tabla*, *nombre-vista* o *apodo* especificado como referencia-tabla debe identificar una tabla, vista o apodo existente del servidor de aplicaciones o el *nombre-tabla* de una expresión de tabla común definida antes de la selección completa que contiene la referencia-tabla. Si *nombre-tabla* hace referencia a una tabla con tipo, el nombre indica UNION ALL de la tabla con todas sus subtablas y solamente con las columnas de *nombre-tabla*. De manera similar, si *nombre-vista* hace referencia a una vista con tipo, el nombre indica UNION ALL de la vista con todas sus subvistas y solamente con las columnas de *nombre-vista*.

El uso de ONLY(*nombre-tabla*) u ONLY(*nombre-vista*) significa que no se incluyen las filas de las subtablas o subvistas correspondientes. Si el *nombre-tabla* utilizado con ONLY no tiene subtablas, ONLY(*nombre-tabla*) equivale a especificar *nombre-tabla*. Si el *nombre-vista* utilizado con ONLY no tiene subvistas, ONLY(*nombre-vista*) equivale a especificar *nombre-vista*.

El uso de OUTER(*nombre-tabla*) u OUTER(*nombre-vista*) representa una tabla virtual. Si el *nombre-tabla* o el *nombre-vista* que se utilice con OUTER no tiene subtablas o subvistas, especificar OUTER equivale a no especificar OUTER.
OUTER(*nombre-tabla*) se deriva de *nombre-tabla* de la manera siguiente:

- En las columnas, se incluyen las columnas de *nombre-tabla* seguidas de las columnas adicionales que ha introducido cada una de sus subtablas (si existen). Las columnas adicionales se añaden a la derecha atravesando la jerarquía de las subtablas por orden de importancia. Se atraviesan las subtablas que tienen un padre común en el orden de creación de sus tipos.
- En las filas, se incluyen todas las filas de *nombre-tabla* y todas las filas de sus subtablas. Se devuelven valores nulos para las columnas que no están en la subtabla para la fila.

Los puntos anteriores también se aplican a OUTER(*nombre-vista*) si se sustituye *nombre-tabla* por *nombre-vista* y subtabla por subvista.

El uso de ONLY o OUTER requiere el privilegio SELECT en cada subtabla de *nombre-tabla* o subvista de *nombre-vista*.

Cada *nombre-función*, junto con los tipos de sus argumentos, especificado como una referencia a tabla, debe resolverse en una función de tabla existente en el servidor de aplicaciones.

Una selección completa entre paréntesis se denomina *expresión de tabla anidada*.

Una *tabla-unida* especifica un conjunto resultante intermedio que es el resultado de una o varias operaciones de unión. Para obtener más información, consulte el apartado “tabla-unida” en la página 740.

Los nombres expuestos de todas las referencias a tabla deben ser exclusivos. Un nombre expuesto es:

- Un *nombre-correlación*
- Un *nombre-tabla* que no va seguido de un *nombre-correlación*
- Un *nombre-vista* que no va seguido de un *nombre-correlación*
- Un *apodo* que no va seguido de un *nombre-correlación*
- Un *nombre-alias* que no va seguido de un *nombre-correlación*

Si una *cláusula-correlación* no sigue a una referencia *nombre-función*, una *expresión-tablaxml*, una expresión de tabla anidada o una *referencia-tabla-cambio-datos*, no hay ningún nombre expuesto para dicha referencia de tabla.

Cada *nombre-correlación* se define como designador del *nombre-tabla*, del *nombre-vista*, del *apodo*, de la referencia *nombre-función*, de la *expresión-tablaxml*, de la expresión de tabla anidada o de la *referencia-tabla-cambio-datos* inmediatamente anterior. Cualquier referencia calificada a una columna debe utilizar el nombre expuesto. Si se especifica dos veces el mismo nombre de tabla, vista o apodo, como mínimo una especificación debe ir seguida de un *nombre-correlación*. El *nombre-correlación* se utiliza para calificar las referencias a las columnas de la tabla, vista o apodo. Cuando se especifica un *nombre-correlación*, también se pueden especificar *nombres-columna* para proporcionar nombres a las columnas de la referencia de tabla. Si la *cláusula-correlación* no incluye *nombres-columna*, los nombres de columna expuestos se determinan del modo siguiente:.

- Nombres de columna de la tabla, vista o apodo de referencia cuando la *referencia-tabla* es un *nombre-tabla*, *nombre-vista*, *apodo* o *nombre-alias*
- Nombres de columna especificados en la cláusula RETURNS de la sentencia CREATE FUNCTION cuando la *referencia-tabla* es una referencia de *nombre-función*
- Nombre de columna especificados en la cláusula COLUMNS de la *expresión-tablaxml* cuando la *referencia-tabla* es una *expresión-tablaxml*
- Nombres de columna expuestos por la selección completa cuando la *referencia-tabla* es una *expresión-tabla-anidada*
- Nombres de columna de la tabla de destino de la sentencia de cambio de datos, junto con las columnas INCLUDE definidas cuando la *referencia-tabla* es una *referencia-tabla-cambio-datos*

En general, las tablas-derivadas-colección, las funciones de tabla y las expresiones de tabla anidadas se pueden especificar en cualquier cláusula-from. Se puede hacer referencia a las columnas de las funciones de tabla, de las expresiones de tablas anidadas o de las tablas derivadas de colección en la lista de selección y en el resto de la subselección utilizando el nombre de correlación. El ámbito de este nombre de correlación es el mismo que los nombres de correlación de otras tablas, vistas o apodos de la cláusula FROM. Una expresión de tabla anidada puede utilizarse:

- En el lugar de una vista para evitar la creación de la vista (cuando no es necesario el uso general de la vista)
- Cuando la tabla resultante deseada se basa en variables del lenguaje principal

Se puede utilizar una *tabla-derivada-colección* para convertir los elementos de matrices en valores de una columna en filas separadas.Si se especifica WITH ORDINALITY, se añade una columna extra con tipo de datos INTEGER. Esta columna contiene la posición del elemento en la matriz. Se puede hacer referencia a las columnas en la lista de selección y el resto de la subselección utilizando los nombres especificados para las columnas en la cláusula-correlación. La cláusula *tabla-derivada-colección* sólo se puede utilizar en un contexto donde se dé soporte a matrices (SQLSTATE 42887). Consulte “Función de tabla UNNEST” para obtener más detalles.

Una expresión de la lista de selección de una expresión de tabla anidada a la que se hace referencia dentro, o que representa el destino de, una sentencia de cambio de datos dentro de una selección completa sólo es válida cuando no incluye:

- Una función que lee o modifica datos de SQL
- Una función que no es determinante

- Una función que tiene acción externa
- Una función OLAP

Si se hace referencia directamente a una vista en, o es el destino de, una expresión de tabla anidada de una sentencia de cambio de datos dentro de una cláusula FROM, la vista debe ser simétrica (debe tener especificado WITH CHECK OPTION) o debe satisfacer la restricción correspondiente a una vista WITH CHECK OPTION.

Si el destino de una sentencia de cambio de datos dentro de una cláusula FROM es una expresión de tabla anidada, las filas modificadas no se vuelven a calificar, los predicados de la cláusula WHERE no se vuelven a evaluar y las operaciones ORDER BY o FETCH FIRST no se vuelven a realizar.

La cláusula-tablesample opcional se puede utilizar para obtener un subconjunto aleatorio (un ejemplo) de las filas a partir del *nombre-tabla* especificado, en lugar del contenido completo de dicho *nombre-tabla*, para esta consulta. Este muestreo se añade a cualquier predicado especificado en la *cláusula-where*. A no ser que se especifique la cláusula REPEATABLE opcional, cada ejecución de la consulta generará un ejemplo distinto, excepto en casos en los que la tabla sea tan pequeña en relación al tamaño del ejemplo que cualquier ejemplo deba devolver las mismas filas. El tamaño del ejemplo se controla mediante *expresión-numérica1* entre paréntesis, que representa un porcentaje aproximado (P) de la tabla que se va a devolver. El método por el que se obtiene el ejemplo se especifica tras la palabra clave TABLESAMPLE y puede ser BERNOULLI o SYSTEM. Para ambos métodos, el número exacto de filas del ejemplo puede ser distinto para cada ejecución de la consulta, pero en promedio debe ser aproximadamente P por ciento de la tabla, antes de que los predicados reduzcan el número de filas.

El *nombre-tabla* debe ser una tabla almacenada. Puede ser un nombre de tabla de consultas materializadas (MQT), pero no una subselección o expresión de tabla para la que se haya definido una MQT, porque no hay ninguna garantía de que el gestor de bases de datos vaya a direccionar a la MQT para dicha subselección.

Semánticamente, el muestreo de una tabla se produce antes de cualquier otro proceso de consulta, como por ejemplo aplicar predicados o realizar uniones. Los accesos repetidos de una tabla de muestreo dentro de una sola ejecución de una consulta (como en una unión de bucle anidado o una subconsulta correlacionada) devolverán el mismo ejemplo. Se pueden obtener ejemplos de más de una tabla en una consulta.

El muestreo BERNOULLI considera cada fila de forma individual. Incluye cada fila en el ejemplo con la probabilidad P/100 (donde P es el valor de *expresión-numérica1*) y ejecuta cada fila con la probabilidad 1 - P/100, independientemente de las demás filas. De modo que si *expresión-numérica1* tiene el valor 10, lo que significa un ejemplo del diez por ciento, cada fila se incluiría con la probabilidad 0,1 y se excluiría con la probabilidad 0,9.

El muestreo SYSTEM permite al gestor de bases de datos determinar la forma más eficiente de realizar el muestreo. En la mayoría de los casos, el muestreo SYSTEM aplicado a *nombre-tabla* significa que cada página de *nombre-tabla* se incluye en el ejemplo con una probabilidad de P/100 y se excluye con una probabilidad de 1 - P/100. Todas las filas de cada página que se incluye están cualificadas para el ejemplo. El muestreo SYSTEM de *nombre-tabla* generalmente se ejecuta con mayor rapidez que el muestreo BERNOULLI porque se tienen que recuperar menos páginas de datos; sin embargo, a menudo puede dar lugar a estimaciones menos

precisas para las funciones de agregación (SUM(SALES), por ejemplo), especialmente si las filas de *nombre-tabla* están en clúster en cualquier columna a la que se haga referencia en dicha consulta. En determinadas circunstancias, el optimizador puede decidir que es más eficiente realizar un muestreo SYSTEM como si fuera un muestreo BERNOULLI, por ejemplo cuando un índice puede aplicar un predicado en *nombre-tabla* y es mucho más selectivo que la tasa de muestreo P.

La *expresión-numérica-1* especifica el tamaño de la muestra que se debe obtener del *nombre-tabla*, expresado como porcentaje. Debe ser una expresión numérica constante que no puede contener columnas. La expresión se debe evaluar en un número positivo que sea inferior o igual a 100, pero puede ser entre 1 y 0. Por ejemplo, un valor 0,01 representa una centésima de un porcentaje, lo que significa que se tomará un muestreo de una fila entre 10.000 como promedio. Una *expresión-numérica-1* que se evalúe en 100 se maneja como si no se hubiera especificado la cláusula-tablesample. Si *expresión-numérica1* se evalúa en un valor nulo o en un valor mayor que 100 o menor que 0, se devuelve un error (SQLSTATE 2202H).

A veces resulta recomendable que el muestreo se repita de una ejecución de la consulta a la siguiente; por ejemplo, durante una prueba de regresión o "depuración" de la consulta. Esto se puede conseguir especificando la cláusula REPEATABLE. La cláusula REPEATABLE necesita la especificación de una *expresión-numérica2* entre paréntesis, que desempeña el mismo rol que el valor raíz de un generador de números aleatorios. La adición de la cláusula REPEATABLE a la cláusula-tablesample de cualquier *nombre-tabla* asegura que las ejecuciones repetidas de dicha consulta (utilizando el mismo valor para *expresión-numérica2*) devuelven el mismo ejemplo, dando por supuesto que los datos no se han actualizado, reorganizado ni reparticionado. Para garantizar que se utiliza el mismo ejemplo de *nombre-tabla* entre diversas consultas, se recomienza utilizar una tabla temporal global. Como alternativa, se pueden combinar varias consultas en una consulta, con varias referencias a un ejemplo definido mediante la cláusula WITH.

A continuación se muestran algunos ejemplos:

Ejemplo 1: Solicitar un ejemplo Bernoulli del 10% de la tabla Sales por motivos de auditoría.

```
SELECT * FROM Sales
  TABLESAMPLE BERNOULLI(10)
```

Ejemplo 2: Calculas los ingresos totales por ventas de la región Northeast para cada categoría de producto, utilizando un ejemplo aleatorio SYSTEM del 1% de la tabla Sales. La semántica de SUM corresponde a la propia muestra, de modo que para extrapolar las ventas a la tabla Sales completa, la consulta debe dividir ducha SUM por la tasa de muestreo (0,01).

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

Ejemplo 3: Utilizando la cláusula REPEATABLE, modificar la consulta anterior para asegurar que se obtiene el mismo resultado (aleatorio) cada vez que se ejecuta la consulta. (El valor de la constante especificada entre paréntesis es arbitrario.)

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1) REPEATABLE(3578231)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

Referencias a las funciones de tabla

En general, se puede hacer referencia a una función de tabla, junto a los valores de sus argumentos en la cláusula FROM de una sentencia SELECT, exactamente de la misma manera que una tabla o una vista. Sin embargo, se aplican algunas consideraciones especiales.

- Nombres de columna de función de tabla

  A menos que se proporcionen nombres de columna alternativos a continuación del *nombre-correlación*, los nombres de columna de la función de tabla son los especificados en la cláusula RETURNS de la sentencia CREATE FUNCTION. Es análogo a los nombres de las columnas de una tabla, que se definen en la sentencia CREATE TABLE.

- Resolución de una función de tabla

  Los argumentos especificados en una referencia de función de tabla, junto con el nombre de la función, se utilizan por un algoritmo llamado *resolución de función* para determinar la función exacta que se va a utilizar. Esta operación no es diferente de lo que ocurre con las demás funciones (por ejemplo, en las funciones escalares) utilizadas en una sentencia.

- Argumentos de función de tabla

  Como en los argumentos de funciones escalares, los argumentos de función de tabla pueden ser en general cualquier expresión SQL válida. Los siguientes ejemplos contienen sintaxis válida:

```
Ejemplo 1:  SELECT c1
            FROM TABLE( tf1('Zachary') ) AS z
            WHERE c2 = 'FLORIDA';

Ejemplo 2:  SELECT c1
            FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;

Ejemplo 3:  SELECT c1
            FROM t
            WHERE c2 IN
                     (SELECT c3 FROM
                      TABLE( tf5(t.c4) ) AS z  -- correlated reference
                     )                         -- a cláusula FROM ant.
```

- Funciones de tabla que modifican datos de SQL

  Las funciones de tabla que se especifican con la opción MODIFIES SQL DATA sólo se pueden utilizar como la última referencia de tabla de una *sentencia-select*, *expresión-tabla-común* o sentencia RETURN que sea una subselección, una función SELECT INTO o una *selección completa de fila* de una sentencia SET. Sólo se permite una función de tabla en una cláusula FROM y los argumentos de la función de tabla deben estar correlacionados con las demás referencias de tabla de la subselección (SQLSTATE 429BL). Los siguientes ejemplos contienen sintaxis válida para una función de tabla con la propiedad MODIFIES SQL DATA:

```
Ejemplo 1:  SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

Ejemplo 2:  SELECT c1
            FROM t1, t2, TABLE( tfmod(t1.c1, t2.c1) ) AS z

Ejemplo 3:  SET var =
            (SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z
```

```
Ejemplo 4:  RETURN SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

Ejemplo 5:  WITH v1(c1) AS
            (SELECT c1
            FROM TABLE( tfmod(:hostvar1) ) AS z)
            SELECT c1
            FROM v1, t1 WHERE v1.c1 = t1.c1
```

Expresión-tabla-anidada tolerante a errores

Determinados errores que se producen en una *expresión-tabla-anidada* pueden tolerarse y, en lugar de devolver u error, la consulta puede continuar y devolver un resultado.

Si se especifica la cláusula RETURN DATA UNTIL, las filas que se devuelvan de la selección completa antes de que se detecte la condición indicada formarán el conjunto de resultados de la selección completa. Esto significa que un conjunto de resultados parcial (que también puede ser un conjunto de resultados nulo) de la selección completa puede aceptarse como resultado para la *expresión-tabla-anidada*.

La palabra clave FEDERATED limita la condición para gestionar sólo los errores que se producen en una fuente de datos remota.

La condición puede especificarse como un valor SQLSTATE, con una longitud de *constante-serie* de 5. Si lo desea, puede especificar un valor SQLCODE para cada valor SQLSTATE especificado. Para las aplicaciones portátiles, especifique valores SQLSTATE siempre que pueda, ya que los valores SQLCODE generalmente no pueden utilizarse en todas las plataformas y no forman parte del estándar SQL.

Sólo pueden tolerarse determinados errores. Los errores que no permiten que se ejecute el resto de la consulta no pueden tolerarse y se de vuelve un error para toda la consulta. El *valor-condición-específica* puede especificar condiciones que en realidad el gestor de bases de datos no tolera, incluso en caso de especificarse un valor SQLSTATE o SQLCODE, y, en estos casos, se devuelve un error.

Si se especifican, es posible que el gestor de bases de datos tolere los valores SQLSTATE y SQLCODE siguientes:

- SQLSTATE 08001; SQLCODEs -1336, -30080, -30081, -30082
- SQLSTATE 08004
- SQLSTATE 42501
- SQLSTATE 42704; SQLCODE -204
- SQLSTATE 42720
- SQLSTATE 28000

Una consulta o una vista que tiene una *expresión-tabla-anidada* tolerante a errores es de sólo lectura.

La selección completa de una *expresión-tabla-anidada* tolerante a errores no se optimiza utilizando tablas de consulta materializada.

Referencias correlacionadas en referencias-tabla

Las referencias correlacionadas se pueden utilizar en expresiones de tabla anidadas o como argumentos para funciones de tabla. La norma básica que se aplica para

ambos casos es que la referencia correlacionada debe ser de una *referencia-tabla* de un nivel superior en la jerarquía de subconsultas. Esta jerarquía incluye las referencias-tabla que ya se han resuelto en el proceso de izquierda a derecha de la cláusula FROM. Para las expresiones de tabla anidadas, la palabra clave LATERAL debe aparecer antes de la selección completa. Por lo tanto, los ejemplos siguientes son de sintaxis válida:

```
Ejemplo 1:  SELECT t.c1, z.c5
            FROM t, TABLE( tf3(t.c2) ) AS z      -- t precede tf3
            WHERE t.c3 = z.c4;                   -- en FROM, por lo que t.c2
                                                 -- se conoce

Ejemplo 2:  SELECT t.c1, z.c5
            FROM t, TABLE( tf4(2 * t.c2) ) AS z  -- t precedes tf4
            WHERE t.c3 = z.c4;                   -- en FROM, por lo que t.c2
                                                 -- se conoce

Ejemplo 3:  SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 LATERAL (SELECT AVG(e.salary) AS avgsal,
                                 COUNT(*) AS empcount
                          FROM employee e           -- departamento precede
                          WHERE e.workdept=d.deptno -- y TABLE se ha
                         ) AS empinfo;              -- especificado, por lo que
                                                    -- d.deptno se conoce
```

Pero los ejemplos siguientes no son válidos:

```
Ejemplo 4:  SELECT t.c1, z.c5
            FROM TABLE( tf6(t.c2) ) AS z, t  -- no se puede resolver t en t.c2!
            WHERE t.c3 = z.c4;                  -- compárese con el Ejemplo 1.

Ejemplo 5:  SELECT a.c1, b.c5
            FROM TABLE( tf7a(b.c2) ) AS a, TABLE( tf7b(a.c6) ) AS b
            WHERE a.c3 = b.c4;                  -- no se puede resolver b en
                                                -- b.c2!

Ejemplo 6:  SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 (SELECT AVG(e.salary) AS avgsal,
                                 COUNT(*) AS empcount
                          FROM employee e           -- departamento precede
                          WHERE e.workdept=d.deptno -- pero TABLE no se ha
                         ) AS empinfo;              -- especificado, por lo que
                                                    -- d.deptno no se conoce
```

Referencias de tabla de cambio de datos

Una cláusula *referencia-tabla-cambio-datos* especifica una tabla de resultados intermedia. Esta tabla se basa en las filas que cambia directamente la sentencia UPDATE buscada, DELETE buscada o INSERT que se incluye en la cláusula. Una *referencia-tabla-cambio-datos* se puede especificar como la única *referencia-tabla* de la cláusula FROM de la selección completa externa que se utiliza en una *sentencia-select*, una sentencia SELECT INTO o una expresión de tabla común. También se puede especificar una *referencia-tabla-cambio-datos* como la única referencia de tabla en la única selección completa de una sentencia SET Variable (SQLSTATE 428FL). Se considera que la tabla o vista de destino de la sentencia de cambio de datos es una tabla o vista a la que se hace referencia en la consulta; por lo tanto, el ID de autorización de la consulta debe tener privilegio SELECT sobre la tabla o vista de destino. Una cláusula *referencia-tabla-cambio-datos* no puede

especificarse en una definición de vista, en una definición de tabla de consultas materializadas ni en una sentencia FOR (SQLSTATE 428FL).

El destino de la sentencia UPDATE, DELETE o INSERT no puede ser una vista temporal definida en una expresión de tabla común (SQLSTATE 42807).

FINAL TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas que cambia la sentencia de cambio de datos de SQL tal como aparecen al final de la sentencia de cambio de datos. Si hay activadores AFTER o restricciones de referencia que dan lugar a más operaciones sobre la tabla que es el destino de la sentencia de cambio de datos de SQL, se devuelve un error (SQLSTATE 560C6). Si el destino de la sentencia de cambio de datos de SQL es una vista que está definida con un activador INSTEAD OF para el tipo de cambio de datos, se devuelve un error (SQLSTATE 428G3).

NEW TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas que cambia la sentencia de cambio de datos de SQL antes de la aplicación de restricciones de referencia y de activadores AFTER. Es posible que los datos de la tabla de destino al final de la sentencia no coincidan con los datos de la tabla de resultados intermedia debido al proceso adicional de restricciones de referencia y activadores AFTER.

OLD TABLE
Especifica que las filas de la tabla de resultados intermedia representan el conjunto de filas cambiadas por la sentencia de cambio de datos de SQL tal como aparecían antes de la aplicación de la sentencia de cambio de datos.

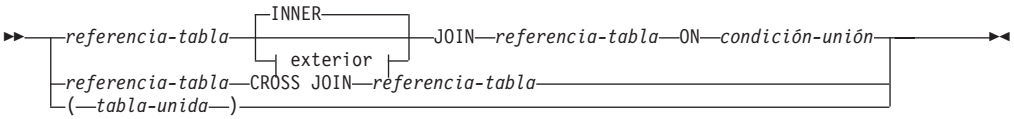
(sentencia-update-buscada)
Especifica una sentencia UPDATE buscada. Una cláusula WHERE o una cláusula SET en la sentencia UPDATE no puede contener referencias correlacionadas a columnas fuera de la sentencia UPDATE.

(sentencia-delete-buscada)
Especifica una sentencia DELETE buscada. Una cláusula WHERE en la sentencia DELETE no puede contener referencias correlacionadas a columnas fuera de la sentencia DELETE.

(sentencia-insert)
Especifica una sentencia INSERT. Una selección completa en la sentencia INSERT no puede contener referencias correlacionadas a columnas fuera de la selección completa de la sentencia INSERT.

El contenido de la tabla de resultados intermedia correspondiente a *referencia-tabla-cambio-datos* se determina cuando se abre el cursor. La tabla de resultados intermedia contiene todas las filas manipuladas, incluidas todas las columnas de la tabla o vista de destino especificada. Todas las columnas de la tabla o vista de destino correspondiente a una sentencia de cambio de datos SQL resultan accesibles utilizando los nombres de columnas de la tabla o vista de destino. Si se ha especificado una cláusula INCLUDE dentro de una sentencia de cambio de datos, la tabla de resultados intermedia contendrá estas columnas adicionales.

tabla-unida



unión externa:

Una *tabla unida* especifica una tabla resultante intermedia que es el resultado de una unión interna o una unión externa. La tabla se deriva aplicando uno de los operadores de unión: CROSS, INNER, LEFT OUTER, RIGHT OUTER o FULL OUTER a los operandos.

Las uniones cruzadas representan el producto cruzado de las tablas, donde cada fila de la tabla izquierda se combina con cada fila de la tabla derecha. Las uniones internas se pueden considerar el producto cruzado de las tablas, conservando sólo las filas donde la condición de unión es verdadera. Es posible que en la tabla de resultados falten filas de una o ambas tablas unidas. Las uniones externas incluyen la unión interna y conservan las filas que faltan. Hay tres tipos de uniones externas:

- ***Unión externa izquierda*** incluye las filas de la tabla de la izquierda que faltaban en la unión interna.
- ***Unión externa derecha*** incluye las filas de la tabla de la derecha que faltaban en la unión interna.
- ***Unión externa completa*** incluye las filas de las tabla de la izquierda y de la derecha que faltaban en la unión interna.

Si no se especifica ningún operador-unión, el implícito es INNER. El orden en que se realizan múltiples uniones puede afectar al resultado. Las uniones pueden estar anidadas en otras uniones. El orden del proceso de uniones es generalmente de izquierda a derecha, pero se basa en la posición de la condición-unión necesaria. Es aconsejable utilizar paréntesis para que se pueda leer mejor el orden de las uniones anidadas. Por ejemplo:

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
  RIGHT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
    ON TB1.C1=TB3.C1
```

es igual a:

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
  RIGHT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
    ON TB1.C1=TB3.C1
```

Una tabla unida se puede utilizar en cualquier contexto en el que se utilice cualquier forma de la sentencia SELECT. Una vista o un cursor es de sólo lectura si su sentencia SELECT incluye una tabla unida.

Una *condición-unión* es una *condición-búsqueda*, excepto que:

- 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
- Las columnas a las que se hace referencia en una expresión de *condición-unión* deben ser columnas de las tablas de operandos de la unión asociada (en el ámbito de la misma cláusula tabla-unida)
- Las funciones a las que se haga referencia en una expresión de *condición-unión* de una unión externa completa deben ser deterministas y no deben tener acciones externas
- No puede incluir una expresión XMLQUERY ni XMLEXISTS

Se produce un error si la condición de unión no cumple estas normas (SQLSTATE 42972).

Las referencias a columnas se resuelven utilizando las normas para la resolución de calificadores de nombres de columna. Las mismas normas que se aplican a los predicados se aplican a las condiciones de unión.

Operaciones de unión

Una *condición-unión* especifica emparejamientos de T1 y T2, donde T1 y T2 son tablas de los operandos izquierdo y derecho del operador JOIN de la *condición-unión*. En todas las combinaciones posibles de filas de T1 y T2, una fila de T1 se empareja con una fila de T2 si la *condición-unión* es verdadera. Cuando una fila de T1 se une con una fila de T2, una fila del resultado consta de los valores de dicha fila de T1 concatenada con los valores de dicha fila de T2. La ejecución puede implicar la generación de una fila nula. La fila nula de una tabla consta de un valor nulo para cada columna de la tabla, sin tener en cuenta si las columnas permiten valores nulos.

A continuación encontrará un resumen del resultado de las operaciones de unión:

- El resultado de T1 CROSS JOIN T2 consta de todos los emparejamientos posibles de las filas.
- El resultado de T1 INNER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera.
- El resultado de T1 LEFT OUTER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera y, para cada fila no emparejada de T1, la concatenación de dicha fila con la fila nula de T2. Todas las columnas derivadas de T2 permiten valores nulos.
- El resultado de T1 RIGHT OUTER JOIN T2 consta de sus filas emparejadas cuando la condición-unión es verdadera y, para cada fila de T2 no emparejada, la concatenación de dicha fila con la fila nula de T1. Todas las columnas derivadas de T1 permiten valores nulos.
- El resultado de T1 FULL OUTER JOIN T2 consta de sus filas emparejadas y, para cada fila de T2 no emparejada, la concatenación de dicha fila con la fila nula de T1 y, para cada fila de T1 no emparejada, la concatenación de dicha fila con la fila nula de T2. Todas las columnas derivadas de T1 y T2 permiten valores nulos.

cláusula-where

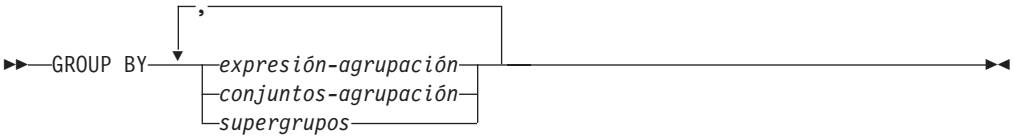
La cláusula WHERE especifica una tabla resultante intermedia que consta de aquellas filas de R para las que se cumple la *condición-búsqueda*. R es el resultado de la cláusula FROM de la subselección.

La *condición-búsqueda* debe ajustarse a las normas siguientes:

- Cada *nombre-columna* debe identificar sin ambigüedades una columna de R o ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.
- No debe especificarse una función agregada a menos que se especifique la cláusula WHERE en una subconsulta de una cláusula HAVING y el argumento de la función sea una referencia correlacionada para un grupo.

Cualquier subconsulta de *condición-búsqueda* se ejecuta de forma efectiva para cada fila de R y los resultados se utilizan en la aplicación de la *condición-búsqueda* en la fila dada de R. Una subconsulta sólo se ejecuta en realidad para cada fila de R si incluye una referencia correlacionada. De hecho, una subconsulta sin referencias correlacionadas sólo se puede ejecutar una vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila.

cláusula-group-by



La cláusula GROUP BY especifica una tabla intermedia de resultados que está formada por una agrupación de las filas de R, que es el resultado de la cláusula anterior de la subselección.

En su forma más simple, una cláusula GROUP BY contiene una *expresión de agrupación*. Una expresión de agrupación es una *expresión* que se utiliza al definir la agrupación de R. Cada expresión o *nombre de columna* incluido en la expresión-agrupación debe identificar sin ambigüedades una columna de R (SQLSTATE 42702 o 42703). Una expresión de agrupación no puede incluir una selección completa escalar ni una expresión XMLQUERY o XMLEXISTS (SQLSTATE 42822), ni ninguna expresión o función que sea no determinista o tenga una acción externa (SQLSTATE 42845).

Nota: Las expresiones siguientes, que no contienen una referencia a columna explícita, se pueden utilizar en una *expresión-agrupación* para identificar una columna de R:

- ROW CHANGE TIMESTAMP FOR *designador-tabla*
- ROW CHANGE TOKEN FOR *designador-tabla*
- Función escalar RID_BIT o RID

Las formas más complejas de la cláusula GROUP BY son los *conjuntos-agrupación* y los *supergrupos*. Para ver una descripción de estas formas, consulte los apartados "conjuntos-agrupación" y "supergrupos" en la página 744, respectivamente.

El resultado de GROUP BY es un conjunto de grupos de filas. Cada fila del resultado representa el conjunto de filas para el que la *expresión-agrupación* es igual. En la agrupación, todos los valores nulos de una *expresión-agrupación* se consideran iguales.

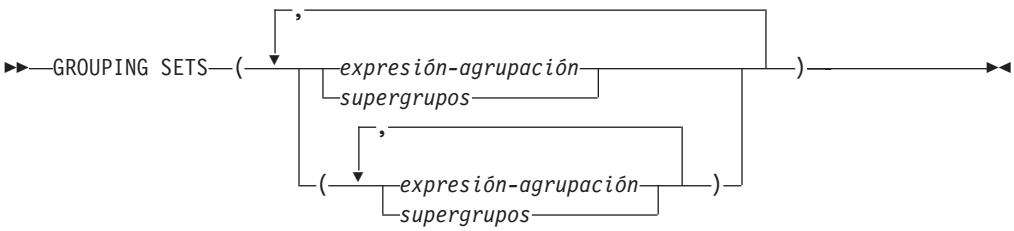
Si una *expresión-agrupación* contiene columnas de coma flotante decimal y hay varias representaciones del mismo número en dichas columnas, el número que se devuelve puede ser cualquiera de las representaciones de dicho número.

Una *expresión-agrupación* se puede utilizar en una condición de búsqueda de una cláusula HAVING, en una expresión de una cláusula SELECT o en una *expresión-clave-clasificación* de una cláusula ORDER BY (consulte el apartado "cláusula-order-by" en la página 749 para ver los detalles). En cada caso, la referencia sólo especifica un valor para cada grupo. Por ejemplo, si la *expresión-agrupación* es *col1+col2*, una expresión permitida en la lista de selección sería *col1+col2+3*. Las normas de asociación para expresiones rechazarían la expresión parecida, *3+col1+col2*, a menos que se utilicen paréntesis para asegurar que la expresión correspondiente se evalúe en el mismo orden. Por lo tanto, *3+(col1+col2)* también se permitiría en la lista de selección. Si se utiliza el operador de concatenación, la *expresión-agrupación* debe utilizarse exactamente como se ha especificado la expresión en la lista de selección.

Si la *expresión-agrupación* contiene series de longitud variable con blancos de cola, los valores del grupo pueden diferir en el número de blancos de cola y puede que no todos tengan la misma longitud. En dicho caso, la referencia a la *expresión-agrupación* continúa especificando sólo un valor para cada grupo, pero el valor para un grupo se elige arbitrariamente entre el conjunto de valores disponibles. Por lo tanto, la longitud real del valor del resultado es imprevisible.

Tal como se ha apuntado, existen casos en los que la cláusula GROUP BY no puede hacer referencia directamente a una columna que esté especificada en la cláusula SELECT como una expresión (selección completa-escalar, no determinista o funciones de acción externa). Para agrupar utilizando una expresión como ésta, utilice una expresión de tabla anidada o una expresión de tabla común para proporcionar primero una tabla resultante con la expresión como una columna del resultado. Para ver un ejemplo de la utilización de expresiones de tabla anidadas, consulte el Ejemplo A9.

conjuntos-agrupación



Una especificación de *conjuntos-agrupación* permite especificar múltiples cláusulas de agrupación en una sola sentencia. Se puede decir que es la unión de dos o más

grupos de filas en un solo conjunto resultante. Es lógicamente equivalente a la unión de múltiples subselecciones con la cláusula group by en cada subselección correspondiente a un conjunto de agrupación. Un conjunto de agrupación puede ser un solo elemento o puede ser una lista de elementos delimitados por paréntesis, donde un elemento es una expresión-agrupación o un supergrupo. La utilización de *conjuntos-agrupación* permite calcular los grupos con una sola pasada por la tabla base.

La especificación de *conjuntos-agrupación* permite utilizar una *expresión-agrupación* simple o las formas más complejas de *supergrupos*. Para ver una descripción de *supergrupos*, consulte el apartado "supergrupos".

Tenga en cuenta que los conjuntos de agrupaciones son los bloques fundamentales para la creación de operaciones GROUP BY. Una operación GROUP BY simple con una sola columna puede considerarse un conjunto de agrupación con un elemento. Por ejemplo:

```
GROUP BY a
```

es igual a

```
GROUP BY GROUPING SETS((a))
```

y

```
GROUP BY a,b,c
```

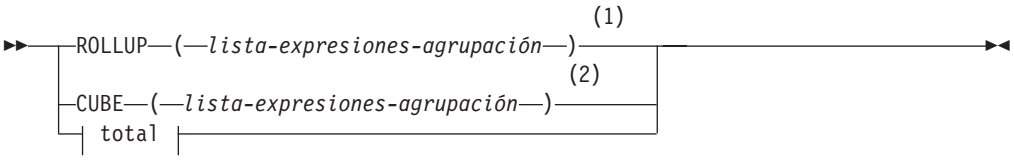
es igual a

```
GROUP BY GROUPING SETS((a,b,c))
```

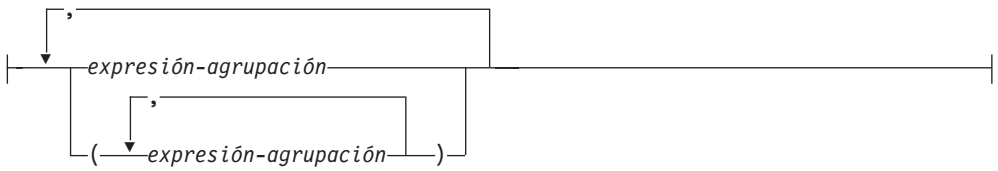
Las columnas de no agregación de la lista de selección de la subselección que se excluyen de un conjunto de agrupación devolverán un nulo para dichas columnas para cada fila generada para dicho conjunto de agrupación. Esto refleja el hecho que la agregación se ha realizado sin tener en cuenta los valores para dichas columnas.

Desde el Ejemplo C2 hasta el Ejemplo C7 se ilustra la utilización de los conjuntos de agrupaciones.

supergrupos



lista-expresiones-agrupación:



total:

Notas:

1 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH ROLLUP.

2 Una especificación alternativa cuando se utiliza sola en la cláusula Group By es: lista-expresiones-agrupación WITH CUBE.

ROLLUP (*lista-expresiones-agrupación* **)**

Una *agrupación ROLLUP* es una extensión de la cláusula GROUP BY que produce un conjunto resultante que contiene filas de *subtotales* además de las filas agrupadas "normales". Las filas de *subtotales* son filas "superagregadas" que contienen más agregados cuyos valores se obtienen al aplicar las mismas funciones agregadas que se han utilizado para obtener las filas agrupadas. Estas filas se denominan filas de subtotales, porque ese es su uso más habitual; sin embargo, se puede utilizar cualquier función agregada para la agregación. Por ejemplo, MAX y AVG se utilizan en el Ejemplo C8. La función agregada GROUPING puede utilizarse para indicar si una fila la ha generado unsupergrupo.

Una agrupación ROLLUP es una serie de *conjuntos-agrupación*. La especificación general de ROLLUP con *n* elementos

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

es equivalente a

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn)
                       (C1,C2,...,Cn-1)
                       ...
                       (C1,C2)
                       (C1)
                       () )
```

Observe que los *n* elementos de ROLLUP se convierten en *n*+1 conjuntos de agrupaciones. Tenga en cuenta también que el orden en el que se especifican las *expresiones-agrupación* es importante para ROLLUP. Por ejemplo:

```
GROUP BY ROLLUP(a,b)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b)
                       (a)
                       () )
```

mientras que

```
GROUP BY ROLLUP(b,a)
```

es igual a

```
GROUP BY GROUPING SETS((b,a)
                       (b)
                       () )
```

La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El Ejemplo C3 ilustra la utilización de ROLLUP.

CUBE (*lista-expresiones-agrupación* **)**

Una *agrupación CUBE* es una extensión a la cláusula GROUP BY que produce

un conjunto resultante que contiene todas las filas de la agregación ROLLUP y, además, contiene filas de "tabulación cruzada". Las filas de *tabulación cruzada* son filas "superagregadas" adicionales que no forman parte de una agregación con subtotales. La función agregada GROUPING puede utilizarse para indicar si una fila la ha generado unsupergrupo.

Igual que ROLLUP, una agrupación CUBE también puede decirse que es una serie de *conjuntos-agrupación*. En el caso de CUBE, todas las permutaciones de la *lista-expresiones-agregación* al cubo se calcula junto con el total. Por lo tanto, los *n* elementos de CUBE se convierten en 2**n* (2 elevado a la potencia *n*) *conjuntos-agrupación*. Por ejemplo, una especificación de:

```
GROUP BY CUBE(a,b,c)
```

es equivalente a:

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (b,c)
                       (a)
                       (b)
                       (c)
                       () )
```

Observe que los tres elementos de CUBE se convierten en ocho conjuntos de agrupaciones.

El orden de especificación de los elementos no importa para CUBE.'CUBE (DayOfYear, Sales_Person)' y 'CUBE (Sales_Person, DayOfYear)' dan los mismos conjuntos del resultado. La utilización de la palabra 'mismos' se aplica al contenido del conjunto resultante, no a su orden. La cláusula ORDER BY es la única manera de garantizar el orden de las filas en el conjunto resultante. El Ejemplo C4 ilustra la utilización de CUBE.

lista-expresiones-agrupación
Una *lista-expresiones-agrupación* se utiliza en la cláusula CUBE o ROLLUP para definir el número de elementos de la operación CUBE o ROLLUP. Se controla utilizando los paréntesis para delimitar los elementos con múltiples *expresiones-agrupación*.

Las normas para la *expresión-agrupación* se describen en el apartado "cláusula-group-by" en la página 742. Por ejemplo, supongamos que una consulta tiene que devolver los gastos totales para ROLLUP de City dentro de una Province pero no de un County. Sin embargo, la cláusula:

```
GROUP BY ROLLUP(Province, County, City)
```

da como resultado filas de subtotales que no se desean para County. En la cláusula:

```
GROUP BY ROLLUP(Province, (County, City))
```

el compuesto (County, City) forma un elemento de ROLLUP y, por lo tanto, una consulta que utiliza esta cláusula dará el resultado deseado. En otras palabras, ROLLUP de dos elementos:

```
GROUP BY ROLLUP(Province, (County, City))
```

genera:

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province)
                       () )
```

y el ROLLUP de tres elementos genera:

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province, County)
                       (Province)
                       () )
```

El Ejemplo C2 también utiliza valores de columna compuestos.

total
Tanto CUBE como ROLLUP devuelven una fila que es la agregación global (total). Esto se puede especificar por separado mediante paréntesis vacíos dentro de la cláusula GROUPING SET. También puede especificarse directamente en la cláusula GROUP BY, aunque no surte ningún efecto en el resultado de la consulta. El Ejemplo C4 utiliza la sintaxis del total.

Combinación de conjuntos de agrupaciones

Se puede utilizar para combinar cualquier tipo de cláusula GROUP BY. Cuando se combinan los campos de una *expresión-agrupación* simple con otros grupos, se "añaden" al principio de los *conjuntos de agrupaciones* resultantes. Cuando se combinan las expresiones ROLLUP o CUBE, funcionan como "multiplicadores" en el resto de la expresión, formando entradas de un conjunto de agrupación adicional según la definición de ROLLUP o CUBE.

Por ejemplo, la combinación de elementos de *expresión-agrupación* actúa de la siguiente manera:

```
GROUP BY a, ROLLUP(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a) )
```

O de manera parecida,

```
GROUP BY a, b, ROLLUP(c,d)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

La combinación de elementos de *ROLLUP* actúa de la siguiente manera:

```
GROUP BY ROLLUP(a), ROLLUP(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

De manera similar,

```
GROUP BY ROLLUP(a), CUBE(b,c)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
                       (b,c)
                       (b)
                       (c)
                       () )
```

La combinación de elementos de *CUBE* y de *ROLLUP* actúa de la siguiente manera:

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b)
                       (a,c,d)
                       (a,c)
                       (a)
                       (b,c,d)
                       (b,c)
                       (b)
                       (c,d)
                       (c)
                       () )
```

Igual que una *expresión-agrupación* simple, la combinación de conjuntos de agrupaciones elimina también los duplicados dentro de cada conjunto de agrupación. Por ejemplo,

```
GROUP BY a, ROLLUP(a,b)
```

es equivalente a

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

Un ejemplo más completo de la combinación de conjuntos de agrupaciones es crear un conjunto resultante que elimine ciertas filas que se devolverían para una agregación CUBE completa.

Por ejemplo, considere la siguiente cláusula GROUP BY:

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

La columna listada inmediatamente a la derecha de GROUP BY está agrupada simplemente, las que están entre paréntesis a continuación de ROLLUP se han avanzado y las que están entre paréntesis a continuación de CUBE se han elevado al cubo. Por lo tanto, la cláusula anterior da como resultado el cubo de MONTH en YEAR que después avanza en WEEK en Sales_Person en la agregación Region. No da como resultado una fila del total ni ninguna fila de tabulación cruzada en Region, Sales_Person o WEEK(Sales_Date), por lo que produce menos filas que la cláusula siguiente:

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                 YEAR(Sales_Date), MONTH(Sales_Date) )
```

cláusula-having

La cláusula HAVING especifica una tabla resultante intermedia que consta de aquellos grupos de R para los que se cumple la *condición-búsqueda*. R es el resultado de la cláusula anterior de la subselección. Si esta cláusula no es GROUP BY, R se considera un solo grupo sin columnas de agrupación.

Cada *nombre-columna* de la condición de búsqueda debe realizar una de las acciones siguientes:

- Identificar sin ambigüedades una columna de agrupación de R.
- Debe especificarse en una función agregada.
- Ser una referencia correlacionada. Un *nombre-columna* es una referencia correlacionada si identifica una columna de una *referencia-tabla* en una subselección externa.

Un grupo de R al que se aplica la condición de búsqueda suministra el argumento para cada función agregada en la condición de búsqueda, excepto para cualquier función cuyo argumento sea una referencia correlacionada.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que la subconsulta se ejecuta cada vez que se aplica la condición de búsqueda a un grupo de R y los resultados se utilizan en la aplicación de la condición de búsqueda. En realidad, la subconsulta sólo se ejecuta para cada grupo si contiene una referencia correlacionada. Para ver una ilustración de las diferencias, consulte el Ejemplo A6 y el Ejemplo A7.

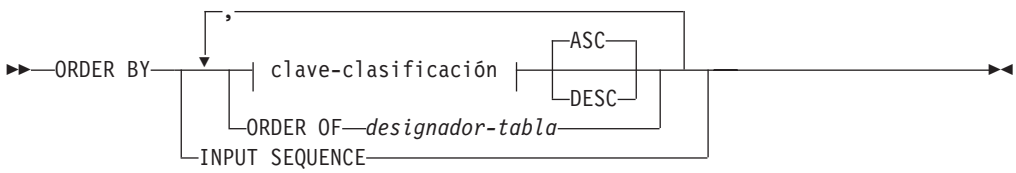
Una referencia correlacionada a un grupo de R debe identificar una columna de agrupación o estar contenida en una función agregada.

Cuando se utiliza HAVING sin GROUP BY, la lista de selección sólo puede incluir nombres de columnas cuando son argumentos de una función agregada, referencias de columna correlacionada, variables globales, variables del lenguaje principal, literales, registros especiales, variables de SQL o parámetros de SQL.

Nota: Las expresiones siguientes sólo se pueden especificar en una cláusula HAVING si están contenidas en una función agregada (SQLSTATE 42803):

- ROW CHANGE TIMESTAMP FOR *designador-tabla*
- ROW CHANGE TOKEN FOR *designador-tabla*
- Función escalar RID_BIT o RID

cláusula-order-by



clave-clasificación:

La cláusula ORDER BY especifica una ordenación de las filas de la tabla resultante. Si se identifica una especificación de clasificación individual (una *clave-clasificación* con una dirección asociada), las filas se ordenan por los valores de dicha especificación de clasificación. Si se indica más de una especificación de clasificación, las filas se ordenan por los valores de la primera especificación de clasificación identificada, después por los valores de la segunda especificación de clasificación identificada, y así sucesivamente. Ninguna *clave-clasificación* puede tener un tipo de datos CLOB, DBCLOB, BLOB,, un tipo diferenciado de cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42907).

Una columna con nombre de la lista de selección se puede identificar mediante una *clave-clasificación* que sea un *entero-simple* o un *nombre-columna-simple*. Una columna sin nombre de la lista de selección debe identificarse por un *entero-simple* o, en algunos casos, por una *expresión-clave-clasificación* que coincida con la expresión de la lista de selección (consulte los detalles de *expresión-clave-clasificación*). Una columna no tiene nombre si no se especifica la cláusula AS y se obtiene a partir de una constante, una expresión con operadores o una función.

La ordenación se realiza de acuerdo con las normas de comparación. Si una cláusula ORDER BY contiene columnas de coma flotante decimal y hay varias representaciones del mismo número en dichas columnas, el orden de las diversas representaciones del mismo número no está especificado. El valor nulo es superior a cualquier otro valor. Si la cláusula ORDER BY no ordena por completo las filas, se visualizan las filas con valores duplicados de todas las columnas identificadas en un orden arbitrario.

nombre-columna-simple
Normalmente identifica una columna de la tabla resultante. En este caso, *nombre-columna-simple* debe ser el nombre de columna de una columna con nombre de la lista de selección.

El *nombre-columna-simple* también puede identificar un nombre de columna de una tabla, vista o tabla anidada identificada en la cláusula FROM si la consulta es una subselección. Esto incluye columnas definidas como ocultas implícitamente. Se produce un error si la subselección:

- Especifica DISTINCT en la cláusula de selección (SQLSTATE 42822)
- Genera un resultado agrupado y el *nombre-columna-simple* no es una *expresión-agrupación* (SQLSTATE 42803).

La determinación de qué columna se utiliza para ordenar el resultado se describe más abajo en el apartado “Nombre de las columnas en claves de clasificación”.

entero-simple
Debe ser mayor que 0 y no ser superior al número de columnas de la tabla resultante (SQLSTATE 42805). El entero *n* identifica la columna *n* de la tabla resultante.

expresión-clave-clasificación
Una expresión que no es simplemente un nombre de columna ni una constante de enteros sin signo. La consulta a la que se aplica la ordenación debe ser una

subselección para utilizar esta forma de clave-clasificación. La *expresión-clave-clasificación* no puede incluir una selección completa escalar correlacionada (SQLSTATE 42703), ni una expresión XMLQUERY o XMLEXISTS (SQLSTATE 42822), ni ninguna función con una acción externa (SQLSTATE 42845).

Cualquier *nombre-columna* de una *expresión-clave-clasificación* debe ajustarse a las normas descritas bajo "Nombre de las columnas en claves de clasificación".

Existen unos cuantos casos especiales que restringen más las expresiones que se pueden especificar.

- DISTINCT se especifica en la cláusula SELECT de la subselección (SQLSTATE 42822).

  La expresión-clave-clasificación debe coincidir exactamente con una expresión de la lista de selección de la subselección (las selecciones completas-escalares nunca se emparejan).
- La subselección está agrupada (SQLSTATE 42803).

  La expresión-clave-clasificación puede:
  - ser una expresión en la lista de selección de la subselección,
  - incluir una *expresión-agrupación* de la cláusula GROUP BY de la subselección
  - incluir una función agregada, una constante o una variable del lenguaje principal.

ASC
Utiliza los valores de la columna en orden ascendente. Es el valor por omisión.

DESC
Utiliza los valores de la columna en orden descendente.

ORDER OF *designador-tabla*
Especifica que debe aplicarse el mismo orden utilizado en *diseñador-tabla* a la tabla resultante de la subselección. Debe haber una referencia de tabla que se corresponda con *diseñador-tabla* en la cláusula FROM de la subselección que especifica esta cláusula (SQLSTATE 42703). La subselección (o selección completa) correspondiente al *diseñador-tabla* especificado debe incluir una cláusula ORDER BY que dependa de los datos (SQLSTATE 428FI). El orden que se aplica es el mismo que si las columnas de la cláusula ORDER BY de la subselección anidada (o selección completa) se incluyeran en la subselección exterior (o selección completa) y estas columnas se especificaran en lugar de la cláusula ORDER OF.

Observe que este formato no se permite en una selección completa(excepto para el formato degenerativo de una selección completa). Por ejemplo, el ejemplo siguiente no es válido:

```
(SELECT C1 FROM T1
   ORDER BY C1)
UNION
SELECT C1 FROM T2
   ORDER BY ORDER OF T1
```

El ejemplo siguiente *sí* es válido:

```
SELECT C1 FROM
   (SELECT C1 FROM T1
      UNION
    SELECT C1 FROM T2
    ORDER BY C1 ) AS UTABLE
ORDER BY ORDER OF UTABLE
```

INPUT SEQUENCE
Especifica que, para una sentencia INSERT, la tabla de resultados reflejará el orden de entrada de filas de datos ordenadas. El orden INPUT SEQUENCE sólo se puede especificar si se utiliza una sentencia INSERT en una cláusula FROM (SQLSTATE 428G4). Consulte el apartado "referencia-tabla" en la página 730. Si se especifica INPUT SEQUENCE y los datos de entrada no están ordenados, la cláusula INPUT SEQUENCE se pasa por alto.

Notas

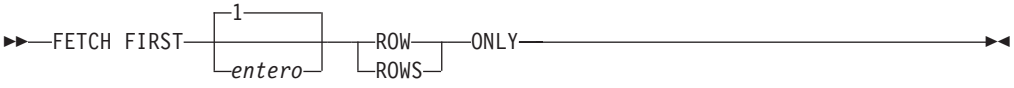
- **Nombres de columna en claves de clasificación**:
  - El nombre de columna está calificado.

    La consulta debe ser una *subselección* (SQLSTATE 42877). El nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección(SQLSTATE 42702). El valor de la columna se utiliza para calcular el valor de la especificación de clasificación.
  - El nombre de columna no está calificado.
    - La consulta es una subselección.

      Si el nombre de columna es idéntico al nombre de más de una columna de la tabla resultante, el nombre de columna debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la subselección de orden (SQLSTATE 42702). Si el nombre de columna es idéntico a una columna, dicha columna se utiliza para calcular el valor de la especificación de clasificación. Si el nombre de columna no es idéntico a una columna de la tabla resultante, debe identificar sin ambigüedades una columna de alguna tabla, vista o tabla anidada en la cláusula FROM de la selección completa de la sentencia-select (SQLSTATE 42702).
    - La consulta no es una subselección (incluye operaciones de conjuntos como la unión, excepción o intersección).

      El nombre de columna no debe ser idéntico al nombre de más de una columna de la tabla resultante (SQLSTATE 42702). El nombre de columna debe ser idéntico a exactamente una columna de la tabla resultante (SQLSTATE 42707) y esta columna se utiliza para calcular el valor de la especificación de clasificación.
- **Límites**: La utilización de una *expresión-clave-clasificación* o un *nombre-columna-simple* donde la columna no está en la lista de selección puede dar como resultado la adición de la columna o expresión a la tabla temporal utilizada para clasificación. Esto puede dar como resultado que se alcance el límite del número de columnas de una tabla o el límite en el tamaño de una fila de una tabla. Si se exceden estos límites se producirá un error si es necesaria una tabla temporal para realizar la operación de clasificación.

cláusula-fetch-first



La *cláusula-fetch-first* establece el número máximo de filas que pueden recuperarse. Indica al gestor de bases de datos que la aplicación no recuperará más de *entero* filas, cualquiera que sea el número de filas que pueda haber en la tabla resultante cuando no se especifica esta cláusula. Cualquier intento de recuperar más filas que

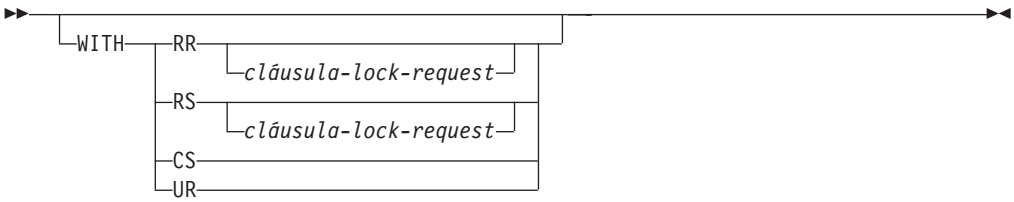
el número indicado por *entero* se trata de la misma manera que un fin de datos normal (SQLSTATE 02000). El valor de *entero* debe ser un entero positivo, distinto de cero.

Utilización de *cláusula-fetch-first* influye en la optimización de consultas de subselección o selección completa, basándose en el hecho de que como máximo se recuperarán *entero* filas. Si se especifica tanto la cláusula-fetch-first en la selección completa más exterior como la cláusula-optimize-for para la sentencia SELECT, el gestor de la base de datos utilizará el *entero* de la cláusula-optimize-for para influir en la optimización de consultas de la selección completa más exterior.

Limitar la tabla resultante a las primeras *entero* filas puede mejorar el rendimiento. El gestor de bases de datos detendrá el proceso de la consulta una vez que haya determinado las *entero* primeras filas. Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utiliza el valor *entero* más bajo de estas cláusulas para determinar el tamaño del almacenamiento intermedio de comunicaciones.

Si la selección completa contiene una sentencia de cambio de datos de SQL en la cláusula FROM, todas las filas se modifican, independientemente del límite del número de filas que hay que recuperar.

cláusula-isolation



La *cláusula-isolation* opcional especifica el nivel de aislamiento en el que se ejecuta la subselección o la selección completa y si debe obtenerse un tipo de bloqueo determinado.

- RR - Lectura repetible
- RS - Estabilidad de lectura
- CS - Estabilidad del cursor
- UR - Lectura no confirmada

cláusula-lock-request

La *cláusula-lock-request* se aplica únicamente a las consultas y a las operaciones de lectura de posicionamiento de una operación de inserción, actualización o supresión. Las operaciones de inserción, actualización y supresión se ejecutarán utilizando el bloqueo que determine el gestor de bases de datos.

La *cláusula-lock-request*, opcional, especifica el tipo de bloqueo que el gestor de bases de datos debe conseguir y retener:

SHARE
Los procesos simultáneos pueden conseguir bloqueos SHARE o UPDATE sobre los datos.

UPDATE
Los procesos simultáneos pueden conseguir bloqueos SHARE sobre los datos pero ningún proceso simultáneo puede conseguir un bloqueo UPDATE o EXCLUSIVE.

EXCLUSIVE
Los procesos simultáneos no pueden conseguir un bloqueo sobre los datos.

Restricciones de la *cláusula-isolation*:

- No se da soporte a la *cláusula-isolation* para una sentencia CREATE TABLE, CREATE VIEW o ALTER TABLE (SQLSTATE 42601).
- La *cláusula-isolation* no puede especificarse para una subselección o selección completa que ocasione la invocación de activadores, exploraciones de integridad de referencia o el mantenimiento de MQT (SQLSTATE 42601).
- Una subselección o una selección completa no puede incluir una *cláusula-lock-request* si esa subselección o selección completa hace referencia a funciones SQL que no se han declarado con la opción INHERIT ISOLATION LEVEL WITH LOCK REQUEST (SQLSTATE 42601).
- Una subselección o una selección completa que contenga una *cláusula-lock-request* no son candidatas al direccionamiento de MQT.
- Si se especifica una *cláusula-isolation* para *subselect* (subselección) o *fullselect* (selección completa) en el cuerpo de una función SQL, método SQL o activador, la cláusula se ignora y se devuelve un mensaje.
- Si se especifica una *cláusula-isolation* para una *subselect* (subselección) o *fullselect* (selección completa) que utiliza un cursor desplazable, la cláusula se ignora y se devuelve un aviso.
- No se puede especificar una *cláusula-isolation* o una *cláusula-lock-request* en un contexto donde provocará un conflicto de aislamiento o un intento de bloqueo en una expresión de tabla común (SQLSTATE 42601). Esta restricción no se aplica a los alias ni a las tablas base. Los ejemplos siguientes crean un conflicto de aislamiento en *a* y devuelven un error:
  - Vista:

    ```
    create view a as (...);
    (select * from a with RR USE AND KEEP SHARE LOCKS)
    UNION ALL
    (select * from a with UR);
    ```

  - Expresión de tabla común:

    ```
    WITH a as (...)
    (select * from a with RR USE AND KEEP SHARE LOCKS)
    UNION ALL
    (select * from a with UR);
    ```

- Una *cláusula-isolation* no se puede especificar en un contexto XML (SQLSTATE 2200M).

Ejemplos de subselecciones

Ejemplo A1: Seleccione todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo A2: Una las tablas EMP_ACT y EMPLOYEE, seleccione todas las columnas de la tabla EMP_ACT y añada el apellido del empleado (LASTNAME) de la tabla EMPLOYEE a cada fila del resultado.

```
SELECT EMP_ACT.*, LASTNAME
  FROM EMP_ACT, EMPLOYEE
  WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

Ejemplo A3: Una las tablas EMPLOYEE y DEPARTMENT, seleccione el número del empleado (EMPNO), el apellido del empleado (LASTNAME), el número del departamento (WORKDEPT en la tabla EMPLOYEE y DEPTNO en la tabla DEPARTMENT) y el nombre del departamento (DEPTNAME) de todos los empleados que han nacido (BIRTHDATE) con anterioridad a 1930.

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

Ejemplo A4: Seleccione el trabajo (JOB) y los salarios máximo y mínimo (SALARY) de cada grupo de filas con el mismo código de trabajo en la tabla EMPLOYEE, pero sólo para los grupos con más de una fila y con un salario máximo mayor o igual que 27000.

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1
  AND MAX(SALARY) >= 27000
```

Ejemplo A5: Seleccione todas las filas de la tabla EMP_ACT para los empleados (EMPNO) del departamento (WORKDEPT) 'E11'. (Los números del departamento del empleado se muestran en la tabla EMPLOYEE.)

```
SELECT *
  FROM EMP_ACT
  WHERE EMPNO IN
           (SELECT EMPNO
               FROM EMPLOYEE
               WHERE WORKDEPT = 'E11')
```

Ejemplo A6: En la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario medio de todos los empleados.

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                           FROM EMPLOYEE)
```

La subconsulta de la cláusula HAVING sólo se ejecutará una vez en este ejemplo.

Ejemplo A7: Utilizando la tabla EMPLOYEE, seleccione el número de departamento (WORKDEPT) y el salario (SALARY) máximo del departamento para todos los departamentos cuyo salario máximo sea menor que el salario medio de los demás departamentos.

```
SELECT WORKDEPT, MAX(SALARY)
   FROM EMPLOYEE EMP_COR
    GROUP BY WORKDEPT
    HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                           FROM EMPLOYEE
                           WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

A diferencia del Ejemplo A6, la subconsulta de la cláusula HAVING se tendría que ejecutar para cada grupos.

Ejemplo A8: Determine el número de empleado y el salario de los representantes de ventas junto con el salario medio y cuenta punta de sus departamentos.

Esta consulta primero debe crear una expresión de tabla anidada (DINFO) para obtener las columnas AVGSALARY y EMPCOUNT, así como la columna DEPTNO que se utiliza en la cláusula WHERE.

```
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
 FROM EMPLOYEE THIS_EMP,
       (SELECT OTHERS.WORKDEPT AS DEPTNO,
                         AVG(OTHERS.SALARY) AS AVGSALARY,
                         COUNT(*) AS EMPCOUNT
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
        ) AS DINFO
 WHERE THIS_EMP.JOB = 'SALESREP'
  AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

Utilizar una expresión de tabla anidada para este caso ahorra la actividad general de crear la vista DIFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y, debido al contexto del resto de la consulta, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

Ejemplo A9: Visualice el nivel de formación medio y el salario de 5 grupos de empleados al azar.

Esta consulta necesita la utilización de una expresión de tabla anidada para establecer el valor aleatorio de cada empleado para que pueda utilizarse posteriormente en la cláusula GROUP BY.

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
           FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

Ejemplo A10: Consultar la tabla EMP_ACT y devolver el número de los proyectos que tengan un empleado cuyo salario se encuentre entre los 10 más altos de todos los empleados.

```
SELECT EMP_ACT.EMPNO,PROJNO
   FROM EMP_ACT
   WHERE EMP_ACT.EMPNO IN
       (SELECT EMPLOYEE.EMPNO
        FROM EMPLOYEE
        ORDER BY SALARY DESC
        FETCH FIRST 10 ROWS ONLY)
```

Ejemplo A11: partiendo de que PHONES e IDS son dos variables de SQL con valores de matriz de la misma cardinalidad, convierta estas matrices en una tabla con tres columnas (una por cada matriz y una por la posición) y una fila por elemento de la matriz.

```
SELECT T.PHONE, T.ID, T.INDEX FROM UNNEST(PHONES, IDS)
  WITH ORDINALITY AS T(PHONE, ID, INDEX)
  ORDER BY T.INDEX
```

Ejemplos de uniones

Ejemplo B1: Este ejemplo ilustra el resultado de varias uniones utilizando las tablas J1 y J2. Estas tablas contienen las filas que se muestran.

```
SELECT * FROM J1

W   X
--- ------
A       11
B       12
C       13

SELECT * FROM J2

Y   Z
--- ------
A       21
C       22
D       23
```

La siguiente consulta realiza una unión interna de J1 y J2, emparejando la primera columna de ambas tablas.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
```

En este ejemplo de unión interna, la fila con la columna W='C' de J1 y la fila con la columna Y='D' de J2 no se incluyen en el resultado porque no tienen una coincidencia en la otra tabla. Observe que la forma alternativa siguiente de una consulta de unión interna genera el mismo resultado.

```
SELECT * FROM J1, J2 WHERE W=Y
```

La unión externa izquierda siguiente recuperará la fila que falta de J1 con nulos para las columnas de J2. Se incluyen todas las filas de J1.

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
B       12 -        -
C       13 C       22
```

La siguiente unión externa derecha recuperará la fila que falta de J2 con nulos para las columnas de J1. Se incluyen todas las filas de J2.

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
```

La siguiente unión externa completa recuperará las filas que faltan de las dos tablas J1 y J2, con nulos cuando sea adecuado. Se incluyen todas las filas de las tablas J1 y J2.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
B       12 -        -
```

Ejemplo B2: Utilizando las tablas J1 y J2 del ejemplo anterior, examine lo que pasa cuando se añade un predicado adicional a la condición de búsqueda.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

La condición adicional ha provocado que la unión interna sólo seleccione 1 fila en comparación con la unión interna del Ejemplo B1.

Observe el impacto que tiene en la unión externa completa.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
-        - A       21
C       13 C       22
-        - D       23
A       11 -        -
B       12 -        -
```

Ahora el resultado tiene 5 filas (a diferencia de 4 sin el predicado adicional) ya que sólo había 1 fila en la unión interna y tienen que devolverse todas las filas de ambas tablas.

La siguiente consulta ilustra que la colocación del mismo predicado adicional en la cláusula WHERE provoca resultados completamente diferentes.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

La cláusula WHERE se aplica después del resultado intermedio de la unión externa completa. Este resultado intermedio sería el mismo que el resultado de la consulta de unión externa completa del Ejemplo B1. La cláusula WHERE se aplica a este resultado intermedio y elimina todas las filas excepto la que contiene X=13. La elección de la ubicación de un predicado cuando se realizan uniones externas puede afectar significativamente a los resultados. Examine lo que pasa si el predicado es X=12 en lugar de X=13. La siguiente unión interna no devuelve ninguna fila.

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

Por lo tanto, la unión externa completa devolvería 6 filas: 3 filas de J1 con nulos para las columnas de J2, y 3 filas de J2 con nulos para las columnas de J1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12

W   X      Y   Z
--- ------ --- ------
```

```
-        - A       21
-        - C       22
-        - D       23
A       11 -        -
B       12 -        -
C       13 -        -
```

En cambio, si el predicado adicional está en la cláusula WHERE, se devuelve la fila 1.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12

W   X      Y   Z
--- ------ --- ------
B       12 -        -
```

Ejemplo B3: Liste todos los departamentos con el número de empleado y el apellido del director, incluyendo los departamentos sin director.

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

Ejemplo B4: Liste todos los números de empleado y el apellido con el número de empleado y el apellido de su director, incluyendo los empleados sin director.

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
  FROM EMPLOYEE E LEFT OUTER JOIN
                                        DEPARTMENT INNER JOIN EMPLOYEE M
       ON MGRNO = M.EMPNO
       ON E.WORKDEPT = DEPTNO
```

La unión interna determina el apellido de cualquier director identificado en la tabla DEPARTMENT y la unión externa izquierda garantiza que se listen todos los empleados incluso si no se encuentra un departamento correspondiente en DEPARTMENT.

Ejemplos de conjuntos de agrupaciones, cube y rollup

Las consultas del Ejemplo C1 al Ejemplo C4 utilizan un subconjunto de las filas de las tablas SALES basadas en el predicado 'WEEK(SALES_DATE) = 13'.

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 LUCCHESSI                 3
         13           6 LUCCHESSI                 1
         13           6 LEE                       2
         13           6 LEE                       2
         13           6 LEE                       3
         13           6 LEE                       5
         13           6 GOUNOT                    3
         13           6 GOUNOT                    1
         13           6 GOUNOT                    7
         13           7 LUCCHESSI                 1
         13           7 LUCCHESSI                 2
         13           7 LUCCHESSI                 1
         13           7 LEE                       7
```

```
          13           7 LEE                      3
          13           7 LEE                      7
          13           7 LEE                      4
          13           7 GOUNOT                   2
          13           7 GOUNOT                  18
          13           7 GOUNOT                   1
```

Ejemplo C1: Esta es una consulta con una cláusula GROUP BY básica en 3 columnas:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
          13           6 GOUNOT                  11
          13           6 LEE                     12
          13           6 LUCCHESSI                4
          13           7 GOUNOT                  21
          13           7 LEE                     21
          13           7 LUCCHESSI                4
```

Ejemplo C2: Genere el resultado basándose en dos conjuntos de agrupaciones diferentes de las filas de la tabla SALES.

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                         (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
          13           - GOUNOT                  32
          13           - LEE                     33
          13           - LUCCHESSI                8
           -           6 GOUNOT                  11
           -           6 LEE                     12
           -           6 LUCCHESSI                4
           -           7 GOUNOT                  21
           -           7 LEE                     21
           -           7 LUCCHESSI                4
```

Las filas con WEEK 13 son del primer conjunto de agrupación y las demás filas son del segundo conjunto de agrupación.

Ejemplo C3: si utiliza las 3 columnas diferenciadas implicadas en los conjuntos de agrupaciones del Ejemplo C2 y realiza ROLLUP, puede ver los conjuntos de agrupaciones para (WEEK,DAY_WEEK,SALES_PERSON), (WEEK, DAY_WEEK), (WEEK) y el total.

```
SELECT WEEK(SALES_DATE) AS WEEK,
      DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
      SALES_PERSON, SUM(SALES) AS UNITS_SOLD
```

```
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - -                        73
          -           - -                        73
```

Ejemplo C4: si ejecuta la misma consulta que el Ejemplo C3 sustituyendo sólo ROLLUP por CUBE, puede ver conjuntos de agrupaciones adicionales para (WEEK,SALES_PERSON), (DAY_WEEK,SALES_PERSON), (DAY_WEEK), (SALES_PERSON) en el resultado.

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

Da como resultado:

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
         13           - -                        73
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           6 -                        27
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
          -           7 -                        46
          -           - GOUNOT                   32
          -           - LEE                      33
          -           - LUCCHESSI                 8
          -           - -                        73
```

Ejemplo C5: Obtenga un conjunto resultante que incluya un total de filas seleccionadas de la tabla SALES junto con un grupo de filas agregadas por SALES_PERSON y MONTH.

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),
                         ()
                       )
ORDER BY SALES_PERSON, MONTH
```

Da como resultado:

```
SALES_PERSON    MONTH       UNITS_SOLD
--------------- ----------- -----------
GOUNOT                    3          35
GOUNOT                    4          14
GOUNOT                   12           1
LEE                       3          60
LEE                       4          25
LEE                      12           6
LUCCHESSI                 3           9
LUCCHESSI                 4           4
LUCCHESSI                12           1
-                         -         155
```

Ejemplo C6: Este ejemplo muestra dos consultas ROLLUP simples seguidas de una consulta que trata los dos ROLLUP como conjuntos de agrupaciones en un sólo conjunto resultante y especifica el orden de filas para cada columna implicada en los conjuntos de agrupaciones.

Ejemplo C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

da como resultado:

```
WEEK        DAY_WEEK    UNITS_SOLD
----------- ----------- -----------
         13           6          27
         13           7          46
         13           -          73
         14           1          31
         14           2          43
         14           -          74
         53           1           8
         53           -           8
          -           -         155
```

Ejemplo C6-2:

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION
```

da como resultado:

```
MONTH       REGION          UNITS_SOLD
----------- --------------- -----------
          3 Manitoba                 22
          3 Ontario-North             8
```

```
           3 Ontario-South            34
           3 Quebec                   40
           3 -                       104
           4 Manitoba                 17
           4 Ontario-North             1
           4 Ontario-South            14
           4 Quebec                   11
           4 -                        43
          12 Manitoba                  2
          12 Ontario-South             4
          12 Quebec                    2
          12 -                         8
           - -                       155
```

Ejemplo C6-3:

```
  SELECT WEEK(SALES_DATE) AS WEEK,
         DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
         MONTH(SALES_DATE) AS MONTH,
         REGION,
         SUM(SALES) AS UNITS_SOLD
  FROM SALES
  GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                           ROLLUP( MONTH(SALES_DATE), REGION )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

da como resultado:

```
  WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
  ----------- ----------- ----------- --------------- -----------
           13           6           - -                        27
           13           7           - -                        46
           13           -           - -                        73
           14           1           - -                        31
           14           2           - -                        43
           14           -           - -                        74
           53           1           - -                         8
           53           -           - -                         8
            -           -           3 Manitoba                 22
            -           -           3 Ontario-North             8
            -           -           3 Ontario-South            34
            -           -           3 Quebec                   40
            -           -           3 -                       104
            -           -           4 Manitoba                 17
            -           -           4 Ontario-North             1
            -           -           4 Ontario-South            14
            -           -           4 Quebec                   11
            -           -           4 -                        43
            -           -          12 Manitoba                  2
            -           -          12 Ontario-South             4
            -           -          12 Quebec                    2
            -           -          12 -                         8
            -           -           - -                       155
            -           -           - -                       155
```

La utilización de los dos ROLLUP como conjuntos de agrupaciones hace que el resultado incluya filas duplicadas. Incluso hay dos filas del total.

Observe cómo la utilización de ORDER BY ha afectado al resultado:

- En el primer conjunto agrupado, la semana 53 se ha cambiado a la posición final.
- En el segundo conjunto agrupado, el mes 12 se ha puesto al final y las regiones aparecen por orden alfabético.
- Los valores nulos se clasifican arriba.

Ejemplo C7: en las consultas que realizan varios ROLLUP en una sola pasada (como, por ejemplo, el Ejemplo C6-3), tiene la posibilidad de indicar, si lo desea, qué conjunto de agrupaciones ha producido cada fila. Los pasos siguientes demuestran cómo proporcionar una columna (denominada GROUP) que indica la fuente de cada fila del conjunto resultante. Por fuente se quiere decir cual de los dos conjuntos de agrupaciones ha producido la fila del conjunto resultante.

Paso 1: Introduzca una manera de ″generar″ los nuevos valores de datos, utilizando una consulta que los selecciona en la cláusula VALUES (que es una forma alternativa de una selección completa). Esta consulta muestra cómo se puede derivar una tabla llamada ″X″ que tienen 2 columnas ″R1″ y ″R2″ y 1 fila de datos.

```
SELECT R1,R2
FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

da como resultado:

```
R1      R2
------- -------
GROUP 1 GROUP 2
```

Paso 2: Forme el producto cruzado de esta tabla ″X″ con la tabla SALES. Esto añade las columnas ″R1″ y ″R2″ a cada fila.

```
SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

Esto añade las columnas ″R1″ y ″R2″ a cada fila.

Paso 3: Ahora se pueden combinar estas columnas con los conjuntos de agrupaciones para que incluyan estas columnas en el análisis de avance.

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                         DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

da como resultado:

```
R1      R2      WEEK     DAY_WEEK  MONTH     REGION          UNITS_SOLD
------- ------- -------- --------- --------- --------------- -----------
GROUP 1 -               13          6           - -                 27
GROUP 1 -               13          7           - -                 46
GROUP 1 -               13          -           - -                 73
GROUP 1 -               14          1           - -                 31
GROUP 1 -               14          2           - -                 43
GROUP 1 -               14          -           - -                 74
GROUP 1 -               53          1           - -                  8
GROUP 1 -               53          -           - -                  8
-       GROUP 2          -          -           3 Manitoba          22
-       GROUP 2          -          -           3 Ontario-North      8
-       GROUP 2          -          -           3 Ontario-South     34
-       GROUP 2          -          -           3 Quebec            40
-       GROUP 2          -          -           3 -                104
-       GROUP 2          -          -           4 Manitoba          17
-       GROUP 2          -          -           4 Ontario-North      1
```

```
-       GROUP 2           -           -           4 Ontario-South     14
-       GROUP 2           -           -           4 Quebec            11
-       GROUP 2           -           -           4 -                 43
-       GROUP 2           -           -          12 Manitoba           2
-       GROUP 2           -           -          12 Ontario-South      4
-       GROUP 2           -           -          12 Quebec             2
-       GROUP 2           -           -          12 -                  8
-       GROUP 2           -           -           - -                155
GROUP 1 -                 -           -           - -                155
```

Paso 4: Tenga en cuenta que puesto que R1 y R2 se utilizan en conjuntos de agrupaciones diferentes, siempre que R1 no sea nulo en el resultado, R2 es nulo y siempre que R2 sea no nulo en el resultado, R1 es nulo. Esto significa que puede consolidar estas columnas en una sola utilizando la función COALESCE. También puede utilizar esta columna en la cláusula ORDER BY para conservar el resultado de los dos conjuntos de agrupaciones juntos.

```
SELECT COALESCE(R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                         DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;
```

da como resultado:

```
GROUP   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
------- ----------- ----------- ----------- --------------- -----------
GROUP 1          13           6           - -                        27
GROUP 1          13           7           - -                        46
GROUP 1          13           -           - -                        73
GROUP 1          14           1           - -                        31
GROUP 1          14           2           - -                        43
GROUP 1          14           -           - -                        74
GROUP 1          53           1           - -                         8
GROUP 1          53           -           - -                         8
GROUP 1           -           -           - -                       155
GROUP 2           -           -           3 Manitoba                 22
GROUP 2           -           -           3 Ontario-North             8
GROUP 2           -           -           3 Ontario-South            34
GROUP 2           -           -           3 Quebec                   40
GROUP 2           -           -           3 -                       104
GROUP 2           -           -           4 Manitoba                 17
GROUP 2           -           -           4 Ontario-North             1
GROUP 2           -           -           4 Ontario-South            14
GROUP 2           -           -           4 Quebec                   11
GROUP 2           -           -           4 -                        43
GROUP 2           -           -          12 Manitoba                  2
GROUP 2           -           -          12 Ontario-South             4
GROUP 2           -           -          12 Quebec                    2
GROUP 2           -           -          12 -                         8
GROUP 2           -           -           - -                       155
```

Ejemplo C8: el ejemplo siguiente ilustra la utilización de varias funciones agregadas cuando se realiza un CUBE. El ejemplo también utiliza funciones de conversión del tipo de datos y el redondeo para producir resultados decimales con una precisión y escala razonables.

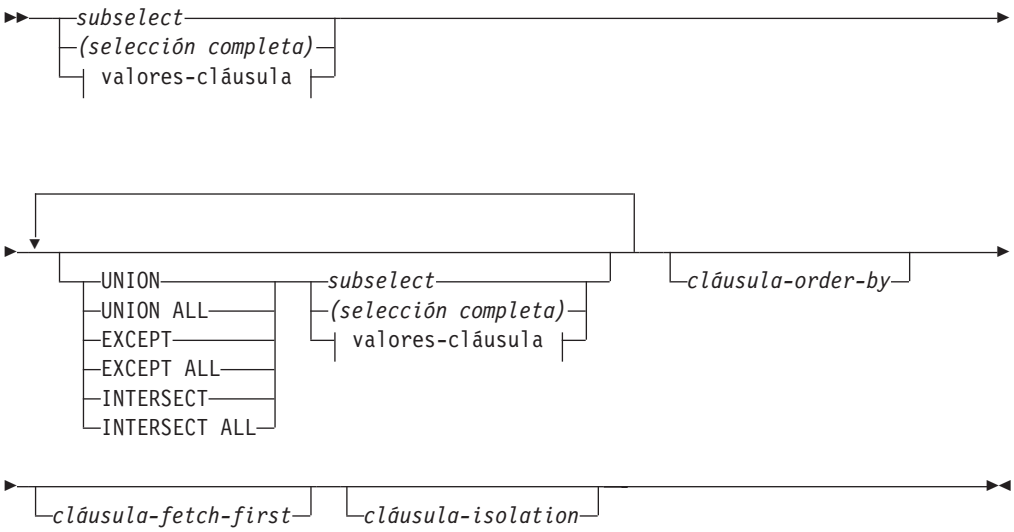
```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,
```

```
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
  FROM SALES
  GROUP BY CUBE(MONTH(SALES_DATE),REGION)
  ORDER BY MONTH, REGION
```

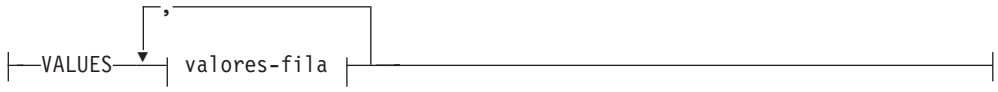
Da como resultado:

```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
          3 Manitoba                 22           7           3.14
          3 Ontario-North             8           3           2.67
          3 Ontario-South            34          14           4.25
          3 Quebec                   40          18           5.00
          3 -                       104          18           4.00
          4 Manitoba                 17           9           5.67
          4 Ontario-North             1           1           1.00
          4 Ontario-South            14           8           4.67
          4 Quebec                   11           8           5.50
          4 -                        43           9           4.78
         12 Manitoba                  2           2           2.00
         12 Ontario-South             4           3           2.00
         12 Quebec                    2           1           1.00
         12 -                         8           3           1.60
          - Manitoba                 41           9           3.73
          - Ontario-North             9           3           2.25
          - Ontario-South            52          14           4.00
          - Quebec                   53          18           4.42
          - -                       155          18           3.87
```

Selección completa



cláusula-values:



fila-valores:

La *selección completa* es un componente de la sentencia select, la sentencia INSERT y la sentencia CREATE VIEW. También es un componente de ciertos predicados que, a su vez, son componentes de una sentencia. Una selección completa que es un componente de un predicado se denomina *subconsulta*, y una selección completa que se especifica entre paréntesis a veces se denomina subconsulta.

Los operadores de conjuntos UNION, EXCEPT e INTERSECT se corresponden a los operadores relacionales de unión, diferencia e intersección.

Una selección completa especifica una tabla resultante. Si no se utiliza un operador de conjunto, el resultado de la selección completa es el resultado de la subselección especificada o la cláusula-values.

La autorización para una *selección completa* se describe en la sección de autorización en ″Consultas de SQL″.

cláusula-values
Obtiene una tabla resultante especificando los valores reales, utilizando expresiones o expresiones de fila, para cada columna de una fila de la tabla resultante. Pueden especificarse múltiples filas. El tipo de resultado de una expresión en la *cláusula-values* no puede ser un tipo de fila (SQLSTATE 428H2).

NULL sólo se puede utilizar con varias especificaciones de *fila-valores*, bien como valor de columna de una tabla resultante de una sola columna, bien en una *expresión-fila*, y como mínimo una fila de la misma columna no debe ser NULL (SQLSTATE 42608).

Una *fila-valores* se especifica por:

- Una sola expresión para una tabla resultante de una sola columna
- *n* expresiones (o NULL) separadas por comas y especificadas entre paréntesis, donde *n* es el número de columnas de la tabla resultante o una expresión de fila para una tabla de resultados de varias columnas.

La cláusula VALUES de varias filas debe tener el mismo número de columnas en cada *fila-valores* (SQLSTATE 42826).

A continuación se muestran ejemplos de *cláusula-values* y sus significados.

```
VALUES (1),(2),(3)          - 3 filas de 1 columna
VALUES 1, 2, 3              - 3 filas de 1 columna
VALUES (1, 2, 3)            - 1 fila de 3 columnas
VALUES (1,21),(2,22),(3,23) - 3 filas de 2 columnas
```

Una *cláusula-values* que está compuesta de *n* especificaciones de *fila-valores*, RE_1 a RE_n, donde *n* es mayor que 1, es equivalente a:

```
RE1 UNION ALL RE2 ... UNION ALL REn
```

Esto supone que las columnas correspondientes de cada *fila-valores* deben ser comparables (SQLSTATE 42825).

UNION o UNION ALL
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se ha especificado UNION ALL, el resultado consta de todas las filas de R1 y de R2. Si se especifica UNION sin la opción ALL, el resultado consta de todas las filas de R1 o R2, con las filas duplicadas eliminadas. Sin embargo, en cualquier caso, todas las filas de la tabla UNION es una fila de R1 o una fila de R2.

EXCEPT o EXCEPT ALL
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica EXCEPT ALL, el resultado consta de todas las filas que no tienen una fila correspondiente en R2, donde las filas duplicadas son significativas. Si se especifica EXCEPT sin la opción ALL, el resultado consta de todas las filas que están sólo en R1, y las filas duplicadas se eliminan del resultado de esta operación.

Por compatibilidad con otras implementaciones de SQL, se puede especificar MINUS como sinónimo de EXCEPT.

INTERSECT o INTERSECT ALL
Obtiene una tabla resultante combinando las otras dos tablas resultantes (R1 y R2). Si se especifica INTERSECT ALL, el resultado consta de todas las filas que están en R1 y en R2. Si se especifica INTERSECT sin la opción ALL, el resultado consta de todas las filas que están en R1 y en R2, con las filas duplicadas eliminadas.

cláusula-order-by
Consulte "subselect" para obtener más detalles sobre la *cláusula-order-by*. No se puede especificar una selección completa que contenga una cláusula ORDER BY en (SQLSTATE 428FJ):

- Una tabla de consultas materializadas
- La selección completa más externa de una vista

Nota: Una cláusula ORDER BY en una selección completa no afecta el orden de las filas que una consulta devuelve. Una cláusula ORDER BY sólo afecta el orden de las filas devueltas si se especifica en la selección completa más externa.

cláusula-fetch-first
Consulte "subselect" para obtener más detalles sobre la *cláusula-fetch-first*. No se puede especificar una selección completa que contenga una cláusula FETCH FIRST en (SQLSTATE 428FJ):

- Una tabla de consultas materializadas
- La selección completa más externa de una vista

Nota: una cláusula FETCH FIRST en una selección completa no influye en el número de filas que devuelve una consulta. Una cláusula FETCH FIRST sólo influye en el número de filas devueltas si se especifica en la selección completa más externa.

cláusula-isolation
Consulte "subselect" para obtener más detalles sobre la *cláusula-isolation*. Si se especifica *cláusula-isolation* para una selección completa y se puede aplicar de la misma forma a una subselección de la selección completa, la *cláusula-isolation* se aplica a la selección completa. Por ejemplo, considere la consulta siguiente.

```
SELECT NAME FROM PRODUCT
UNION
SELECT NAME FROM CATALOG
WITH UR
```

Aunque la cláusula de aislamiento WITH UR podría aplicarse únicamente a la subselección SELECT NAME FROM CATALOG, se aplica a toda la selección completa.

El número de columnas de las tablas resultantes R1 y R2 han de ser iguales (SQLSTATE 42826). Si no se especifica la palabra clave ALL, R1 y R2 no deben contener ninguna columna que tenga el tipo de datos de CLOB, DBCLOB, BLOB, un tipo diferenciado de estos tipos o un tipo estructurado (SQLSTATE 42907).

Las columnas del resultado tienen estos nombres:

- Si las columnas *n* de R1 y *n* de R2 tienen el mismo nombre de columna del resultado, la columna *n* de R tiene el nombre de la columna del resultado.
- Si la columna *n* de R1 y la columna *n* de R2 tienen nombres de columna del resultado diferente, se genera un nombre. Este nombre no se puede utilizar como nombre de columna en una cláusula ORDER BY o UPDATE.

El nombre generado se puede determinar ejecutando DESCRIBE de la sentencia de SQL y consultando el campo SQLNAME.

Filas duplicadas: Dos filas se consideran duplicadas si cada valor de la primera es igual al valor correspondiente de la segunda. Para la determinación de duplicados, dos valores nulos se consideran iguales y dos representaciones de coma flotante

decimal diferentes del mismo número se consideran iguales. Por ejemplo, 2,00 y 2,0 tienen el mismo valor (2,00 y 2,0 se comparan como iguales) pero tienen diferentes exponentes, que le permite representar tanto 2,00 como 2,0. Por tanto, si por ejemplo la tabla de resultados de una operación de UNION contiene una columna de coma flotante decimal y hay varias representaciones del mismo número, la que se devuelve (por ejemplo, 2,00 ó 2,0) es imprevisible. Para obtener más información, consulte el apartado "Comparaciones numéricas" en la página 133.

Cuando se combinan múltiples operaciones en una expresión, las operaciones entre paréntesis se llevan a cabo primero. Si no hay paréntesis, las operaciones se llevan a cabo de izquierda a derecha a excepción de que todas las operaciones INTERSECT se efectúan antes que las operaciones UNION o EXCEPT.

En el ejemplo siguiente, los valores de las tablas R1 y R2 se muestran en la izquierda. Las otras cabeceras listadas muestran los valores como resultado de varias operaciones de conjunto en R1 y en R2.

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					
		3					
		4					
		4					
		4					
		5					

Ejemplos de una selección completa

Ejemplo 1: Selección de todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Liste los números de empleado (EMPNO) de todos los empleados de la tabla EMPLOYEE cuyo número de departamento (WORKDEPT) empiece por 'E' **o** que estén asignados a proyectos de la tabla EMP_ACT cuyo número de proyecto (PROJNO) sea igual a 'MA2100', 'MA2110' o 'MA2112'.

```
SELECT EMPNO
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
```

```
UNION
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 3: Haga la misma consulta que en el ejemplo 2 y, además, "identifique" las filas de la tabla EMPLOYEE con 'emp' y las filas de la tabla EMP_ACT con 'emp_act'. A diferencia del resultado del ejemplo 2, esta consulta puede devolver el mismo EMPNO más de una vez, identificando la tabla del que proviene mediante el "identificador" asociado.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'emp_act' FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 4: Realice la misma consulta que en el ejemplo 2, sólo que utilice UNION ALL para que no se elimine ninguna fila duplicada.

```
SELECT EMPNO
    FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

Ejemplo 5: Realice la misma consulta que en el ejemplo 3, sólo que esta vez incluya dos empleados adicionales que no están actualmente en ninguna tabla e identifíquelos como ″new″.

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHEREWORKDEPTLIKE 'E%'
UNION
SELECT EMPNO, 'emp_act'
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
   VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

Ejemplo 6: Este ejemplo de EXCEPT genera todas las filas que están en T1 pero no en T2.

```
(SELECT * FROM T1)
EXCEPT ALL
(SELECT * FROM T2)
```

Si no hay ningún valor NULL implicado, este ejemplo devuelve los mismos resultados que

```
SELECT ALL *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                     WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

Ejemplo 7: Este ejemplo de INTERSECT genera todas las filas que están en ambas tablas, T1 y T2, eliminando los duplicados.

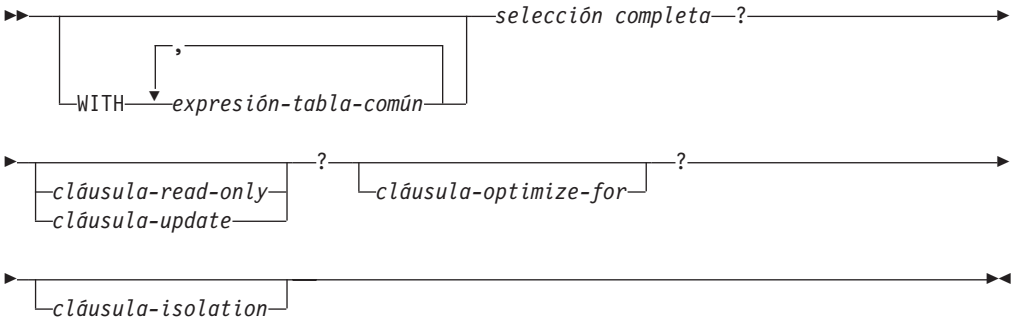
```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2)
```

Si no hay valores NULL implicados, este ejemplo devuelve el mismo resultado que

```
SELECT DISTINCT * FROM T1
   WHERE EXISTS (SELECT * FROM T2
                 WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

donde C1, C2, etcétera representan las columnas de T1 y T2.

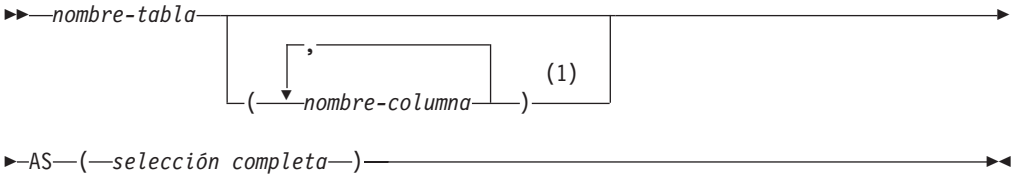
sentencia-select



La *sentencia-select* es la forma de una consulta que puede especificarse directamente en una sentencia DECLARE CURSOR o prepararse y después hacerse referencia en una sentencia DECLARE CURSOR. También puede emitirse mediante sentencias de SQL dinámico utilizando el procesador de la línea de mandatos (o herramientas similares) haciendo que se visualice una tabla resultante en la pantalla del usuario. En cualquier caso, la tabla especificada por una *sentencia-select* es el resultado de la selección completa.

La autorización para una *sentencia-select* se describe en la sección de autorización en ″Consultas de SQL″.

expresión-tabla-común



Notas:

1 Si una expresión de tabla común es recursiva o si la selección completa da como resultado nombres de columna duplicados, deben especificarse los nombres de columna.

Una *expresión de tabla común* permite la definición de una tabla resultante con un *nombre-tabla* que puede especificarse como un nombre de tabla en cualquier cláusula FROM de la selección completa que sigue. Se pueden especificar múltiples expresiones de tabla comunes a continuación de una sola palabra clave WITH. Las expresiones de tabla comunes subsiguientes también pueden referirse por nombre en su cláusula FROM a cualquier expresión de tabla común especificada.

Si se especifica una lista de columnas, debe constar de tantos nombres como el número de columnas de la tabla resultante de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifican estos nombres de columna, los nombres se obtienen de la lista de selección de la selección completa utilizada para definir la expresión de tabla común.

El *nombre-tabla* de una expresión de tabla común debe ser diferente de cualquier otro *nombre-tabla* de una expresión de tabla común de la misma sentencia

(SQLSTATE 42726). Si se especifica la expresión de tabla común en una sentencia INSERT, el *nombre-tabla* no puede ser el mismo que el nombre de tabla o vista que es el objeto de la inserción (SQLSTATE 42726). Un *nombre-tabla* de una expresión de tabla común puede especificarse como nombre de tabla en cualquier cláusula FROM de toda la selección completa. Un *nombre-tabla* de una expresión de tabla común prevalece sobre cualquier tabla, vista o alias existentes (en el catálogo) que tenga el mismo nombre calificado.

Si se define más de una expresión de tabla común en la misma sentencia, no están permitidas las referencias cíclicas entre las expresiones de tabla comunes (SQLSTATE 42835). Una *referencia cíclica* se produce cuando dos expresiones de tabla comunes *dt1* y *dt2* están creadas de tal manera que *dt1* hace referencia a *dt2* y *dt2* hace referencia a *dt1*.

Si una selección completa de una expresión de tabla común contiene una *referencia a tabla de cambio de datos* en la cláusula FROM, se dice que la expresión de tabla común modifica los datos. Una expresión de tabla común que modifica datos siempre se evalúa cuando se procesa la sentencia, independientemente de si la expresión de tabla común se utiliza en cualquier otro lugar de la sentencia. Si hay al menos una expresión de tabla común que lee o modifica datos, todas las expresiones de tabla común se procesan en el orden en el que aparecen, y cada expresión de tabla común que lee o modifica datos se ejecuta por completo, incluidas todas las restricciones y activadores, antes de que se ejecuten las expresiones de tabla comunes siguientes.

La expresión de tabla común también es opcional antes de la selección completa en las sentencias CREATE VIEW e INSERT.

Una expresión de tabla común puede utilizarse:

- En lugar de una vista para evitar crear la vista (cuando no sea necesaria la utilización general de la vista y no se utilicen actualizaciones ni supresiones colocadas)
- Para permitir la agrupación por una columna que se obtiene de una subselección o función escalar que no es determinista o tiene una acción externa
- Cuando la tabla resultante deseada se basa en variables del lenguaje principal
- Cuando la misma tabla resultante necesite compartirse en una selección completa
- Cuando el resultado necesita obtenerse mediante recurrencia
- Cuando se tienen que procesar varias sentencias de cambio de datos SQL dentro de la consulta

Si la selección completa de una expresión de tabla común contiene una referencia a sí misma en una cláusula FROM, la expresión de tabla común es *recursiva*. Las consultas que utilizan la recurrencia son útiles en las aplicaciones que permiten su uso, tales como la lista de material (BOM), sistemas de reservas y planificación de la red.

Deben cumplirse las condiciones siguientes en una expresión de tabla común recursiva:

- Cada selección completa que forma parte del ciclo de repetición debe empezar por SELECT o SELECT ALL. La utilización de SELECT DISTINCT no está permitida (SQLSTATE 42925). Además, las uniones deben utilizar UNION ALL (SQLSTATE 42925).

- Los nombres de columna deben especificarse a continuación del *nombre-tabla* de la expresión de tabla común (SQLSTATE 42908).
- La primera selección completa de la primera unión (la selección completa de inicialización) no debe incluir ninguna referencia a ninguna columna de la expresión de tabla común de cualquier cláusula FROM (SQLSTATE 42836).
- Si se hace referencia a un nombre de columna de la expresión de tabla común en la selección completa repetida, el tipo de datos, longitud y página de códigos para la columna se determinan basándose en la selección completa de inicialización. La columna correspondiente de la selección completa recursiva debe tener el mismo tipo de datos y longitud que el tipo de datos y longitud determinados en base a la selección completa de inicialización y la página de códigos debe coincidir (SQLSTATE 42825). Sin embargo, para los tipos de serie de caracteres, la longitud de los dos tipos de datos puede diferir. En este caso, la columna de la selección completa recursiva debe tener una longitud que podría asignarse siempre a la longitud determinada de la selección completa de inicialización.
- Cada selección completa que forma parte del ciclo de repetición no debe incluir ninguna función agregada, cláusula-group-by ni cláusula-having (SQLSTATE 42836).

  Las cláusulas FROM de estas selecciones completas pueden incluir como máximo una referencia a una expresión de tabla común que forme parte de un ciclo de repetición (SQLSTATE 42836).
- Ni la selección completa iterativa ni la selección completa recursiva global puede incluir una cláusula-order-by (SQLSTATE 42836).
- Las subconsultas (escalares o cuantificadas) no deben formar parte de ciclos de repetición (SQLSTATE 42836).

Cuando desarrolle expresiones de tabla comunes recursivas, recuerde que se puede crear un ciclo de repetición infinito (bucle). Compruebe que los ciclos de repetición terminen. Es muy importante si los datos implicados son cíclicos. Se espera que una expresión de tabla común recursiva incluya un predicado que impida un bucle infinito. Se espera que la expresión de tabla común recursiva incluya:

- Una selección completa recursiva, una columna de enteros incrementada por una constante.
- Un predicado en la cláusula where de la selección completa recursiva con el formato col_contador < constante" o "col_contador < :var_lengpral".

Se emite un aviso si no se encuentra esta sintaxis en la expresión de tabla común recursiva (SQLSTATE 01605).

Ejemplo de recurrencia: Lista de material

Las aplicaciones de tipo Lista de material (BOM) son una necesidad habitual en muchos entornos comerciales. Para ilustrar la capacidad de una expresión de tabla común recursiva para aplicaciones BOM, considere una tabla de piezas con subpiezas asociadas y la cantidad de subpiezas que se precisan en la pieza. Para este ejemplo, cree la tabla como se muestra a continuación:

```
CREATE TABLE PARTLIST
              (PIEZA VARCHAR(8),
               SUBPIEZA VARCHAR(8),
               CANTIDAD INTEGER);
```

Para obtener resultados de consulta en este ejemplo, supongamos que la tabla LISTA DE PIEZAS contiene los siguientes valores:

```
PIEZA    SUBPIEZA CANTIDAD
-------- -------- -----------
00       01                 5
00       05                 3
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       14                 8
07       12                 8
```

Ejemplo 1: Explosión de primer nivel

El primer ejemplo se denomina explosión de primer nivel. Responde a la pregunta "¿Qué piezas son necesarias para crear la pieza identificada mediante '01'?". La lista incluirá las subpiezas directas, subpiezas de subpiezas, etc. Sin embargo, si una pieza se utiliza varias veces, las subpiezas correspondientes sólo aparecerán en la lista una vez.

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
        FROM LISTA DE PIEZAS RAIZ
        WHERE PIEZA.RAIZ = '01'
      UNION ALL
        SELECT PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
        FROM RPL PADRE, LISTA DE PIEZAS HIJA
        WHERE SUBPIEZA.PADRE = PIEZA.HIJA
     )
SELECT DISTINCT PIEZA, SUBPIEZA, CANTIDAD
 FROM RPL
  ORDER BY PIEZA, SUBPIEZA, CANTIDAD;
```

La consulta anterior incluye una expresión de tabla común, identificada mediante el nombre *RPL*, que expresa la pieza repetitiva de esta consulta. Ilustra los elementos básicos de una expresión de tabla común recursiva.

El primer operando (selección completa) de la UNION, al que se hace referencia como la *selección completa de inicialización*, obtiene los hijos directos de la pieza '01'. La cláusula FROM de esta selección completa hace referencia a la tabla fuente y nunca se hará referencia a sí misma (*RPL* en este caso). El resultado de la primera selección completa va a la expresión de tabla común *RPL* (LISTA DE PIEZAS recursiva). Como en este ejemplo, UNION debe ser siempre UNION ALL.

El segundo operando (selección completa) de UNION utiliza *RPL* para calcular las subpiezas de subpiezas haciendo que la cláusula FROM hará referencia a la expresión de tabla común *RPL* y la tabla fuente con una unión de una pieza de la tabla fuente (hija) a una subpieza del resultado actual contenido en *RPL* (padre). El resultado vuelve a *RPL* de nuevo. El segundo operando de UNION se utiliza entonces repetidamente hasta que ya no existan más hijas.

SELECT DISTINCT de la selección completa principal de esta consulta, garantiza que no aparezca en la lista la misma pieza/subpieza más de una vez.

El resultado de la consulta es el siguiente:

```
PIEZA    SUBPIEZA CANTIDAD
-------- -------- -----------
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
06       12                10
06       13                10
07       12                 8
07       14                 8
```

Observe, en el resultado, que de la pieza '01' se pasa a la pieza '02', que a su vez pasa a la '06', etc. Observe también que la pieza '06' se alcanza dos veces, una a través de '01' directamente y otra a través de '02'. En el resultado, sin embargo, los subcomponentes sólo aparecen una vez en la lista (es el resultado de utilizar SELECT DISTINCT), tal como se requiere.

Es importante recordar que con las expresiones de tabla comunes recursivas puede generarse un *bucle infinito*. En este ejemplo, se produciría un bucle infinito si la condición de búsqueda del segundo operando que une las tablas madre e hija tuviera esta codificación:

```
SUBPIEZA.PADRE = SUBPIEZA.HIJA
```

Este ejemplo de bucle infinito es consecuencia de no codificar lo que se intenta codificar. Sin embargo, debe extremar la precaución al determinar qué es lo que se ha de codificar, de forma que se consiga un final definitivo del ciclo de recurrencia.

El resultado de esta consulta de ejemplo puede producirse en un programa de aplicación sin utilizar una expresión de tabla común recursiva. Sin embargo, ello requeriría iniciar una nueva consulta para cada nivel de repetición. Además, la aplicación necesita colocar de nuevo todos los resultados en la base de datos para ordenar el resultado. Todo ello hace que la lógica de la aplicación se complique y que el funcionamiento no sea el esperado. La lógica de la aplicación resulta aún más complicada e ineficaz para consultas de otras listas de material, tales como consultas resumidas y de explosión.

Ejemplo 2: Explosión resumida

El segundo ejemplo es una explosión resumida. La cuestión que se plantea aquí es la cantidad total de cada pieza que se requiere para crear la pieza '01'. La diferencia principal de la explosión de un solo nivel es la necesidad de agregar las cantidades. El primer ejemplo indica la cantidad de subpiezas necesarias para la pieza siempre que se requiera. No indica cuántas de las subpiezas se necesitan para crear la pieza '01'.

```
WITH RPL (PART, SUBPART, QUANTITY) AS
   (
      SELECT PIEZA.RAIZ, SUBPIEZA.RAIZ, CANTIDAD.RAIZ
       FROM LISTA DE PIEZAS RAIZ
       WHERE PIEZA.RAIZ = '01'
    UNION ALL
      SELECT PIEZA.PADRE, SUBPIEZA.HIJA, CANTIDAD.PADRE*CANTIDAD.HIJA
```

```
      FROM RPL PADRE, LISTA DE PIEZAS HIJA
      WHERE SUBPIEZA.PADRE = PIEZA.HIJA
  )
SELECT PIEZA, SUBPIEZA, SUM(CANTIDAD) AS "Cantidad total utilizada"
 FROM RPL
  GROUP BY PIEZA, SUBPIEZA
  ORDER BY PIEZA, SUBPIEZA;
```

En la consulta anterior, la lista de selección del segundo operando de UNION en la expresión de tabla común recursiva, identificada mediante el nombre *RPL*, muestra la agregación de la cantidad. Para averiguar qué porcentaje de subpieza se utiliza, la cantidad del elemento madre se multiplica por la cantidad por madre de una hija. Si una pieza se utiliza varias veces en lugares diferentes, requerirá otra agregación final. Esto se realiza mediante la agrupación con la expresión de tabla común *RPL* y utilizando la función agregada SUM en la lista de selección de la selección completa.

El resultado de la consulta es el siguiente:

```
PIEZA    SUBPIEZA Cant tot usada
-------- -------- --------------
01       02                    2
01       03                    3
01       04                    4
01       05                   14
01       06                   15
01       07                   18
01       08                   40
01       09                   44
01       10                  140
01       11                  140
01       12                  294
01       13                  150
01       14                  144
```

A la vista del resultado, considere la línea de la subpieza '06'. La cantidad total utilizada, con valor 15, deriva de la cantidad de 3 directamente para la pieza '01' y la cantidad de 6 para la pieza '02', que se necesita 2 veces en la pieza '01'.

Ejemplo 3: Control de profundidad

Puede surgir la cuestión de qué es lo que ocurre cuando existen más niveles de piezas en la tabla de los que está interesado para su consulta. Es decir, cómo se escribe una consulta para responder a la pregunta "Cuáles son los dos primeros niveles de piezas necesarias para crear la pieza identificada como '01'?" Por cuestiones de claridad en el ejemplo, el nivel se incluye en el resultado.

```
WITH RPL (NIVEL, PIEZA, SUBPIEZA, CANTIDAD) AS
      (
        SELECT 1,              PIEZA.RAIZ  SUBPIEZA.RAIZ, CANTIDAD.RAIZ
         FROM LISTA DE PIEZAS RAIZ
         WHERE PIEZA.RAIZ = '01'
       UNION ALL
        SELECT NIVEL+1.PADRE, PIEZA.HIJA, SUBPIEZA.HIJA, CANTIDAD.HIJA
         FROM RPL PADRE, LISTA DE PIEZAS HIJA
         WHERE SUBPIEZA.PADRE = PIEZA.HIJA
           AND NIVEL.PADRE < 2
      )
 SELECT PIEZA, NIVEL, SUBPIEZA, CANTIDAD
   FROM RPL;
```

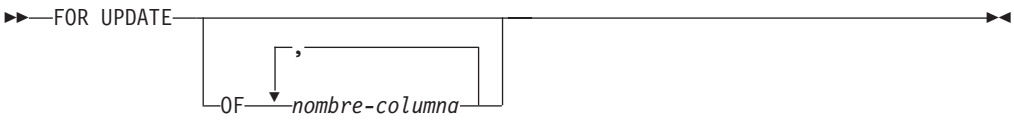
Esta consulta es similar al ejemplo 1. La columna *NIVEL* se ha introducido para contar los niveles desde la pieza original. En la selección completa de

inicialización, el valor de la columna *NIVEL* se inicializa en 1. En la selección completa subsiguiente, el nivel padre se incrementa en 1. A continuación, para controlar el número de niveles del resultado, la segunda selección completa incluye la condición de que el nivel padre debe ser menor que 2. Esto garantiza que la segunda selección completa sólo procesará hijos en el segundo nivel.

El resultado de la consulta es como sigue:

```
PIEZA    NIVEL       SUBPIEZA CANTIDAD
-------- ----------- -------- -----------
01                 1 02                 2
01                 1 03                 3
01                 1 04                 4
01                 1 06                 3
02                 2 05                 7
02                 2 06                 6
03                 2 07                 6
04                 2 08                10
04                 2 09                11
06                 2 12                10
06                 2 13                10
```

cláusula-update



La cláusula FOR UPDATE identifica las columnas que se pueden actualizar en una sentencia UPDATE con posición posterior. Cada *nombre-columna* debe estar sin calificar y debe identificar una columna de la tabla o vista identificada en la primera cláusula FROM de la selección completa. Si la cláusula FOR UPDATE se especifica sin nombres de columna, se incluyen todas las columnas actualizables de la tabla o vista identificadas en la primera cláusula FROM de la selección completa.

La cláusula FOR UPDATE no puede utilizarse si es verdadera una de las siguientes situaciones:

- El cursor asociado con la sentencia select no se puede suprimir.
- Una de las columnas seleccionadas no es una columna actualizable de una tabla del catálogo y la cláusula FOR UPDATE no se ha utilizado para excluir dicha columna.

cláusula-read-only

La cláusula FOR READ ONLY indica que la tabla resultante es de sólo lectura y que, por lo tanto, no se puede hacer referencia al cursor en las sentencias UPDATE con posición y DELETE. FOR FETCH ONLY tiene el mismo significado.

Algunas tablas resultantes son de sólo lectura por naturaleza. (Por ejemplo, una tabla basada en una vista de sólo lectura.) Se puede seguir especificando FOR READ ONLY para dichas tablas, pero la especificación no surtirá efecto.

Para las tablas resultantes en las que están permitidas las actualizaciones y las supresiones, la especificación de FOR READ ONLY (o FOR FETCH ONLY) posiblemente mejorará el rendimiento de las operaciones FETCH, ya que permite al gestor de bases de datos realizar el bloqueo. Por ejemplo, en los programas que contienen sentencias de SQL dinámico sin la cláusula FOR READ ONLY ni ORDER BY, el gestor de bases de datos puede abrir cursores como si hubiese especificado la cláusula FOR UPDATE. Por lo tanto, se recomienda utilizar la cláusula FOR READ ONLY para mejorar el rendimiento, excepto en los casos en que se utilizarán las consultas en sentencias UPDATE o DELETE.

No se debe hacer referencia a una tabla resultante de sólo lectura en una sentencia UPDATE con posición o DELETE, ya sea de sólo lectura por naturaleza o especificada como FOR READ ONLY (FOR FETCH ONLY).

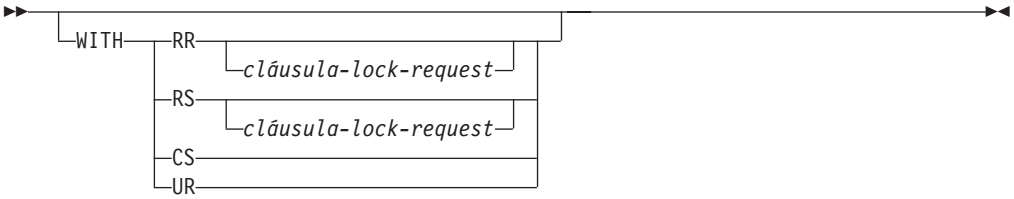
cláusula-optimize-for

La cláusula OPTIMIZE FOR pide el proceso especial de la *sentencia select*. Si se omite la cláusula, se supone que se recuperarán todas las filas de la tabla resultante; si se especifica, se supone que el número de filas recuperado probablemente no excederá de *n* donde *n* es el valor de *entero*. El valor de *n* debe ser un entero positivo. La utilización de la cláusula OPTIMIZE FOR influye en la optimización de la consulta basándose en la suposición de que se recuperarán *n* filas. Además, cuando los cursores están bloqueados, esta cláusula afecta al número de filas que se devuelven en cada bloque (es decir, no se devolverán más de *n* filas en cada bloque). Si se especifican la *cláusula-fetch-first* y la *cláusula-optimize-for*, se utilizará el valor entero menor de estas cláusulas para determinar el tamaño del almacenamiento intermedio de comunicaciones. Los valores se tienen en cuenta de forma independiente por motivos de optimización.

Esta cláusula no limita el número de filas que se pueden recuperar ni afecta al resultado de ninguna otra manera que no sea en el rendimiento. La utilización de OPTIMIZE FOR *n* ROWS puede mejorar el rendimiento si no se recuperan más de *n* filas, pero puede reducir el rendimiento si se recuperan más de *n* filas.

Si el valor de *n* multiplicado por el tamaño de la fila sobrepasa el tamaño del almacenamiento intermedio de comunicaciones, la cláusula OPTIMIZE FOR no tendrá ningún efecto sobre los almacenamientos intermedios de los datos. El tamaño del almacenamiento intermedio de comunicaciones está definido por el parámetro de configuración **rqrioblk** o **aslheapsz**.

cláusula-isolation



La *cláusula-isolation*, opcional, especifica el nivel de aislamiento en el que se ejecuta la sentencia y si debe obtenerse un tipo de bloqueo determinado.

- 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. Cuando se utiliza un apodo en una *sentencia-select* para acceder a datos de fuentes de datos de la familia DB2 y de Microsoft SQL Server, se puede incluir la *cláusula-isolation* en la sentencia para especificar el nivel de aislamiento de sentencia. Si la *cláusula-isolation* se incluye en sentencias que acceden a otras fuentes de datos, el nivel de aislamiento especificado se omite. El nivel de aislamiento actual del servidor federado se correlaciona con el nivel de aislamiento correspondiente en la fuente de datos de cada conexión a la fuente de datos. Tras establecer una conexión con una fuente de datos, el nivel de aislamiento no puede modificarse a lo largo de la conexión.

cláusula-lock-request

La *cláusula-lock-request*, opcional, especifica el tipo de bloqueo que el gestor de bases de datos debe conseguir y retener:

SHARE
Los procesos simultáneos pueden conseguir bloqueos SHARE o UPDATE sobre los datos.

UPDATE
Los procesos simultáneos pueden conseguir bloqueos SHARE sobre los datos pero ningún proceso simultáneo puede conseguir un bloqueo UPDATE o EXCLUSIVE.

EXCLUSIVE
Los procesos simultáneos no pueden conseguir un bloqueo sobre los datos.

La *cláusula-lock-request* se aplica a todas las exploraciones básicas de tabla y de índice que la consulta necesita, incluidas aquellas contenidas en subconsultas, funciones de SQL y métodos de SQL. No tiene ningún efecto sobre los bloqueos realizados por procedimientos, funciones externas o métodos externos. Todas las funciones de SQL o los métodos de SQL invocados (directa o indirectamente) por la sentencia deben crearse con INHERIT ISOLATION LEVEL WITH LOCK REQUEST (SQLSTATE 42601). La *cláusula-lock-request* no puede utilizarse con una consulta de modificación que pueda invocar a activadores o que necesite comprobaciones de la integridad referencial (SQLSTATE 42601).

Ejemplos de sentencia-select

Ejemplo 1: Selección de todas las columnas y filas de la tabla EMPLOYEE.

```
SELECT * FROM EMPLOYEE
```

Ejemplo 2: Selección del nombre del proyecto (PROJNAME), la fecha de inicio (PRSTDATE) y la fecha de finalización (PRENDATE) de la tabla PROJECT. Ordenación de la tabla resultante por la fecha de finalización con las fechas más recientes primero.

```
SELECT PROJNAME, PRSTDATE, PRENDATE
 FROM PROJECT
 ORDER BY PRENDATE DESC
```

Ejemplo 3: Selección del número de departamento (WORKDEPT) y el salario medio del departamento (SALARY) para todos los departamentos de la tabla EMPLOYEE. Ordenación la tabla resultante por orden ascendente por el salario medio del departamento.

```
SELECT WORKDEPT, AVG(SALARY)
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY 2
```

Ejemplo 4: Declaración de un cursor llamado UP_CUR para utilizarlo en un programa C para actualizar las columnas de fecha de inicio (PRSTDATE) y de fecha de finalización (PRENDATE) en la tabla PROJECT. El programa debe recibir los dos valores junto con el valor de número del proyecto (PROJNO) para cada fila.

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE;
```

Ejemplo 5: Este ejemplo denomina a la expresión SAL+BONUS+COMM como TOTAL_PAY

```
  SELECT SALARY+BONUS+COMM AS TOTAL_PAY
    FROM EMPLOYEE
    ORDER BY TOTAL_PAY
```

Ejemplo 6: Determinación del número de empleado y el salario de los representantes de ventas junto con el salario medio y el número total de empleados de sus departamentos. También, listado del salario medio del departamento con el salario medio más alto.

La utilización de una expresión de tabla común para este caso ahorra la actividad de crear una vista DINFO como una vista normal. Durante la preparación de la sentencia, se evita el acceso al catálogo para la vista y, debido al contexto del resto de la selección completa, sólo se han de tener en cuenta las filas para el departamento de representantes de ventas para la vista.

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
       (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
       ),
   DINFOMAX AS
       (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
       DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
 FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

Ejemplo 7: Dadas dos tablas, EMPLOYEE y PROJECT, sustitución del empleado SALLY por el nuevo empleado GEORGE, asignación de todos los proyectos controlados por SALLY a GEORGE y devolución de los nombres de los proyectos actualizados.

```
WITH
  NEWEMP AS (SELECT EMPNO FROM NEW TABLE
              (INSERT INTO EMPLOYEE(EMPNO, FIRSTNME)
                  VALUES(NEXT VALUE FOR EMPNO_SEQ, 'GEORGE'))),
  OLDEMP AS (SELECT EMPNO FROM EMPLOYEE WHERE FIRSTNME = 'SALLY'),
  UPPROJ AS (SELECT PROJNAME FROM NEW TABLE
              (UPDATE PROJECT
                  SET RESPEMP = (SELECT EMPNO FROM NEWEMP)
                  WHERE RESPEMP = (SELECT EMPNO FROM OLDEMP))),
  DELEMP AS (SELECT EMPNO FROM OLD TABLE
              (DELETE FROM EMPLOYEE
                  WHERE EMPNO = (SELECT EMPNO FROM OLDEMP)))
SELECT PROJNAME FROM UPPROJ;
```

Ejemplo 8: Recuperación de los datos de la tabla DEPT. Estos datos se actualizarán más tarde con una actualización buscada y deben estar bloqueados cuando se ejecute la consulta.

```
SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPT
  WHERE ADMRDEPT ='A00'
  FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS
```

Apéndice A. Límites de SQL y XML

Las tablas siguientes describen determinados límites de SQL y XML. Si se ajusta al caso más restrictivo le servirá de ayuda para diseñar programas de aplicación que sean portátiles.

Tabla 64 lista los límites en bytes. Estos límites se imponen después de la conversión de la página de códigos de la aplicación a la página de códigos de la base de datos al crear identificadores. Los límites también se imponen después de la conversión de la página de códigos de la base de datos a la página de códigos de la aplicación al recuperar identificadores de la base de datos. Si, durante cualquiera de estos procesos, se supera el límite de longitud del identificador, se produce un truncado o se devuelve un error.

Los límites en los caracteres varían en función de la página de códigos de la base de datos y la página de códigos de la aplicación. Por ejemplo, debido a que la anchura de un carácter UTF-8 puede ir de 1 a 4 bytes, el límite de caracteres para un identificador en una tabla Unicode cuyo límite sea 128 bytes irá de 32 a 128 caracteres, en función de los caracteres que se utilicen. Si se efectúa un intento de crear un identificador cuyo nombre sea más largo que el límite para esta tabla después de la conversión a la página de códigos de la base de datos, se devolverá un error.

Las aplicaciones que almacenen nombres de identificador deben poder manejar los incrementos potenciales del tamaño de los identificadores una vez se haya producido la conversión de página de códigos. Cuando se recuperan identificadores del catálogo, éstos se convierten a la página de códigos de la aplicación. La conversión de la página de códigos de la base de datos a la página de códigos de la aplicación puede dar como resultado que un identificador llegue a tener una longitud mayor que el límite de bytes para la tabla. Si una variable del lenguaje principal declarada por la aplicación no puede almacenar todo el identificador después de la conversión de la página de códigos, éste se trunca. Si eso es inaceptable, puede aumentar el tamaño de la variable del lenguaje principal para que pueda aceptar el nombre de identificador completo.

Las mismas normas se aplican a los programas de utilidad de DB2 que recuperan datos y los convierten a una página de códigos especificada por el usuario. Si un programa de utilidad de DB2, como por ejemplo una exportación, está recuperando los datos y forzando la conversión a una página de códigos especificada por el usuario (utilizando el modificador CODEPAGE de exportación o la variable de registro **DB2CODEPAGE** y el identificador se expanda más allá del límite documentado en esta tabla debido a la conversión de la página de códigos, es posible que se devuelva un error o que se trunque el identificador.

Tabla 64. Límites de longitud del identificador

Descripción	Máximo en bytes
Nombre de alias	128
Nombre de atributo	128
Nombre de política de comprobación	128
Nombre de autorización (sólo puede ser de caracteres de un solo byte)	128

Tabla 64. Límites de longitud del identificador (continuación)

Descripción	Máximo en bytes
Nombre de agrupación de almacenamiento intermedio	18
Nombre de columna[2]	128
Nombre de restricción	128
Nombre de correlación	128
Nombre de cursor	128
Nombre de partición de datos	128
Nombre de columna de fuente de datos	255
Nombre de índice de fuente de datos	128
Nombre de fuente de datos	128
Nombre de tabla de fuente de datos (*nombre-tabla-remota*)	128
Nombre de grupo de particiones de base de datos	128
Nombre de partición de base de datos	128
Nombre de supervisor de sucesos	128
Nombre de programa externo	128
Nombre de correlación de funciones	128
Nombre de grupo	128
Identificador del lenguaje principal[1]	255
Identificador de un usuario de fuente de datos (*nombre-autorización-remota*)	128
Identificador en un procedimiento SQL (nombre de condición, para identificador de bucle, etiqueta, localizador de conjunto de resultados, nombre de sentencia, nombre de variable)	128
Nombre de índice	128
Nombre de extensión del índice	18
Nombre de especificación del índice	128
Nombre de etiqueta	128
Identificador de recursos uniforme (URI) de espacios de nombre	1000
Apodo	128
Nombre de paquete	128
ID de versión del paquete	64
Nombre de parámetro	128
Contraseña para acceder a una fuente de datos	32
Nombre de procedimiento	128
Nombre de rol	128
Nombre de punto de salvaguarda	128
Nombre de esquema[2]	128
Nombre de componente de etiqueta de seguridad	128
Nombre de etiqueta de seguridad	128
Nombre de política de seguridad	128

Tabla 64. Límites de longitud del identificador (continuación)

Descripción	Máximo en bytes
Nombre de secuencia	128
Nombre de servidor (alias de base de datos)	8
Nombre específico	128
Nombre de condición SQL	128
nombre de variable de SQL	128
Nombre de sentencia	128
Nombre de tabla	128
Nombre de espacio de tablas	18
Nombre del grupo de transformación	18
Nombre de activador	128
Nombre de contexto fiable	128
Nombre de correlación de tipos	18
Nombre de función definida por el usuario	128
Nombre de método definido por el usuario	128
Nombre de tipo definido por el usuario[2]	128
Nombre de vista	128
Nombre de derivador	128
Nombre de elemento XML, nombre de atributo o nombre de prefijo	1000
Identificador de recursos uniforme (URI) de ubicación de esquema XML	1000

Nota:

1. Los compiladores de lenguaje principal individuales pueden aplicar límites más restrictivos a los nombres de variables.
2. La estructura SQLDA está limitada a almacenar nombres de columna de 30 bytes, nombres de tipo definidos por el usuario de 18 bytes y nombres de esquema de 8 bytes para tipos definidos por el usuario. Puesto que el SQLDA se utiliza en la sentencia DESCRIBE, las aplicaciones SQL incorporadas que utilizan la sentencia DESCRIBE para recuperar la columna o la información de nombre de tipo definido por el usuario deben adaptarse a estos límites.

Tabla 65. Límites numéricos

Descripción	Límite
Valor SMALLINT mínimo	-32.768
Valor SMALLINT máximo	+32.767
Valor INTEGER más pequeño	-2.147.483.648
Valor INTEGER máximo	+2.147.483.647
Valor BIGINT más pequeño	-9.223.372.036.854.775.808
Valor BIGINT máximo	+9.223.372.036.854.775.807
Precisión decimal máxima	31
Exponente máximo ($E_{máx}$) para valores REAL	38
Valor REAL mínimo	-3,402E+38

Tabla 65. Límites numéricos (continuación)

Descripción	Límite
Valor REAL máximo	+3,402E+38
Exponente mínimo ($E_{mín}$) para valores REAL	-37
Valor REAL positivo mínimo	+1,175E-37
Valor REAL negativo máximo	-1,175E-37
Exponente máximo ($E_{máx}$) para valores DOUBLE	308
Valor DOUBLE mínimo	-1,79769E+308
Valor DOUBLE máximo	+1,79769E+308
Exponente mínimo ($E_{mín}$) para valores DOUBLE	-307
Valor DOUBLE positivo mínimo	+2,225E-307
Valor DOUBLE negativo máximo	-2,225E-307
Exponente máximo ($E_{máx}$) para valores DECFLOAT(16)	384
Valor mínimo de DECFLOAT(16)[1]	-9.999999999999999E+384
Valor máximo de DECFLOAT(16)	9.999999999999999E+384
Exponente mínimo ($E_{mín}$) para valores DECFLOAT(16)	-383
Valor mínimo positivo de DECFLOAT(16)	1.000000000000000E-383
Valor máximo negativo de DECFLOAT(16)	-1.000000000000000E-383
Exponente máximo ($E_{máx}$) para valores DECFLOAT(34)	6144
Valor mínimo de DECFLOAT(34)[1]	-9.999999999999999999999999999999999E+6144
Valor máximo de DECFLOAT(34)	9.999999999999999999999999999999999E+6144
Exponente mínimo ($E_{mín}$) para valores DECFLOAT(34)	-6143
Valor mínimo positivo de DECFLOAT(34)	1.000000000000000000000000000000000E-6143
Valor máximo negativo de DECFLOAT(34)	-1.000000000000000000000000000000000E-6143

Tabla 65. Límites numéricos (continuación)

Descripción	Límite
Nota: 1. Estos son los límites de los números de coma flotante decimal normal. Entre los valores de coma flotante decimal válidos se incluyen los siguientes: NAN, -NAN, SNAN, -SNAN, INFINITY y -INFINITY. Además, los valores válidos incluyen números anormales. Los números anormales son números distintos de cero cuyos exponentes ajustados son menores que E_{min}. Para un número anormal, el valor mínimo del exponente es E_{min} - (*precisión*-1), llamado E_{tiny}, donde *precisión* es la precisión de trabajo (16 ó 34). Es decir, los números anormales amplían el rango de los números próximos a cero en 15 ó 33 órdenes de magnitud para DECFLOAT(16) o DECFLOAT(34), respectivamente. Los números anormales son diferentes de los números normales debido a que el número máximo de dígitos para un número anormal es inferior a la precisión de trabajo (16 ó 34). La coma flotante decimal no puede representar los números anormales con la misma precisión que puede representar los números normales. El número anormal más pequeño positivo para DECFLOAT(34) es $1x10^{-6176}$, que contiene sólo un dígito, en tanto que el número normal más pequeño positivo para DECFLOAT(34) es $1.000000000000000000000000000000000x10^{-6143}$, que contiene 34 dígitos. El número anormal más pequeño positivo para DECFLOAT(16) es $1x10^{-398}$.	

Tabla 66. Límites de series

Descripción	Límite
Longitud máxima de CHAR (en bytes)	254
Longitud máxima de VARCHAR (en bytes)	32.672
Longitud máxima de LONG VARCHAR (en bytes)[1]	32.700
Longitud máxima de CLOB (en bytes)	2.147.483.647
Longitud máxima de XML serializado (en bytes)	2.147.483.647
Longitud máxima de GRAPHIC (en caracteres de doble byte)	127
Longitud máxima de VARGRAPHIC (en caracteres de doble byte)	16.336
Longitud máxima de LONG VARGRAPHIC (en caracteres de doble byte)[1]	16.350
Longitud máxima de DBCLOB (en caracteres de doble byte)	1.073.741.823
Longitud máxima de BLOB (en bytes)	2.147.483.647
Longitud máxima de constante de caracteres	32.672
Longitud máxima de constante gráfica	16.336
Longitud máxima de series de caracteres concatenadas	2.147.483.647
Longitud máxima de series gráficas concatenadas	1.073.741.823
Longitud máxima de series binarias concatenadas	2.147.483.647
El número máximo de dígitos de constante hexadecimal	32.672
Instancia mayor de un objeto de una columna de tipo estructurado durante la ejecución (en gigabytes)	1
Tamaño máximo de un comentario del catálogo (en bytes)	254
Nota: 1. Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC han quedado obsoletos y se pueden eliminar en un release futuro.	

Tabla 67. Límites de XML

Descripción	Límite
Profundidad máxima de un documento XML (en niveles)	125
Tamaño máximo de un documento de esquema XML (en bytes)	31.457.280

Tabla 68. Límites de fecha y hora

Descripción	Límite
Valor DATE mínimo	0001-01-01
Valor DATE máximo	9999-12-31
Valor TIME mínimo	00:00:00
Valor TIME máximo	24:00:00
Valor TIMESTAMP mínimo	0001-01-01-00.00.00.000000000000
Valor TIMESTAMP máximo	9999-12-31-24.00.00.000000000000
Precisión de indicación de fecha y hora más baja	0
Precisión de indicación de fecha y hora más alta	12

Tabla 69. Límites del gestor de bases de datos

Descripción	Límite
Aplicaciones	
Número máximo de declaraciones de variables del lenguaje principal en un programa precompilado[3]	almacenamiento
Longitud máxima del valor de la variable del lenguaje principal (en bytes)	2.147.483.647
Número máximo de cursores declarados en un programa	almacenamiento
El número máximo de filas cambiadas en una unidad de trabajo	almacenamiento
El número máximo de cursores abiertos a la vez	almacenamiento
El número máximo de conexiones por proceso dentro de un cliente DB2	512
El número máximo de localizadores LOB abiertos simultáneamente en una transacción	4.194.304
Tamaño máximo de una SQLDA (en bytes)	almacenamiento
El número máximo de sentencias preparadas	almacenamiento
Agrupaciones de almacenamientos intermedios	
Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 32 bits	1.048.576
Valor máximo de NPAGES en una agrupación de almacenamientos intermedios para emisiones de 64 bits	2.147.483.647
Tamaño total máximo de todas las ranuras de almacenamientos intermedios de memoria (4K)	2.147.483.646
Simultaneidad	
Número máximo de usuarios simultáneos de un servidor[4]	64.000

Tabla 69. Límites del gestor de bases de datos (continuación)

Descripción	Límite
El número máximo de usuarios simultáneos por instancia	64.000
El número máximo de aplicaciones simultáneas por base de datos	60.000
El número máximo de bases de datos por instancia en uso simultáneo	256
Restricciones	
El número máximo de restricciones en una tabla	almacenamiento
Número máximo de columnas en una restricción UNIQUE (soportado a través de un índice UNIQUE)	64
Longitud máxima combinada de las columnas en una restricción UNIQUE (soportada a través de un índice UNIQUE, en bytes)[9]	8192
Número máximo de columnas de referencia en una clave foránea	64
Longitud máxima combinada de columnas de referencia en una clave foránea (en bytes)[9]	8192
Longitud máxima de una especificación de restricción de comprobación (en bytes)	65.535
Bases de datos	
Número de partición de base de datos máximo	999
Índices	
Número máximo de índices en una tabla	32.767 o almacenamiento
Número máximo de columnas en una clave de índice	64
Longitud máxima de una clave de índice que incluye toda la actividad general[7] [9]	*tamañopáginaíndice*/4
Longitud máxima de una parte de clave de índice variable (en bytes)[8]	1022 o almacenamiento
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas SMS (en gigabytes)[7]	16.384
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas DMS normal (en gigabytes)[7]	512
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas DMS grande (en gigabytes)[7]	16.384
Tamaño máximo de un índice sobre datos XML por partición de base de datos (en terabytes)	2
Longitud máxima de una parte de clave de índice variable de un índice sobre datos XML (en bytes)[7]	*tamañopágina*/4 - 207
Registros de anotación cronológica	
Número de secuencia de anotación cronológica máximo	17.984.000.000.000.000.000
Supervisión	
El número máximo de supervisores de sucesos activos simultáneamente	128
Con DB2 Database Partitioning Feature (DPF), número máximo de supervisores de sucesos GLOBAL activos simultáneamente	32

Tabla 69. Límites del gestor de bases de datos (continuación)

Descripción	Límite
Rutinas	
Número máximo de parámetros en un procedimiento	32.767
Número máximo de parámetros en un constructor de valor de cursor	32.767
Número máximo de parámetros en una función definida por el usuario	90
Número máximo de niveles de anidamiento para rutinas	64
Número máximo de esquemas en la vía de acceso de SQL	64
Longitud máxima de la vía de acceso de SQL (en bytes)	2048
Seguridad	
Número máximo de elementos en un componente de una etiqueta de seguridad de conjunto de tipo o árbol	64
Número máximo de elementos en un componente de una etiqueta de seguridad de matriz de tipos	65.535
Número máximo de componentes de etiqueta de seguridad en una política de seguridad	16
SQL	
Longitud total máxima de una sentencia de SQL (en bytes)	2.097.152
Número máximo de tablas de referencia en una sentencia de SQL o una vista	almacenamiento
Número máximo de referencias de variables del lenguaje principal en una sentencia de SQL	32.767
Número máximo de constantes en una sentencia	almacenamiento
Número máximo de elementos en una lista de selección[7]	1.012
Número máximo de predicados en una cláusula WHERE o HAVING	almacenamiento
Número máximo de columnas en una cláusula GROUP BY[7]	1.012
Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes)[7]	32.677
Número máximo de columnas en una cláusula ORDER BY[7]	1.012
Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes)[7]	32.677
Nivel máximo de anidamiento de subconsultas	almacenamiento
El número máximo de subconsultas en una sola sentencia	almacenamiento
Número máximo de valores en una operación de inserción[7]	1.012
Número máximo de cláusulas SET en una sola operación de actualización[7]	1.012
Tablas y vistas	
Número máximo de columnas en una tabla [7]	1.012
Número máximo de columnas en una vista[1]	5000
El número máximo de columnas en una vista o tabla de fuente de datos a la que se hace referencia mediante un apodo	5000
Número máximo de columnas en una clave de distribución[5]	500

Tabla 69. Límites del gestor de bases de datos (continuación)

Descripción	Límite
Longitud máxima de una fila que incluye toda la actividad general[2 7]	32.677
Número máximo de filas en una tabla no particionada por partición de base de datos	128×10^{10}
Número máximo de filas en una partición de datos por partición de base de datos	128×10^{10}
Tamaño máximo de una tabla por partición de base de datos en un espacio de tablas normal (en gigabytes)[3 7]	512
Tamaño máximo de una tabla por partición de base de datos en un espacio de tablas DMS grande (en gigabytes)[7]	16.384
Número máximo de particiones de datos para una sola tabla	32.767
Número máximo de columnas de particionamiento de tablas	16
Número máximo de campos en un tipo de fila definido por el usuario	1.012
Espacios de tablas	
Tamaño máximo de un objeto LOB (en terabytes)	4
Tamaño máximo de un objeto LF (en terabytes)	2
El número máximo de espacios de tablas en una base de datos	32.768
Número máximo de tablas en un espacio de tablas SMS	65.534
Tamaño máximo de un espacio de tablas DMS normal (en gigabytes) [3 7]	512
Tamaño máximo de un espacio de tablas DMS grande (en terabytes) [3 7]	64
Tamaño máximo de un espacio de tablas DMS temporal (en terabytes) [3 7]	64
Número máximo de objetos de tabla en un espacio de tablas DMS[6]	51.000
Número máximo de vías de acceso de almacenamiento en una base de datos de almacenamiento automática	128
Longitud máxima de una vía de acceso de almacenamiento que se asocia con una base de datos de almacenamiento automática (en bytes)	175
Activadores	
Profundidad máxima en tiempo de ejecución de activadores en cascada	16
Tipos definidos por el usuario	
El número máximo de atributos en un tipo estructurado	4082

Tabla 69. Límites del gestor de bases de datos (continuación)

Descripción	Límite
Nota: 1. Este máximo puede conseguirse utilizando una unión en la sentencia CREATE VIEW. La selección de dicha vista está sujeta al límite del número máximo de elementos de una lista de selección. 2. Los datos reales para las columnas BLOB, CLOB, LONG VARCHAR, DBCLOB y LONG VARGRAPHIC no se incluyen en esta cuenta. Sin embargo, la información acerca de la ubicación de los datos ocupa espacio en la fila. 3. Los números mostrados son límites y aproximaciones arquitectónicos. En la práctica los límites pueden ser menores. 4. El valor actual está controlado por los parámetros de configuración del gestor de bases de datos **conexiones_max** y **coordagents_max**. 5. Es un límite de arquitectura. El límite en la mayoría de las columnas de una clave de índice debe utilizarse como el límite práctico. 6. Los objetos de tabla incluyen datos, índices, columnas LONG VARCHAR o VARGRAPHIC y columnas LOB. Los objetos de tabla que están en el mismo espacio de tablas que los datos de tabla no cuentan como adicionales respecto al límite. No obstante, cada objeto de tabla que está en un espacio de tablas diferente de los datos de tabla representa uno respecto al límite para cada tipo de objeto de tabla por tabla en el espacio de tablas en que reside el objeto de tabla. 7. Para ver los valores relativos al tamaño, consulte la Tabla 70. 8. Está limitado solamente por la clave de índice más larga que incluye toda la actividad general (en bytes). A medida que aumenta el número de partes de claves de índice, disminuye la longitud máxima de cada parte de clave. 9. El máximo puede ser inferior, según las opciones del índice.	

Tabla 70. Límites específicos de tamaño de página del gestor de bases de datos

Descripción	Límite de tamaño de página de 4K	Límite de tamaño de página de 8K	Límite de tamaño de página de 16K	Límite de tamaño de página de 32K
Número máximo de columnas en una tabla	500	1.012	1.012	1.012
Longitud máxima de una fila que incluye toda la actividad general	4.005	8.101	16.293	32.677
Tamaño máximo de una tabla por partición de base de datos en un espacio de tablas normal (en gigabytes)	64	128	256	512
Tamaño máximo de una tabla por partición de base de datos en un espacio de tablas grande (en gigabytes)	2048	4096	8192	16.384
Longitud máxima de la clave de índice que incluye toda la actividad general (en bytes)	1024	2048	4096	8192
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas SMS (en gigabytes)	2048	4096	8192	16.384

Tabla 70. Límites específicos de tamaño de página del gestor de bases de datos (continuación)

Descripción	Límite de tamaño de página de 4K	Límite de tamaño de página de 8K	Límite de tamaño de página de 16K	Límite de tamaño de página de 32K
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas DMS normal (en gigabytes)	64	128	256	512
Tamaño máximo de un índice por partición de base de datos en un espacio de tablas DMS grande (en gigabytes)	2048	4096	8192	16.384
Tamaño máximo de un índice sobre datos XML por partición de base de datos (en terabytes)	2	2	2	2
Tamaño máximo de un espacio de tablas DMS normal (en gigabytes)	64	128	256	512
Tamaño máximo de un espacio de tablas DMS grande (en terabytes)	8	16	32	64
Tamaño máximo de un espacio de tablas DMS temporal (en terabytes)	8	16	32	64
Número máximo de elementos en una lista de selección	500	1.012	1.012	1.012
El número máximo de columnas en una cláusula GROUP BY	500	1.012	1.012	1.012
Longitud total máxima de las columnas en una cláusula GROUP BY (en bytes)	4.005	8.101	16.293	32.677
El número máximo de columnas en una cláusula ORDER BY	500	1.012	1.012	1.012
Longitud total máxima de las columnas en una cláusula ORDER BY (en bytes)	4.005	8.101	16.293	32.677
Número máximo de valores de una operación de inserción	500	1.012	1.012	1.012
Número máximo de cláusulas SET en una sola sentencia de actualización	500	1.012	1.012	1.012

Tabla 70. Límites específicos de tamaño de página del gestor de bases de datos (continuación)

Descripción	Límite de tamaño de página de 4K	Límite de tamaño de página de 8K	Límite de tamaño de página de 16K	Límite de tamaño de página de 32K
Registros máximos por página para un espacio de tablas normal	251	253	254	253
Registros máximos por página para un espacio de tablas grande	287	580	1165	2335

Apéndice B. SQLCA (área de comunicaciones SQL)

Una SQLCA es un conjunto de variables que se actualiza al final de la ejecución de cada sentencia de SQL. Un programa que contiene sentencias de SQL ejecutables y se precompila con la opción LANGLEVEL SAA1 (el valor por omisión) o MIA debe proporcionar exactamente una SQLCA, aunque es posible que exista más de una SQLCA por paso en una aplicación de múltiples pasos.

Cuando se precompila un programa con la opción LANGLEVEL SQL92E, puede declararse una variable SQLCODE o SQLSTATE en la sección de declaración SQL o se puede declarar una variable SQLCODE en algún otro lugar del programa.

No se debe proporcionar ninguna SQLCA cuando se utiliza LANGLEVEL SQL92E. La sentencia de SQL INCLUDE puede utilizarse para proporcionar la declaración de la SQLCA en todos los lenguajes excepto en REXX. La SQLCA se proporciona automáticamente en REXX.

Para visualizar la SQLCA después de cada mandato ejecutado a través del procesador de línea de mandatos, utilice el mandato db2 -a. La SQLCA se proporciona como parte de la salida para los mandatos posteriores. La SQLCA también se vuelca en el archivo de anotaciones cronológicas db2diag.

Descripciones de los campos de la SQLCA

Tabla 71. Campos de la SQLCA. Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlcaid	CHAR(8)	Una "indicación visual" para los vuelcos de almacenamiento que contienen la 'SQLCA'. El sexto byte es 'L' si el número de línea devuelto procede del análisis sintáctico del cuerpo de un procedimiento SQL.
sqlcabc	INTEGER	Contiene la longitud de la SQLCA, 136.
sqlcode	INTEGER	Contiene el código de retorno SQL. **Código** **Significado** **0** Ejecución satisfactoria (aunque pueden haberse establecido uno o varios indicadores SQLWARN). **positivo** Ejecución satisfactoria, pero con una condición de aviso. **negativo** Condición de error.
sqlerrml	SMALLINT	Indicador de longitud para *sqlerrmc*, en el rango de 0 a 70. 0 significa que el valor de *sqlerrmc* no es relevante.

Tabla 71. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrmc	VARCHAR (70)	Contiene uno o más símbolos, separados por X'FF', que se sustituyen por variables en las descripciones de condiciones de errores. Este campo también se utiliza cuando se establece una conexión satisfactoria. Cuando se emite una sentencia de SQL compuesto NOT ATOMIC, puede contener información acerca de un máximo de siete errores. El último símbolo puede ir seguido de X'FF'. El valor *sqlerrml* incluirá cualquier X'FF' de cola.
sqlerrp	CHAR(8)	Empieza con un identificador de tres letras que indica el producto, seguido de cinco caracteres alfanuméricos indicando la versión, release y nivel de modificación del producto. Los caracteres A-Z indican un nivel de modificación superior a 9. A indica el nivel de modificación 10, B indica el nivel de modificación 11, y así sucesivamente. Por ejemplo, SQL0907C corresponde a DB2 Versión 9, release 7, nivel de modificación 12). Si SQLCODE indica una condición de error, este campo identifica el módulo que ha devuelto el error. Este campo también se utiliza cuando se establece una conexión satisfactoria.
sqlerrd	ARRAY	Seis variables INTEGER que proporcionan información de diagnóstico. Generalmente, estos valores están vacíos si no hay errores, excepto sqlerrd(6) de una base de datos particionada.
sqlerrd(1)	INTEGER	Si se invoca la conexión y es satisfactoria, contiene la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la base de datos de la página de códigos de la aplicación. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción. Cuando un procedimiento SQL termina satisfactoriamente su ejecución, este campo contiene el valor del estado de devolución del procedimiento SQL.
sqlerrd(2)	INTEGER	Si se invoca la conexión y es satisfactoria, contiene la diferencia máxima esperada en la longitud de datos de caracteres mixtos (tipos de datos CHAR) cuando se convierten a la página de códigos de la aplicación 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. Si la SQLCA es el resultado de una sentencia de SQL compuesto NOT ATOMIC que ha encontrado uno o varios errores, el valor se establece en el número de sentencias que han fallado.

Tabla 71. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrd(3)	INTEGER	Si se invoca PREPARE y la operación es satisfactoria, contiene una estimación del número de filas que se devolverán. Después de INSERT, UPDATE, DELETE o MERGE, contiene el número real de filas que estaban cualificadas para la operación. Para una sentencia TRUNCATE, el valor será -1. Si se invoca SQL compuesto, contiene una acumulación de todas las filas de subsentencia. Si se invoca CONNECT, contiene 1 si la base de datos se puede actualizar, o 2 si la base de datos es de sólo lectura. Si se invoca la sentencia OPEN y el cursor contiene sentencias de cambio de datos de SQL, este campo contiene la suma del número de filas cualificadas para las operaciones incorporadas de inserción, actualización, supresión o fusión. Si se invoca CREATE PROCEDURE para un procedimiento SQL y se detecta un error durante el análisis del cuerpo del procedimiento SQL, contiene el número de línea donde se encontró el error. El sexto byte de sqlcaid debe ser 'L' para que este valor sea un número de línea válido.
sqlerrd(4)	INTEGER	Si se invoca PREPARE y es satisfactoria, contiene una estimación del coste relativo de los recursos necesarios para procesar la sentencia. Si se invoca SQL compuesto, contiene una cuenta del número de subsentencias satisfactorias. Si se invoca CONNECT, contiene 0 para una confirmación en una fase de un cliente de nivel inferior; 1 para una confirmación de una fase; 2 para confirmación de sólo lectura de una fase; y 3 para una confirmación de dos fases.
sqlerrd(5)	INTEGER	Contiene el número total de filas suprimidas, insertadas o actualizadas como resultado de: • La imposición de las restricciones después de una operación de supresión satisfactoria • El proceso de sentencias de SQL activadas por activadores activados. Si se invoca SQL compuesto, contiene una acumulación del número de dichas filas para todas las subsentencias. En algunos casos cuando se encuentra un error, este campo contiene un valor negativo que es un puntero de un error interno. Si se invoca CONNECT, contiene el valor de tipo de autentificación 0 para una autentificación de servidor; 1 para la autentificación de cliente; 2 para la autentificación utilizando DB2 Connect; 4 para autentificación de SERVER_ENCRYPT; 5 para la autentificación utilizando DB2 Connect con cifrado; 7 para la autentificación de KERBEROS; 9 para la autentificación de GSSPLUGIN ; 11 para la autentificación de DATA_ENCRYPT y 255 para la autentificación no especificada.

Tabla 71. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlerrd(6)	INTEGER	Para una base de datos particionada, contiene el número de partición de la partición de base de datos que ha encontrado el error o aviso. Si no se han encontrado errores ni avisos, este campo contiene el número de partición del coordinador. El número de este campo es igual al especificado para la partición de base de datos del archivo db2nodes.cfg.
sqlwarn	Matriz	Un conjunto de indicadores de aviso, que contiene cada uno un blanco o W. Si se invoca SQL compuesto, contiene una acumulación de indicadores de aviso establecidos para todas las subsentencias.
sqlwarn0	CHAR(1)	Espacio en blanco si todos los demás indicadores están en blanco; contiene una 'W' si como mínimo otro indicador no está en blanco.
sqlwarn1	CHAR(1)	Contiene una 'W' si el valor de una columna de serie se ha truncado cuando se ha asignado a una variable del lenguaje principal. Contiene una 'N' si el terminador nulo se ha truncado. Contiene una 'A' si la operación CONNECT o ATTACH se realiza satisfactoriamente y el nombre de autorización de la conexión tiene más de 8 bytes. Contiene una 'P' si el coste estimado relativo de la sentencia PREPARE de sqlerrd(4) sobrepasaba el valor que podía almacenarse en un INTEGER o era menor que 1 y el registro especial CURRENT EXPLAIN MODE o CURRENT EXPLAIN SNAPSHOT se ha establecido en un valor distinto de NO.
sqlwarn2	CHAR(1)	Contiene 'W' si los valores nulos se eliminaron del argumento de una función agregada. [a] Si se invoca CONNECT y se ejecuta satisfactoriamente, contiene una 'D' si la base de datos se encuentra inmovilizada o 'I' si la instancia se encuentra inmovilizada.
sqlwarn3	CHAR(1)	Contiene una 'W' si el número de columnas no es igual al número de variables del lenguaje principal. Contiene una 'Z' si el número de localizadores del conjunto de resultados especificado en la sentencia ASSOCIATE LOCATORS es menor que el número de conjuntos de resultados devuelto por un procedimiento.
sqlwarn4	CHAR(1)	Contiene una 'W' si una sentencia UPDATE o DELETE preparada no incluye una cláusula WHERE.
sqlwarn5	CHAR(1)	Contiene una 'E' si se ha tolerado un error durante la ejecución de la sentencia de SQL.
sqlwarn6	CHAR(1)	Contiene una 'W' si se ha ajustado el resultado del cálculo de una fecha para evitar una fecha imposible.
sqlwarn7	CHAR(1)	Reservado para una utilización futura. Si se invoca CONNECT y se ejecuta satisfactoriamente, contiene una 'E' si el parámetro de configuración **dyn_query_mgmt** de la base de datos está habilitado.

Tabla 71. Campos de la SQLCA (continuación). Los nombres de los campos que se muestran son aquellos presentes en una SQLCA obtenida mediante una sentencia INCLUDE.

Nombre	Tipo de datos	Valores de campos
sqlwarn8	CHAR(1)	Contiene una 'W' si el carácter que no se ha podido convertir se ha sustituido por un carácter de sustitución. Contiene 'Y' si ha habido un intento no satisfactorio de establecer una conexión fiable.
sqlwarn9	CHAR(1)	Contiene 'W' si se han ignorado expresiones aritméticas con errores durante el proceso de la función agregada.
sqlwarn10	CHAR(1)	Contiene una 'W' si ha habido un error de conversión al convertir un valor de datos de caracteres de uno de los campos de la SQLCA.
sqlstate	CHAR(5)	Un código de retorno que indica el resultado de la sentencia de SQL ejecutada más recientemente.

[a] Es posible que algunas funciones no establezcan SQLWARN2 en W aunque se hayan eliminado los valores nulos, porque el resultado no dependía de la eliminación de dichos valores.

Informe de errores

El orden en que se informa de los errores es el siguiente:

1. Las condiciones de error graves siempre se informan. Cuando se informa de un error grave, no hay ninguna adición a la SQLCA.
2. Si no se produce ningún error grave, un error de punto muerto tiene prioridad sobre los demás errores.
3. En el resto de errores, se devuelve la SQLCA para el primer código SQL negativo.
4. Si no se detecta ningún código SQL negativo, se devuelve la SQLCA para el primer aviso (es decir, código SQL positivo).

   En un sistema de base de datos particionada, se produce una excepción a esta norma si se invoca una operación de manipulación de datos en una tabla que está vacía en una partición de base de datos, pero que tiene datos en otras particiones de base de datos. Sólo se devuelve SQLCODE +100 a la aplicación si los agentes de todas las particiones devuelven SQL0100W, porque la tabla está vacía en todas las particiones de base de datos o porque no hay más filas que cumplan la cláusula WHERE de una sentencia UPDATE.

Utilización de SQLCA en sistemas de bases de datos particionadas

En sistemas de bases de datos particionadas, una sentencia de SQL pueden ejecutarla varios agentes de distintas particiones de base de datos y cada agente puede devolver una SQLCA distinta para diferentes errores o avisos. El agente coordinador también tiene su propia SQLCA.

Para proporcionar una vista coherente para las aplicaciones, todos los valores de SQLCA se fusionan en una estructura y los campos de SQLCA indican cuentas globales como, por ejemplo:

- Para todos los errores y avisos, el campo *sqlwarn* contiene los distintivos de aviso recibidos de todos los agentes.

- Los valores de los campos *sqlerrd* que indican cuentas de fila son acumulaciones de todos los agentes.

Observe que es posible que no se devuelva SQLSTATE 09000 cada vez que se produzca un error durante el proceso de una sentencia de SQL activada.

Apéndice C. SQLDA (área de descriptores de SQL)

Una SQLDA es un conjunto de variables que son necesarias para la ejecución de la sentencia de SQL DESCRIBE. Las variables SQLDA son opciones que las sentencias PREPARE, OPEN, FETCH y EXECUTE pueden utilizar. Una SQLDA se comunica con SQL dinámico; puede utilizarse en una sentencia DESCRIBE, modificarse con las direcciones de las variables del lenguaje principal y después volverse a utilizar en una sentencia FETCH o EXECUTE.

Se da soporte a SQLDA para todos los lenguajes, pero sólo se proporcionan declaraciones predefinidas para C, REXX, FORTRAN y COBOL.

El significado de la información de una SQLDA depende de su utilización. En PREPARE y DESCRIBE, una SQLDA proporciona información para un programa de aplicación acerca de una sentencia preparada. En OPEN, EXECUTE y FETCH, una SQLDA describe las variables del lenguaje principal.

En DESCRIBE y PREPARE, si cualquiera de las columnas que se describen es de tipo LOB (los localizadores de LOB y las variables de referencia de archivo no necesitan doblar las SQLDA), de tipo de referencia o un tipo definido por el usuario, el número de entradas de SQLVAR para la SQLDA completa se doblará. Por ejemplo:

- Cuando se describe una tabla con 3 columnas VARCHAR y 1 columna INTEGER, habrán 4 entradas SQLVAR
- Cuando se describe una tabla con 2 columnas VARCHAR, 1 columna CLOB y 1 columna de enteros, habrán 8 entradas SQLVAR

En EXECUTE, FETCH y OPEN, si cualquiera de las variables que se describen es de tipo LOB (los localizadores de LOB y las variables de referencia de archivos no necesitan doblar las SQLDA), de tipo estructurado, el número de entradas de SQLVAR para la SQLDA completa debe doblarse. (Estos casos no son aplicables a los tipos diferenciados ni a los tipos de referencia, porque la base de datos no necesita la información adicional que proporcionan las entradas dobles No se ofrece soporte para los tipos de matriz, cursor y fila como variables SQLDA en sentencias EXECUTE, FETCH y OPEN.)

Descripción de los campos de la SQLDA

Una SQLDA consta de cuatro variables seguidas de un número arbitrario de apariciones de una secuencia de variables llamadas en conjunto SQLVAR. En OPEN, FETCH y EXECUTE, cada aparición de SQLVAR describe una variable del lenguaje principal. En DESCRIBE y PREPARE, cada aparición de SQLVAR describe una columna de una tabla resultante o un marcador de parámetro. Hay dos tipos de entradas SQLVAR:

- **SQLVAR base:** Estas entradas siempre están presentes. Contienen la información base acerca de la columna, el marcador de parámetro o la variable del lenguaje principal como, por ejemplo, el código del tipo de datos, el atributo de longitud, el nombre de la columna, la dirección de la variable del lenguaje principal y la dirección de la variable de indicador.
- **SQLVAR secundarias:** Estas entradas sólo están presentes si el número de entradas SQLVAR se dobla por las normas indicadas arriba. Para los tipos definidos por el usuario (excluidos los tipos de referencia), contienen el nombre

del tipo definido por el usuario. Para los tipos de referencia, contienen el tipo de destino de la referencia. Para los LOB, contienen el atributo de longitud de la variable del lenguaje principal y un puntero que indica el almacenamiento intermedio que contiene la longitud real. (La información de tipo diferenciado y de LOB no se solapa, por lo que los tipos diferenciados pueden estar basados en LOB sin hacer que se triplique el número de entradas de SQLVAR en DESCRIBE). Si se utilizan localizadores o variables de referencia a archivos para representar los LOB, estas entradas no son necesarias.

En las SQLDA que contienen ambos tipos de entradas, las SQLVAR base están en un bloque antes del bloque de SQLVAR secundarias. En cada una, el número de entradas es igual al valor de SQLD (incluso aunque muchas de las entradas SQLVAR secundarias pueden estar sin utilizar).

Las circunstancias bajo las que DESCRIBE establece las entradas SQLVAR se detallan en el apartado “Efecto de DESCRIBE en la SQLDA” en la página 808.

Campos en la cabecera SQLDA

Tabla 72. Campos en la cabecera SQLDA

Nombre C	Tipo de datos SQL	Uso en DESCRIBE y PREPARE (establecido por el gestor de bases de datos excepto para SQLN)	Uso en FETCH, OPEN y EXECUTE (establecido por la aplicación antes de ejecutar la sentencia)
sqldaid	CHAR(8)	El séptimo byte de este campo es un byte de distintivo llamado SQLDOUBLED. El gestor de bases de datos establece SQLDOUBLED en el carácter ’2’ si se han creado dos entradas SQLVAR para cada columna; de lo contrario, se establece en un blanco (X’20’ en ASCII, X’40’ en EBCDIC). Consulte el apartado “Efecto de DESCRIBE en la SQLDA” en la página 808 para ver los detalles de cuándo se establece SQLDOUBLED.	El séptimo byte de este campo se utiliza cuando se dobla el número de SQLVAR. Se denomina SQLDOUBLED. Si alguna de las variables del lenguaje principal que se describen es un tipo estructurado, BLOB, CLOB o DBCLOB, el séptimo byte debe establecerse en el carácter ’2’; en otro caso puede establecerse en cualquier carácter, pero es aconsejable utilizar un espacio en blanco.
sqldabc	INTEGER	Para 32 bits, la longitud de la SQLDA, que es igual a SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es igual a SQLN*56+16	Para 32 bits, la longitud de la SQLDA, que es >= SQLN*44+16. Para 64 bits, la longitud de la SQLDA, que es >= SQLN*56+16.
sqln	SMALLINT	Sin cambiar por el gestor de bases de datos. Debe estar establecido en un valor mayor o igual que cero antes de que se ejecute la sentencia DESCRIBE. Indica el número total de apariciones de SQLVAR.	El número total de apariciones de SQLVAR proporcionadas en la SQLDA. SQLN debe estar establecido en un valor mayor o igual que cero.
sqld	SMALLINT	Establecido por el gestor de bases de datos en el número de columnas de la tabla resultante o en el número de marcadores de parámetro.	El número de variables del lenguaje principal descritas por las apariciones de SQLVAR.

Campos de una aparición de una SQLVAR base

Tabla 73. Campos en una SQLVAR base

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
sqltype	SMALLINT	Indica el tipo de datos de la columna o el marcador de parámetro y si puede contener nulos. (Los marcadores de parámetro siempre se consideran anulables.) La Tabla 75 en la página 810 lista todos los valores permitidos y sus significados. Observe que para un tipo diferenciado de matriz, cursor, fila o referencia, el tipo de datos del tipo base se coloca en este campo. Para un tipo estructurado, el tipo de datos del resultado de la función de transformación FROM SQL del grupo de transformación (basado en el registro especial CURRENT DEFAULT TRANSFORM GROUP) se coloca en este campo. No existe ninguna indicación en la SQLVAR base de que forma parte de la descripción de un tipo definido por el usuario o tipo de referencia.	Igual que para la variable del lenguaje principal. Las variables del lenguaje principal para los valores de fecha y hora deben ser variables de serie de caracteres. Para FETCH, un código de tipo de fecha y hora significa una serie de caracteres de longitud fija. Si sqltype es un valor de número par, se ignora el campo sqlind.
sqllen	SMALLINT	El atributo de longitud de la columna o el marcador de parámetro. Para columnas y marcadores de parámetro de fecha y hora, la longitud de la representación de serie de los valores. Consulte la Tabla 75 en la página 810. Observe que el valor está establecido en 0 para las series de objeto grande (incluso para aquellas cuyo atributo de longitud sea lo suficientemente pequeño como para caber en un entero de dos bytes).	El atributo de longitud de la variable del lenguaje principal. Consulte la Tabla 75 en la página 810. Observe que el gestor de bases de datos ignora el valor para las columnas CLOB, DBCLOB y BLOB. Se utiliza el campo len.sqllonglen de la SQLVAR secundaria en su lugar.
sqldata	puntero	En las SQLVAR de series, sqldata contiene la página de códigos. En las SQLVAR de serie-caracteres, si la columna está definida con el atributo FOR BIT DATA, sqldata contiene 0. En otras SQLVAR de series-caracteres, sqldata contiene la página de códigos SBCS, para datos SBCS, o la página de códigos SBCS asociada a la página de códigos MBCS compuesta, para datos MBCS. Para las SQLVAR de series de caracteres de japonés EUC, chino tradicional EUC e Unicode UTF-8,sqldata contiene 954, 964 y 1208, respectivamente. Para todos los tipos de columna, sqldata es indefinido.	Contiene la dirección de la variable del lenguaje principal (donde se almacenarán los datos leídos).

Tabla 73. Campos en una SQLVAR base (continuación)

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
sqlind	puntero	En las SQLVAR de series-caracteres, sqlind contiene 0, excepto para los datos MBCS, si sqlind contiene la página de códigos DBCS asociada con la página de códigos MBCS compuesta. Para el resto de tipos, sqlind no está definido.	Contiene la dirección de una variable de indicador asociada si existe una; de lo contrario, no se utiliza. Si sqltype es un valor de número par, se ignora el campo sqlind.
sqlname	VARCHAR (30)	Contiene el nombre no calificado de la columna o el marcador de parámetro. Para columnas y marcadores de parámetro que tengan el nombre generado por el sistema, el byte número trece se establece en X'FF'. Para los nombres de columna especificados por la cláusula AS, el byte es X'00'.	Cuando se conecta a una base de datos del sistema principal, se puede establecer sqlname para que indique una serie FOR BIT DATA de la forma siguiente: • El sexto byte del SQLDAID en la cabecera SQLDA se ha establecido en '+' • La longitud de sqlname es 8 • Los dos primeros bytes de sqlname son X'0000' • El tercer y cuarto byte de sqlname son X'0000' • Los cuatro bytes restantes de sqlname están reservados y deben establecerse en X'00000000' Cuando se trabaja con datos XML, se puede establecer sqlname para que indique un subtipo XML de la forma siguiente: • La longitud de sqlname es 8 • Los dos primeros bytes de sqlname son X'0000' • El tercer y cuarto byte de sqlname son X'0000' • El quinto byte de sqlname es X'01' • Los tres bytes restantes de sqlname están reservados y deben establecerse en X'000000'

Campos de una aparición de una SQLVAR secundaria

Tabla 74. Campos en una SQLVAR secundaria

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
len.sqllonglen	INTEGER	El atributo de longitud de una columna o marcador de parámetro de BLOB, CLOB o DBCLOB.	El atributo de longitud de una variable del lenguaje principal BLOB, CLOB o DBCLOB. El gestor de bases de datos pasa por alto el campo SQLLEN de la SQLVAR base para los tipos de datos. El atributo de longitud almacena el número de bytes para BLOB o CLOB y el número de caracteres de doble byte para DBCLOB.

Tabla 74. Campos en una SQLVAR secundaria (continuación)

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE
reservado2	CHAR(3) para 32 bits, y CHAR(11) para 64 bits.	No se utiliza.	No se utiliza.
sqlflag4	CHAR(1)	El valor es X'01' si la SQLVAR representa un tipo de referencia con un tipo de destino denominado en sqldatatype_name. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipodatosSQL contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'.	Se establece en X'01' si la SQLVAR representa un tipo de referencia con un tipo de destino mencionado en nombre_tipodatosSQL. El valor es X'12' si SQLVAR representa un tipo estructurado, y nombre_tipodatosSQL contiene el nombre del tipo definido por el usuario. En otro caso, el valor es X'00'.
sqldatalen	puntero	No se utiliza.	Utilizado solamente para las variables del lenguaje principal BLOB, CLOB y DBCLOB. Si este campo es NULL, entonces la longitud real (en caracteres de doble byte) debe almacenarse en los 4 bytes inmediatamente antes del inicio de los datos y SQLDATA debe apuntar al primer byte de la longitud de campo. Si este campo no es NULL, contiene un puntero que indica un almacenamiento intermedio de 4 bytes de longitud que contiene la longitud real *en bytes* (incluso para DBCLOB) de los datos del almacenamiento intermedio al que apunta el campo de SQLDATA de la SQLVAR base coincidente. Observe que, sin tener en cuenta si este campo se utiliza o no, debe establecerse el campo len.sqllonglen.
sqldatatype_name	VARCHAR(27)	Para un tipo definido por el usuario, el gestor de bases de datos establece este campo en el nombre del tipo definido por el usuario, calificado al completo.[1] Para un tipo de referencia, el gestor de bases de datos establece este campo en el nombre del tipo calificado al completo del tipo de destino de la referencia.	Para los tipos estructurados, se establece en el nombre del tipo definido por el usuario, calificado al completo, con el formato indicado en la nota de la tabla.[1]
reserved	CHAR(3)	No se utiliza.	No se utiliza.

Tabla 74. Campos en una SQLVAR secundaria (continuación)

Nombre	Tipo de datos	Uso en DESCRIBE y PREPARE	Uso en FETCH, OPEN y EXECUTE

[1] Los 8 primeros bytes contienen el nombre de esquema del tipo (ampliado por la derecha con espacios, si es necesario). El byte 9 contiene un punto (.). Los bytes del 10 al 27 contienen la parte de orden inferior del nombre de tipo, que *no* se extiende por la derecha con espacios.

Tenga en cuenta que, aunque el principal propósito de este campo es para el nombre de los tipos definidos por el usuario, el campo también se establece para los tipos de datos predefinidos por IBM. En este caso, el nombre de esquema es SYSIBM y la parte de orden inferior del nombre es el nombre almacenado en la columna TYPENAME de la vista de catálogo DATATYPES. Por ejemplo:

```
nombre tipo      longit   sqldatatype_name
---------        ------   ----------------
A.B              10       A       .B
INTEGER          16       SYSIBM  .INTEGER
"Frank's".SMINT  13       Frank's .SMINT
MY."type  "      15       MY      .type
```

Efecto de DESCRIBE en la SQLDA

Para una sentencia DESCRIBE OUTPUT o PREPARE OUTPUT INTO, el gestor de bases de datos siempre establece SQLD en el número de columnas del conjunto resultante o en el número de marcadores de parámetro de la salida. Para una sentencia DESCRIBE INPUT o PREPARE INPUT INTO, el gestor de bases de datos siempre establece SQLD en el número de marcadores de parámetro de entrada de la sentencia. Observe que un marcador de parámetro que se corresponda con un parámetro INOUT en una sentencia CALL se describe en los descriptores tanto de entrada como de salida.

Las SQLVAR de la SQLDA se establecen en los casos siguientes:

- SQLN >= SQLD y ninguna entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco.
- SQLN >= 2*SQLD y como mínimo una entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  Las entradas SQLD SQLVAR se establecen dos veces y SQLDOUBLED se establece en '2'.
- SQLD <= SQLN < 2*SQLD y como mínimo una entrada es un tipo diferenciado de matriz, cursor, fila o referencia, pero no hay entradas LOB ni entradas de tipo estructurado

  Se establecen las primeras entradas SQLD SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +237 (SQLSTATE 01594).

Las SQLVAR de la SQLDA NO se establecen (es necesaria la asignación de espacio adicional y otra DESCRIBE) en los casos siguientes:

- SQLN < SQLD y ninguna entrada es un LOB, un tipo definido por el usuario o un tipo de referencia

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +236 (SQLSTATE 01005).

  Asigne las SQLD SQLVAR para una operación DESCRIBE satisfactoria.

- SQLN < SQLD y como mínimo una entrada es un tipo diferenciado de matriz, cursor, fila o referencia, pero no hay entradas LOB ni entradas de tipo estructurado

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Si la opción SQLWARN es YES, se emite un aviso SQLCODE +239 (SQLSTATE 01005).

  Asigne un número de estructuras SQLVAR igual a 2*SQLD para lograr una operación DESCRIBE satisfactoria que incluya los nombres de los tipos diferenciados de matriz, cursor y filay tipos de destino de tipos de referencia.
- SQLN < 2*SQLD y como mínimo una entrada es un LOB o un tipo estructurado

  No se establece ninguna entrada SQLVAR y SQLDOUBLED se establece en blanco. Se emite un aviso SQLCODE +238 (SQLSTATE 01005) (sin tener en cuenta el valor de la opción de vinculación SQLWARN).

  Asigne las 2*SQLD SQLVAR para una operación DESCRIBE satisfactoria.

Las referencias de las listas anteriores a entradas LOB incluyen las entradas de tipo diferenciado cuyo tipo fuente es un tipo LOB.

La opción SQLWARN del mandato BIND o PREP se utiliza para controlar si DESCRIBE (o PREPARE INTO) devolverá los SQLCODE de aviso +236, +237, +239. Se recomienda que el código de la aplicación tenga siempre en cuenta que podrían devolverse estos SQLCODE. El SQLCODE de aviso +238 siempre se devuelve cuando hay entradas LOB o de tipo estructurado en la lista de selección y no hay suficientes SQLVAR en la SQLDA. Es la única manera de que la aplicación pueda saber el número de estructuras SQLVAR que deben doblarse debido a que la existencia de una entrada LOB o de tipo estructurado en el conjunto resultante.

Si se está describiendo una entrada de 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 42741) o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744)), DESCRIBE devolverá un error. Este error es el mismo que se devuelve para una estructura DESCRIBE de una tabla con una entrada de tipo estructurado.

Si el gestor de base de datos devuelve identificadores que son más largos que los que pueden almacenarse en el SQLDA, el identificador se truncará y se devolverá un aviso (SQLSTATE 01665); sin embargo, cuando se trunca el nombre de un tipo estructurado, se devuelve un error (SQLSTATE 42622). Para obtener más detalles sobre las limitaciones en la longitud del identificador, consulte el apartado “Límites de SQL y XQuery” .

SQLTYPE y SQLLEN

La Tabla 75 en la página 810 muestra los valores que pueden aparecer en los campos SQLTYPE y SQLLEN de la SQLDA. En DESCRIBE y PREPARE INTO, un valor par de SQLTYPE significa que la columna no permite nulos y un valor impar significa que la columna permite nulos. En FETCH, OPEN y EXECUTE, un valor par de SQLTYPE significa que no se proporciona una variable de indicador y un valor impar significa que SQLIND contiene la dirección de una variable de indicador.

Tabla 75. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN y EXECUTE

	Para DESCRIBE y PREPARE INTO		Para FETCH, OPEN y EXECUTE	
SQLTYPE	**Tipo de datos de columna**	**SQLLEN**	**Tipo de datos de variable del lenguaje principal**	**SQLLEN**
384/385	fecha	10	representación de una fecha en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
388/389	hora	8	representación de una hora en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
392/393	indicación de fecha y hora	19 para TIMESTAMP(0), si no 20+*p* para TIMESTAMP(*p*)	representación de una indicación de fecha y hora en serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
400/401	N/D	N/D	serie gráfica de terminación nula	atributo de longitud de la variable del lenguaje principal
404/405	BLOB	0 *	BLOB	No se utiliza. *
408/409	CLOB	0 *	CLOB	No se utiliza. *
412/413	DBCLOB	0 *	DBCLOB	No se utiliza. *
448/449	serie de caracteres de longitud variable	atributo de longitud de la columna	serie de caracteres de longitud variable	atributo de longitud de la variable del lenguaje principal
452/453	serie de caracteres de longitud fija	atributo de longitud de la columna	serie de caracteres de longitud fija	atributo de longitud de la variable del lenguaje principal
456/457	serie larga de caracteres de longitud variable	atributo de longitud de la columna	serie larga de caracteres de longitud variable	atributo de longitud de la variable del lenguaje principal
460/461	no aplicable	no aplicable	serie de caracteres de terminación nula	atributo de longitud de la variable del lenguaje principal
464/465	serie gráfica de longitud variable	atributo de longitud de la columna	serie gráfica de longitud variable	atributo de longitud de la variable del lenguaje principal
468/469	serie gráfica de longitud fija	atributo de longitud de la columna	serie gráfica de longitud fija	atributo de longitud de la variable del lenguaje principal
472/473	serie gráfica de longitud variable larga	atributo de longitud de la columna	serie gráfica larga	atributo de longitud de la variable del lenguaje principal
480/481	coma flotante	8 para precisión doble, 4 para precisión simple	coma flotante	8 para precisión doble, 4 para precisión simple
484/485	decimal empaquetado	precisión en byte 1; escala en byte 2	decimal empaquetado	precisión en byte 1; escala en byte 2
492/493	entero superior	8	entero superior	8

Tabla 75. Valores SQLTYPE y SQLLEN para DESCRIBE, FETCH, OPEN y EXECUTE (continuación)

	Para DESCRIBE y PREPARE INTO		Para FETCH, OPEN y EXECUTE	
SQLTYPE	**Tipo de datos de columna**	**SQLLEN**	**Tipo de datos de variable del lenguaje principal**	**SQLLEN**
496/497	entero grande	4	entero grande	4
500/501	entero pequeño	2	entero pequeño	2
916/917	no aplicable	no aplicable	variable de referencia a archivos BLOB	267
920/921	no aplicable	no aplicable	variable de referencia a archivos CLOB	267
924/925	no aplicable	no aplicable	variable de referencia a archivos DBCLOB.	267
960/961	no aplicable	no aplicable	localizador de BLOB	4
964/965	no aplicable	no aplicable	localizador de CLOB	4
968/969	no aplicable	no aplicable	localizador de DBCLOB	4
988/989	XML	0	no aplicable; en su lugar, utilice una variable del lenguaje principal XML AS <serie o binaria de tipo LOB>	no utilizado
996	coma flotante decimal	8 para DECFLOAT(16), 16 para DECFLOAT(34)	coma flotante decimal	8 para DECFLOAT(16), 16 para DECFLOAT(34)
2440/2441	fila	no aplicable	fila	no utilizado
2440/2441	cursor	no aplicable	fila	no utilizado

Nota:

- El campo len.sqllonglen de la SQLVAR secundaria contiene el atributo de longitud de la columna.
- SQLTYPE se ha modificado desde la versión anterior para portabilidad en DB2. Los valores de las versiones anteriores (consulte Referencia SQL de la versión anterior) seguirán siendo soportados.

SQLTYPE no reconocidos y no soportados

Los valores que aparecen en el campo SQLTYPE de SQLDA dependen del nivel de soporte de tipo de datos disponible en el encargado de enviar los datos así como en el que los recibe. Esto es especialmente importante cuando se añaden nuevos tipos de datos al producto.

Los tipos de datos nuevos pueden recibir o no soporte del que envía los datos o del que los recibe y pueden estar reconocidos o incluso no estarlo por el que envía los datos o el que los recibe. Según la situación, puede devolverse el tipo de datos nuevo o puede devolverse un tipo de datos compatible con el acuerdo del que envía los datos y del que los recibe o bien puede dar como resultado un error.

Cuando el que envía los datos y el que los recibe se ponen de acuerdo para utilizar un tipo de datos compatible, la indicación siguiente expresa la correlación que tendrá lugar. Esta correlación tendrá lugar cuando, como mínimo, o el que envía los datos o el que los recibe no dé soporte al tipo de datos proporcionado. Tanto la aplicación como el gestor de bases de datos pueden proporcionar el tipo de datos

no soportado.

Tipo de datos	Tipo de datos compatible
BIGINT	DECIMAL(19, 0)
ROWID[1]	VARCHAR(40) FOR BIT DATA

[1] ROWID está soportado por DB2 Universal Database para z/OS Versión 8.

Tenga en cuenta que en SQLDA no se proporciona ninguna indicación de que se sustituya el tipo de datos.

Números decimales empaquetados

Los números decimales empaquetados se almacenan en una variación de la notación Decimal codificado en binario (BCD). En BCD, cada nybble (cuatro bits) representa un dígito decimal. Por ejemplo, 0001 0111 1001 representa 179. Por lo tanto, se lee un valor decimal empaquetado nybble a nybble. Se almacena el valor en bytes y después se lee estos bytes en representación hexadecimal para volver a decimal. Por ejemplo, 0001 0111 1001 se convierte en 00000001 01111001 en la representación binaria. Leyendo este número como hexadecimal, se convierte en 0179.

La coma decimal se determina por la escala. En el caso de una columna DEC(12,5), por ejemplo, los 5 dígitos más a la derecha están a la derecha de la coma decimal.

El signo lo indica un nybble a la derecha de los nybbles que representa los dígitos. Un signo positivo o negativo se indica de la siguiente manera:

Tabla 76. Valores para el indicador de signo de un número decimal empaquetado

	Representación		
Signo	**Binaria**	**Decimal**	**Hexadecimal**
Positivo (+)	1100	12	C
Negativo (-)	1101	13	D

En resumen:

- Para almacenar cualquier valor, asigne $p/2+1$ bytes, donde p es la precisión.
- Asigne los nybbles de izquierda a derecha para representar el valor. Si un número tiene una precisión par, se añade un nybble de cero inicial. Esta asignación incluye los dígitos cero iniciales (sin significado) y de cola (significativos).
- El nybble de signo será el segundo nybble del último byte.

Por ejemplo:

Columna	Valor	Nybbles en hexadecimal agrupados por bytes
DEC(8,3)	6574.23	00 65 74 23 0C
DEC(6,2)	-334.02	00 33 40 2D
DEC(7,5)	5.2323	05 23 23 0C
DEC(5,2)	-23.5	02 35 0D

Campo SQLLEN para decimales

El campo SQLLEN contiene la precisión (primer byte) y la escala (segundo byte) de la columna decimal. Al escribir una aplicación portátil, los bytes de la precisión y de la escala se deben establecer individualmente, en lugar de establecerlos conjuntamente como un entero corto. Esto evitará los problemas en la inversión de bytes de enteros.

Por ejemplo, en C:

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

Apéndice D. Vistas de catálogo del sistema

El gestor de bases de datos crea y mantiene dos conjuntos de vistas de catálogo del sistema que se definen además de las tablas base de catálogo del sistema.

- Las vistas SYSCAT son vistas de catálogo de sólo lectura que se encuentran en el esquema SYSCAT. La opción RESTRICT en la sentencia CREATE DATABASE determina cómo se otorga el privilegio SELECT. Cuando la opción RESTRICT no está especificada, el privilegio SELECT se otorga a PUBLIC.
- Las vistas SYSSTAT son vistas de catálogo actualizables que se encuentran en el esquema SYSSTAT. Las vistas actualizables contienen información estadística utilizada por el optimizador. Es posible cambiar los valores de algunas de las columnas de estas vistas para comprobar el rendimiento. (Antes de cambiar algún dato estadístico, se recomienda invocar el mandato RUNSTATS para que toda la información estadística refleje el estado actual.)

Las aplicaciones deben grabarse en las vistas SYSCAT y SYSSTAT en lugar de en las tablas base de catálogo.

Todas las vistas de catálogo del sistema se crean durante la creación de la base de datos. Las vistas de catálogo no pueden crearse ni eliminarse explícitamente. En una base de datos Unicode, las vistas de catálogos se crean con una clasificación IDENTITY. En las bases de datos no Unicode, las vistas de catálogos se crean con la clasificación de base de datos. Las vistas se actualizan durante la actividad normal, en respuesta a sentencias de definición de datos SQL, rutinas de entorno y determinados programas de utilidad. Los datos de las vistas de catálogo del sistema están disponibles mediante las funciones de la consulta SQL normal. Las vistas de catálogo del sistema (a excepción de algunas vistas de catálogo que se puedan actualizar) no se pueden modificar utilizando sentencias de manipulación de datos SQL normales.

Una tabla de objeto, columna u objeto de índice sólo aparecerá en la vista de catálogo SYSSTAT actualizable de un usuario si dicho usuario posee el privilegio CONTROL sobre el objeto o posee la autorización DATAACCESS explícita. Un objeto de rutina aparecerá en una vista de catálogo SYSSTAT.ROUTINES actualizable de un usuario si dicho usuario posee la rutina o la autorización SQLADM.

El orden de las columnas en las vistas puede cambiar de un release a otro. Para evitar que esto afecte a la lógica de programación, las columnas deben especificarse en una lista de selección explícitamente y debe evitarse la utilización de `SELECT *`. Las columnas tienen nombres coherentes basados en los tipos de objetos que describen.

Tabla 77. Ejemplos de nombres de columna consistentes para los objetos que describen

Objeto descrito	Nombres de columna
Tabla	TABSCHEMA, TABNAME
Índice	INDSCHEMA, INDNAME
Extensión de índice	IESCHEMA, IENAME
Vista	VIEWSCHEMA, VIEWNAME
Restricción	CONSTSCHEMA, CONSTNAME

Tabla 77. Ejemplos de nombres de columna consistentes para los objetos que describen (continuación)

Objeto descrito	Nombres de columna
Activador	TRIGSCHEMA, TRIGNAME
Paquete	PKGSCHEMA, PKGNAME
Tipo	TYPESCHEMA, TYPENAME, TYPEID
Función	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
Método	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
Procedimiento	ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, ROUTINEID
Columna	COLNAME
Esquema	SCHEMANAME
Espacio de tablas	TBSPACE
Grupo de particiones de base de datos	DBPGNAME
Política de control	AUDITPOLICYNAME, AUDITPOLICYID
Agrupación de almacenam. intermedios	BPNAME
Supervisor de sucesos	EVMONNAME
Condición	CONDSCHEMA, CONDMODULENAME, CONDNAME, CONDMODULEID
Fuente de datos	SERVERNAME, SERVERTYPE, SERVERVERSION
Variable global	VARSCHEMA, VARMODULENAME, VARNAME, VARMODULEID
Plantilla de histograma	TEMPLATENAME, TEMPLATEID
Módulo	MODULESCHEMA, MODULENAME, MODULEID
Rol	ROLENAME, ROLEID
Etiqueta de seguridad	SECLABELNAME, SECLABELID
Política de seguridad	SECPOLICYNAME, SECPOLICYID
Secuencia	SEQSCHEMA, SEQNAME
Umbral	THRESHOLDNAME, THRESHOLDID
Contexto fiable	CONTEXTNAME, CONTEXTID
Acción de trabajo	ACTIONNAME, ACTIONID
Conjunto de acciones de trabajo	ACTIONSETNAME, ACTIONSETID
Clase de trabajo	WORKCLASSNAME, WORKCLASSID

Tabla 77. Ejemplos de nombres de columna consistentes para los objetos que describen (continuación)

Objeto descrito	Nombres de columna
Conjunto de clases de trabajo	WORKCLASSSETNAME, WORKCLASSSETID
Carga de trabajo	WORKLOADID, WORKLOADNAME
Derivador	WRAPNAME
Indicación de fecha y hora de modificación	ALTER_TIME
Indicación de la fecha y hora de creación	CREATE_TIME

Guía básica para las vistas de catálogo

Tabla 78. Guía básica para las vistas de catálogo de sólo lectura

Descripción	Vista de catálogo
atributos de tipos de datos estructurados	"SYSCAT.ATTRIBUTES" en la página 823
políticas de comprobación	"SYSCAT.AUDITPOLICIES" en la página 825
	"SYSCAT.AUDITUSE" en la página 827
autorizaciones en base de datos	"SYSCAT.DBAUTH" en la página 862
configuración de agrupación de almacenamientos intermedios en grupo de particiones de base de datos	"SYSCAT.BUFFERPOOLS" en la página 829
tamaño de agrupación de almacenamientos intermedios en partición de base de datos	"SYSCAT.BUFFERPOOLDBPARTITIONS" en la página 828
funciones de conversión	"SYSCAT.CASTFUNCTIONS" en la página 830
restricciones de comprobación	"SYSCAT.CHECKS" en la página 831
privilegios de columna	"SYSCAT.COLAUTH" en la página 833
columnas	"SYSCAT.COLUMNS" en la página 842
columnas referenciadas por restricciones de comprobación	"SYSCAT.COLCHECKS" en la página 834
columnas utilizadas en dimensiones	"SYSCAT.COLUSE" en la página 848
columnas utilizadas en claves	"SYSCAT.KEYCOLUSE" en la página 901
condiciones	"SYSCAT.CONDITIONS" en la página 849
dependencias de restricción	"SYSCAT.CONSTDEP" en la página 850
particiones de base de datos de grupo de particiones de base de datos	"SYSCAT.DBPARTITIONGROUPDEF" en la página 864
definiciones de grupo de particiones de base de datos	"SYSCAT.DBPARTITIONGROUPS" en la página 865
particiones de datos	"SYSCAT.DATAPARTITIONEXPRESSION" en la página 853
	"SYSCAT.DATAPARTITIONS" en la página 854
dependencias de tipos de datos	"SYSCAT.DATATYPEDEP" en la página 857
tipos de datos	"SYSCAT.DATATYPES" en la página 858
estadísticas detalladas de grupo de columnas	"SYSCAT.COLGROUPCOLS" en la página 836
	"SYSCAT.COLGROUPDIST" en la página 837
	"SYSCAT.COLGROUPDISTCOUNTS" en la página 838
	"SYSCAT.COLGROUPS" en la página 839
opciones de columna detalladas	"SYSCAT.COLOPTIONS" en la página 841
estadísticas detalladas de columna	"SYSCAT.COLDIST" en la página 835
correlaciones de distribución	"SYSCAT.PARTITIONMAPS" en la página 917
definiciones de supervisor de sucesos	"SYSCAT.EVENTMONITORS" en la página 866
sucesos supervisados actualmente	"SYSCAT.EVENTS" en la página 868
	"SYSCAT.EVENTTABLES" en la página 869
campos de tipos de datos de fila	"SYSCAT.ROWFIELDS" en la página 942
dependencias de funciones[1]	"SYSCAT.ROUTINEDEP" en la página 925
correlación de funciones	"SYSCAT.FUNCMAPPINGS" en la página 873
opciones de correlación de funciones	"SYSCAT.FUNCMAPOPTIONS" en la página 871

Tabla 78. Guía básica para las vistas de catálogo de sólo lectura (continuación)

Descripción	Vista de catálogo
opciones de correlación de parámetros de función	"SYSCAT.FUNCMAPPARMOPTIONS" en la página 872
parámetros de funciones[1]	"SYSCAT.ROUTINEPARMS" en la página 929
funciones[1]	"SYSCAT.ROUTINES" en la página 932
variables globales	"SYSCAT.VARIABLEAUTH" en la página 991
	"SYSCAT.VARIABLEDEP" en la página 992
	"SYSCAT.VARIABLES" en la página 994
jerarquías (tipos, tablas, vistas)	"SYSCAT.HIERARCHIES" en la página 874
	"SYSCAT.FULLHIERARCHIES" en la página 870
columnas de identidad	"SYSCAT.COLIDENTATTRIBUTES" en la página 840
columnas de índice	"SYSCAT.INDEXCOLUSE" en la página 879
particiones de datos de índice	"SYSCAT.INDEXPARTITIONS" en la página 896
dependencias de índice	"SYSCAT.INDEXDEP" en la página 880
explotación del índice	"SYSCAT.INDEXEXPLOITRULES" en la página 889
dependencias de extensión de índice	"SYSCAT.INDEXEXTENSIONDEP" en la página 890
parámetros de extensión de índice	"SYSCAT.INDEXEXTENSIONPARMS" en la página 893
métodos de búsqueda de extensiones de índice	"SYSCAT.INDEXEXTENSIONMETHODS" en la página 892
extensiones de índice	"SYSCAT.INDEXEXTENSIONS" en la página 894
opciones de índice	"SYSCAT.INDEXOPTIONS" en la página 895
privilegios de índice	"SYSCAT.INDEXAUTH" en la página 878
índices	"SYSCAT.INDEXES" en la página 882
objetos no válidos	"SYSCAT.INVALIDOBJECTS" en la página 900
dependencias de métodos[1]	"SYSCAT.ROUTINEDEP" en la página 925
parámetros de métodos[1]	"SYSCAT.ROUTINES" en la página 932
métodos[1]	"SYSCAT.ROUTINES" en la página 932
objetos de módulo	"SYSCAT.MODULEOBJECTS" en la página 903
privilegios de módulo	"SYSCAT.MODULEAUTH" en la página 902
módulos	"SYSCAT.MODULES" en la página 904
apodos	"SYSCAT.NICKNAMES" en la página 906
correlación de objeto	"SYSCAT.NAMEMAPPINGS" en la página 905
dependencias de paquete	"SYSCAT.PACKAGEDEP" en la página 910
privilegios de paquete	"SYSCAT.PACKAGEAUTH" en la página 909
paquetes	"SYSCAT.PACKAGES" en la página 912
tablas particionadas	"SYSCAT.TABDETACHEDDEP" en la página 966
privilegios de paso a través	"SYSCAT.PASSTHRUAUTH" en la página 918
especificaciones de predicado	"SYSCAT.PREDICATESPECS" en la página 919
opciones de procedimiento	"SYSCAT.ROUTINEOPTIONS" en la página 927
opciones de parámetro de procedimiento	"SYSCAT.ROUTINEPARMOPTIONS" en la página 928
parámetros de procedimientos[1]	"SYSCAT.ROUTINEPARMS" en la página 929
procedimientos[1]	"SYSCAT.ROUTINES" en la página 932

Tabla 78. Guía básica para las vistas de catálogo de sólo lectura (continuación)

Descripción	Vista de catálogo
tablas protegidas	"SYSCAT.SECURITYLABELACCESS" en la página 945
	"SYSCAT.SECURITYLABELCOMPONENTELEMENTS" en la página 946
	"SYSCAT.SECURITYLABELCOMPONENTS" en la página 947
	"SYSCAT.SECURITYLABELS" en la página 948
	"SYSCAT.SECURITYPOLICIES" en la página 949
	"SYSCAT.SECURITYPOLICYCOMPONENTRULES" en la página 950
	"SYSCAT.SECURITYPOLICYEXEMPTIONS" en la página 951
	"SYSCAT.SURROGATEAUTHIDS" en la página 960
proporciona DB2 para compatibilidad con z/OS	"SYSIBM.SYSDUMMY1" en la página 1020
restricciones de referencia	"SYSCAT.REFERENCES" en la página 920
opciones de tabla remota	"SYSCAT.TABOPTIONS" en la página 976
roles	"SYSCAT.ROLEAUTH" en la página 921
	"SYSCAT.ROLES" en la página 922
dependencias de rutina	"SYSCAT.ROUTINEDEP" en la página 925
parámetros de rutinas[1]	"SYSCAT.ROUTINEPARMS" en la página 929
privilegios de rutina	"SYSCAT.ROUTINEAUTH" en la página 923
rutinas[1]	"SYSCAT.ROUTINES" en la página 932
	"SYSCAT.ROUTINESFEDERATED" en la página 940
privilegios de esquema	"SYSCAT.SCHEMAAUTH" en la página 943
esquemas	"SYSCAT.SCHEMATA" en la página 944
privilegios de secuencia	"SYSCAT.SEQUENCEAUTH" en la página 952
secuencias	"SYSCAT.SEQUENCES" en la página 953
opciones del servidor	"SYSCAT.SERVEROPTIONS" en la página 955
opciones de usuario específicas de servidor	"SYSCAT.USEROPTIONS" en la página 990
sentencias en paquetes	"SYSCAT.STATEMENTS" en la página 959
procedimientos almacenados	"SYSCAT.ROUTINES" en la página 932
servidores del sistema	"SYSCAT.SERVERS" en la página 956
restricciones de tabla	"SYSCAT.TABCONST" en la página 963
dependencias de tabla	"SYSCAT.TABDEP" en la página 964
privilegios de tabla	"SYSCAT.TABAUTH" en la página 961
privilegios de uso de espacios de tablas	"SYSCAT.TBSPACEAUTH" en la página 977
espacios de tablas	"SYSCAT.TABLESPACES" en la página 974
tablas	"SYSCAT.TABLES" en la página 967
transformaciones	"SYSCAT.TRANSFORMS" en la página 981
dependencias de activador	"SYSCAT.TRIGDEP" en la página 982
activadores	"SYSCAT.TRIGGERS" en la página 984

Tabla 78. Guía básica para las vistas de catálogo de sólo lectura (continuación)

Descripción	Vista de catálogo
contextos fiables	"SYSCAT.CONTEXTATTRIBUTES" en la página 851
	"SYSCAT.CONTEXTS" en la página 852
correlación de tipos	"SYSCAT.TYPEMAPPINGS" en la página 986
funciones definidas por el usuario	"SYSCAT.ROUTINES" en la página 932
dependencias de vista	"SYSCAT.TABDEP" en la página 964
vistas	"SYSCAT.TABLES" en la página 967
	"SYSCAT.VIEWS" en la página 996
gestión de carga de trabajo	"SYSCAT.HISTOGRAMTEMPLATEBINS" en la página 875
	"SYSCAT.HISTOGRAMTEMPLATES" en la página 876
	"SYSCAT.HISTOGRAMTEMPLATEUSE" en la página 877
	"SYSCAT.SERVICECLASSES" en la página 957
	"SYSCAT.THRESHOLDS" en la página 978
	"SYSCAT.WORKACTIONS" en la página 997
	"SYSCAT.WORKACTIONSETS" en la página 1000
	"SYSCAT.WORKCLASSES" en la página 1001
	"SYSCAT.WORKCLASSSETS" en la página 1002
	"SYSCAT.WORKLOADAUTH" en la página 1003
	"SYSCAT.WORKLOADCONNATTR" en la página 1004
	"SYSCAT.WORKLOADS" en la página 1005
opciones de derivador	"SYSCAT.WRAPOPTIONS" en la página 1008
derivadores	"SYSCAT.WRAPPERS" en la página 1009
series XML	"SYSCAT.XMLSTRINGS" en la página 1012
índice de valores XML	"SYSCAT.INDEXXMLPATTERNS" en la página 899
Objetos XSR	"SYSCAT.XDBMAPGRAPHS" en la página 1010
	"SYSCAT.XDBMAPSHREDTREES" en la página 1011
	"SYSCAT.XSROBJECTAUTH" en la página 1013
	"SYSCAT.XSROBJECTCOMPONENTS" en la página 1014
	"SYSCAT.XSROBJECTDEP" en la página 1015
	"SYSCAT.XSROBJECTDETAILS" en la página 1017
	"SYSCAT.XSROBJECTHIERARCHIES" en la página 1018
	"SYSCAT.XSROBJECTS" en la página 1019

[1] Todavía están disponibles las vistas de catálogo siguientes para funciones, métodos y procedimientos definidos en DB2 Versión 7.1 y anterior:

```
Funciones:      SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Métodos:        SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Procedimientos: SYSCAT.PROCEDURES, SYSCAT.PROCPARMS
```

No obstante, estas vistas no se han actualizado desde DB2 Versión 7.1. Utilice en su lugar la vista de catálogo SYSCAT.ROUTINES, SYSCAT.ROUTINEDEP o SYSCAT.ROUTINEPARMS.

Guía básica para las vistas de catálogo

Tabla 79. Guía básica para las vistas de catálogo actualizables

Descripción	Vista de catálogo
columnas	"SYSSTAT.COLUMNS" en la página 1025
estadísticas detalladas de grupo de columnas	"SYSSTAT.COLGROUPDIST" en la página 1022
	"SYSSTAT.COLGROUPDISTCOUNTS" en la página 1023
	"SYSSTAT.COLGROUPS" en la página 1024
estadísticas detalladas de columna	"SYSSTAT.COLDIST" en la página 1021
índices	"SYSSTAT.INDEXES" en la página 1027
rutinas[1]	"SYSSTAT.ROUTINES" en la página 1031
tablas	"SYSSTAT.TABLES" en la página 1033

[1] La vista de catálogo SYSSTAT.FUNCTIONS todavía existe para actualizar los datos estadísticos de funciones y métodos. Sin embargo, esta vista no refleja ningún cambio desde DB2 Versión 7.1.

SYSCAT.ATTRIBUTES

Cada fila representa un atributo que está definido para un tipo de datos estructurado definido por el usuario. Incluye atributos heredados de subtipos.

Tabla 80. Vista de catálogo SYSCAT.ATTRIBUTES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos estructurado que incluye el atributo.
TYPEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos estructurado. El valor es nulo si no es un tipo de datos estructurado de módulo.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos estructurado que incluye el atributo.
ATTR_NAME	VARCHAR (128)		Nombre del atributo.
ATTR_TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos de un atributo.
ATTR_TYPEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos de un atributo. El valor es nulo si no es un atributo de módulo.
ATTR_TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos de un atributo.
TARGET_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de fila de destino. Sólo se aplica a los tipos de referencia; valor nulo en caso contrario.
TARGET_TYPEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el tipo de fila de destino. El valor es nulo si no es un tipo de fila de módulo. Sólo se aplica a los tipos de referencia; valor nulo en caso contrario.
TARGET_TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo de fila de destino. Sólo se aplica a los tipos de referencia; valor nulo en caso contrario.
SOURCE_TYPESCHEMA	VARCHAR (128)		Para atributos heredados, el nombre de esquema del tipo de datos con el que se ha definido el atributo en primer lugar. Para atributos no heredados, esta columna es igual que TYPESCHEMA.
SOURCE_TYPEMODULENAME	VARCHAR(128)	S	Para atributos heredados, el nombre no calificado del módulo al que pertenece el tipo de datos con el que se definió el atributo en primer lugar. Para atributos no heredados, esta columna es igual que TYPEMODULEID. El valor es nulo si no es un tipo de datos de módulo.
SOURCE_TYPENAME	VARCHAR (128)		Para atributos heredados, el nombre no calificado del tipo de datos con el que se ha definido el atributo en primer lugar. Para atributos no heredados, esta columna es igual que TYPENAME.

Tabla 80. Vista de catálogo SYSCAT.ATTRIBUTES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ORDINAL	SMALLINT		Posición del atributo en la definición del tipo de datos estructurado, empezando por 0.
LENGTH	INTEGER		Longitud del tipo de datos de atributo. 0 si el atributo es un tipo definido por el usuario.
SCALE	SMALLINT		Escala si el tipo de datos de atributo es DECIMAL o un tipo diferenciado basado en DECIMAL; el número de dígitos de segundos fraccionarios si el tipo de datos de atributo es TIMESTAMP o un tipo diferenciado basado en TIMESTAMP; de lo contrario, 0.
CODEPAGE	SMALLINT		Para tipos de serie, indica la página de códigos; 0 indica FOR BIT DATA; 0 para tipos que no sean de serie.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para el atributo; un valor nulo en caso contrario.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para el atributo; un valor nulo en caso contrario.
LOGGED	CHAR (1)		Sólo se aplica a los tipos LOB; blanco en caso contrario. • N = Los cambios no se anotan cronológicamente • Y = Los cambios se anotan cronológicamente
COMPACT	CHAR (1)		Sólo se aplica a los tipos LOB; blanco en caso contrario. • N = Se almacena en formato no compacto • Y = Se almacena en formato compacto
DL_FEATURES	CHAR(10)		Esta columna ya no se utiliza y se eliminará en un próximo release.
JAVA_FIELDNAME	VARCHAR (256)	S	Reservado para una utilización futura.

SYSCAT.AUDITPOLICIES

Cada fila representa una política de comprobación.

Tabla 81. Vista de catálogo SYSCAT.AUDITPOLICIES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AUDITPOLICYNAME	VARCHAR (128)		Nombre de la política de comprobación.
AUDITPOLICYID	INTEGER		Identificador de la política de comprobación.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la política de comprobación.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez la política de comprobación.
AUDITSTATUS	CHAR (1)		Estado para la categoría AUDIT. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
CONTEXTSTATUS	CHAR (1)		Estado para la categoría CONTEXT. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
VALIDATESTATUS	CHAR (1)		Estado para la categoría VALIDATE. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
CHECKINGSTATUS	CHAR (1)		Estado para la categoría CHECKING. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
SECMAINTSTATUS	CHAR (1)		Estado para la categoría SECMAINT. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
OBJMAINTSTATUS	CHAR (1)		Estado para la categoría OBJMAINT. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio

Tabla 81. Vista de catálogo SYSCAT.AUDITPOLICIES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SYSADMINSTATUS	CHAR (1)		Estado para la categoría SYSADMIN. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
EXECUTESTATUS	CHAR (1)		Estado para la categoría EXECUTE. • B = Ambos • F = Error • N = Ninguno • S = Satisfactorio
EXECUTEWITHDATA	CHAR (1)		Variables del lenguaje principal y marcadores de parámetro conectados con la categoría EXECUTE. • N = No • Y = Sí
ERRORTYPE	CHAR (1)		Tipo de error de comprobación. • A = Comprobación • N = Normal
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.AUDITUSE

Cada fila representa una política de comprobación asociada a un objeto no de base de datos, como por ejemplo, USER, GROUP o autorización (SYSADM, SYSCTRL, SYSMAINT).

Tabla 82. Vista de catálogo SYSCAT.AUDITUSE

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AUDITPOLICYNAME	VARCHAR (128)		Nombre de la política de comprobación.
AUDITPOLICYID	INTEGER		Identificador de la política de comprobación.
OBJECTTYPE	CHAR(1)		El tipo de objeto al que está asociada esta política de comprobación. • S = MQT • T = Tabla • g = Autorización • i = ID de autorización • x = Contexto fiable • Blanco = Base de datos
SUBOBJECTTYPE	CHAR(1)		Si OBJECTTYPE es 'i', este es el tipo que representa el ID de autorización. • G = Grupo • R = Rol • U = Usuario • Blanco = No se aplica
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto para el se está utilizando la política de comprobación. OBJECTSCHEMA es nulo si OBJECTTYPE identifica un objeto para el que no se aplica un esquema.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto para el se está utilizando esta política de comprobación.

SYSCAT.BUFFERPOOLDBPARTITIONS

Cada fila representa una combinación de una agrupación de almacenamientos intermedios y una partición de base de datos, en la que el tamaño de la agrupación de almacenamientos intermedios de dicha partición es distinto del tamaño por omisión correspondiente a otras particiones del mismo grupo de particiones de base de datos (representado en SYSCAT.BUFFERPOOLS).

Tabla 83. Vista de catálogo SYSCAT.BUFFERPOOLDBPARTITIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
BUFFERPOOLID	INTEGER		Identificador interno de la agrupación de almacenamientos intermedios.
DBPARTITIONNUM	SMALLINT		Número de partición de base de datos.
NPAGES	INTEGER		Número de páginas de esta agrupación de almacenamientos intermedios en esta partición de base de datos.

SYSCAT.BUFFERPOOLS

Cada fila representa la configuración de una agrupación de almacenamientos intermedios de un grupo de particiones de base de datos de una base de datos o de todas las particiones de base de datos de una base de datos.

Tabla 84. Vista de catálogo SYSCAT.BUFFERPOOLS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
BPNAME	VARCHAR (128)		Nombre de la agrupación de almacenamientos intermedios.
BUFFERPOOLID	INTEGER		Identificador de la agrupación de almacenamientos intermedios.
DBPGNAME	VARCHAR (128)	S	Nombre de un grupo de particiones de base de datos (el valor es nulo si la agrupación de almacenamientos intermedios existe en todas las particiones de base de datos de la base de datos).
NPAGES	INTEGER		Número de página por omisión de esta agrupación de almacenamientos intermedios de las particiones de base de datos de este grupo de particiones de base de datos.
PAGESIZE	INTEGER		Tamaño de página correspondiente a esta agrupación de almacenamientos intermedios de las particiones de base de datos de este grupo de particiones de base de datos.
ESTORE	INTEGER		Siembre 'N'. El almacenamiento ampliado ya no se aplica.
NUMBLOCKPAGES	INTEGER		Número de páginas de la agrupación de almacenamientos intermedios que deben estar en un área basada en bloque. Sólo los captadores previos que realizan una captación previa secuencial utilizan el área basada en bloque de la agrupación de almacenamientos intermedios.
BLOCKSIZE	INTEGER		Número de páginas de un *bloque*.
NGNAME[1]	VARCHAR (128)	S	Nombre de un grupo de particiones de base de datos (el valor es nulo si la agrupación de almacenamientos intermedios existe en todas las particiones de base de datos de la base de datos).

Nota:

1. La columna NGNAME se incluye por razones de compatibilidad con versiones anteriores. Consulte DBPGNAME.

SYSCAT.CASTFUNCTIONS

Cada fila representa una función de conversión, sin incluir las funciones de conversión incorporadas.

Tabla 85. Vista de catálogo SYSCAT.CASTFUNCTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FROM_TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos del parámetro.
FROM_TYPEMODULENAME	VARCHAR (128)		Nombre no calificado del módulo al que pertenece el tipo de datos del parámetro. El valor es nulo si no es un tipo de datos de módulo.
FROM_TYPENAME	VARCHAR (128)		Nombre del tipo de datos del parámetro.
FROM_TYPEMODULEID	INTEGER	S	Identificador del módulo al que pertenece el tipo de datos del parámetro. El valor es nulo si no es un tipo de datos de módulo.
TO_TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos del resultado después de la conversión.
TO_TYPEMODULENAME	VARCHAR (128)		Nombre no calificado del módulo al que pertenece el tipo de datos del resultado tras la conversión. El valor es nulo si no es un tipo de datos de módulo.
TO_TYPENAME	VARCHAR (128)		Nombre del tipo de datos del resultado después de la conversión.
TO_TYPEMODULEID	INTEGER	S	Identificador del módulo al que pertenece el tipo de datos del resultado tras la conversión. El valor es nulo si no es un tipo de datos de módulo.
FUNCSCHEMA	VARCHAR (128)		Nombre de esquema de la función.
FUNCMODULENAME	VARCHAR (128)		Nombre no calificado del módulo al que pertenece la función. El valor es nulo si no es una función de módulo.
FUNCNAME	VARCHAR (128)		Nombre no calificado de la función.
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
FUNCMODULEID	INTEGER	S	Identificador del módulo al que pertenece la función. El valor es nulo si no es una función de módulo.
ASSIGN_FUNCTION	CHAR (1)		• N = No es una función de asignación • Y = Función de asignación implícita

SYSCAT.CHECKS

Cada fila representa una restricción de comprobación o una columna derivada de una tabla de consultas materializadas. Para jerarquías de tablas, cada restricción de comprobación se registra sólo a nivel de la jerarquía en la que se ha creado la restricción.

Tabla 86. Vista de catálogo SYSCAT.CHECKS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre de la restricción de comprobación.
OWNER	VARCHAR (128)		ID de autorización del propietario de la restricción.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla a la que se aplica esta restricción.
TABNAME	VARCHAR (128)		Nombre de la tabla a la que se aplica esta restricción.
CREATE_TIME	TIMESTAMP		Hora a la que se ha definido la restricción. Se utiliza para resolver funciones que forman parte de esta restricción. Las funciones creadas después de que se definiera la restricción no se eligen.
QUALIFIER	VARCHAR (128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.
TYPE	CHAR (1)		Tipo de restricción de comprobación: • C = Restricción de comprobación • F = Dependencia funcional • O = La restricción es una propiedad del objeto • S = Restricción de comprobación generada por el sistema para la columna GENERATED ALWAYS
FUNC_PATH	CLOB (2K)		Vía de acceso de SQL en vigor en el momento en que se definió la restricción.
TEXT	CLOB(2M)		Texto de la condición de comprobación o definición de la columna derivad.[1]
PERCENTVALID	SMALLINT		Número de filas para las que es válida la restricción informativa, expresado como un porcentaje del total.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para la restricción.
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para la restricción.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en la restricción.
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en la restricción.
DEFINER[2]	VARCHAR (128)		ID de autorización del propietario de la restricción.

Tabla 86. Vista de catálogo SYSCAT.CHECKS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción

Nota:

1. En la vista de catálogo, el texto de la cláusula de comprobación siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. La restricción de comprobación siempre se aplicará en la página de códigos de la tabla destino y no contendrá ningún carácter de sustitución cuando se aplique. (La restricción de comprobación se aplicará en base al texto original de la página de códigos de la tabla destino, que es posible que no incluya los caracteres de sustitución.)
2. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.COLAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado uno o más privilegios sobre una columna.

Tabla 87. Vista de catálogo SYSCAT.COLAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado un privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene un privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o vista en la que se mantiene el privilegio.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o vista en la que se mantiene el privilegio.
COLNAME	VARCHAR (128)		Nombre de la columna a la que se aplica este privilegio.
COLNO	SMALLINT		Número de columna de esta columna dentro de la tabla (empezando por 0).
PRIVTYPE	CHAR (1)		• R = Hace referencia a privilegio • U = Actualiza privilegio
GRANTABLE	CHAR (1)		• G = El privilegio se puede otorgar • N = El privilegio no se puede otorgar

Nota:

1. Los privilegios se pueden otorgar por columna, pero sólo se pueden revocar en base a una tabla.

SYSCAT.COLCHECKS

Cada fila representa una columna a la que hace referencia una restricción de comprobación o la definición de una tabla de consultas materializadas. Para jerarquías de tablas, cada restricción de comprobación se registra sólo a nivel de la jerarquía en la que se ha creado la restricción.

Tabla 88. Vista de catálogo SYSCAT.COLCHECKS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre de la restricción de comprobación.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla que contiene la columna a la que se hace referencia.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla que contiene la columna a la que se hace referencia.
COLNAME	VARCHAR (128)		Nombre de la columna.
USAGE	CHAR (1)		• D = La columna es el hijo de una dependencia funcional • P = La columna es el padre de una dependencia funcional • R = Se hace referencia a la columna en la restricción de comprobación • S = La columna es una fuente de la restricción de comprobación de columna generada por el sistema que da soporte a una tabla de consultas materializadas • T = La columna es un destino de la restricción de comprobación de columna generada por el sistema que da soporte a una tabla de consultas materializadas

SYSCAT.COLDIST

Cada fila representa el valor número *n* más frecuente de alguna columna, o el valor cuantil *n* (distribución acumulada) de la columna. Sólo se aplica a columnas de tablas reales (no a vistas). No se registra ninguna estadística para columnas heredadas de tablas con tipo.

Tabla 89. Vista de catálogo SYSCAT.COLDIST

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla a la que se aplican las estadísticas.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla a la que se aplican las estadísticas.
COLNAME	VARCHAR (128)		Nombre de la columna a la que se aplican las estadísticas.
TYPE	CHAR (1)		• F = Valor de frecuencia • Q = Valor cuantil
SEQNO	SMALLINT		Si TYPE = 'F', *n* en esta columna identifica el valor número *n* más frecuente. Si TYPE = 'Q', *n* en esta columna identifica el valor cuantil número *n*.
COLVALUE[1]	VARCHAR (254)	S	El valor de los datos como un literal de caracteres o un valor nulo.
VALCOUNT	BIGINT		Si TYPE = 'F', VALCOUNT es el número de apariciones de COLVALUE en la columna. Si TYPE = 'Q', VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE.
DISTCOUNT[2]	BIGINT	S	Si TYPE = 'Q', esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (valor nulo si no está disponible).

Nota:

1. En la vista de catálogo, el valor de COLVALUE siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.
2. DISTCOUNT sólo se recopila para las columnas que son la primera columna de clave de un índice.

SYSCAT.COLGROUPCOLS

Cada fila representa una columna que forma parte de un grupo de columnas.

Tabla 90. Vista de catálogo SYSCAT.COLGROUPCOLS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLGROUPID	INTEGER		Identificador del grupo de columnas.
COLNAME	VARCHAR (128)		Nombre de la columna del grupo de columnas.
TABSCHEMA	VARCHAR (128)		Número de esquema de la tabla correspondiente a la columna del grupo de columnas.
TABNAME	VARCHAR (128)		Número no calificado de la tabla correspondiente a la columna del grupo de columnas.
ORDINAL	SMALLINT		Número ordinal de la columna del grupo de columnas.

SYSCAT.COLGROUPDIST

Cada fila representa el valor de la columna del grupo de columnas que forma el valor más frecuente número *n* del grupo de columnas o el valor cuantil *n* del grupo de columnas.

Tabla 91. Vista de catálogo SYSCAT.COLGROUPDIST

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLGROUPID	INTEGER		Identificador del grupo de columnas.
TYPE	CHAR (1)		• F = Valor de frecuencia • Q = Valor cuantil
ORDINAL	SMALLINT		Número ordinal de la columna del grupo de columnas.
SEQNO	SMALLINT		Si TYPE = 'F', *n* en esta columna identifica el valor número *n* más frecuente. Si TYPE = 'Q', *n* en esta columna identifica el valor cuantil número *n*.
COLVALUE[1]	VARCHAR (254)		El valor de los datos como un literal de caracteres o un valor nulo.

Nota:

1. En la vista de catálogo, el valor de COLVALUE siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

SYSCAT.COLGROUPDISTCOUNTS

Cada fila representa las estadísticas de distribución que se aplican al valor más frecuente número *n* de un grupo de columnas o el valor cuantil *n* de un grupo de columnas.

Tabla 92. Vista de catálogo SYSCAT.COLGROUPDISTCOUNTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLGROUPID	INTEGER		Identificador del grupo de columnas.
TYPE	CHAR (1)		• F = Valor de frecuencia • Q = Valor cuantil
SEQNO	SMALLINT		El número de secuencia *n* que representa el valor *n* de TYPE.
VALCOUNT	BIGINT		Si TYPE = 'F', VALCOUNT es el número de apariciones de COLVALUE para el grupo de columnas con este SEQNO. Si TYPE = 'Q', VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE para el grupo de columnas con este SEQNO.
DISTCOUNT	BIGINT		Si TYPE = 'Q', esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE para el grupo de columnas con este SEQNO (valor nulo si no está disponible).

SYSCAT.COLGROUPS

Cada fila representa un grupo de columnas y estadísticas que se aplican a todo el grupo de columnas.

Tabla 93. Vista de catálogo SYSCAT.COLGROUPS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLGROUPSCHEMA	VARCHAR (128)		Nombre de esquema del grupo de columnas.
COLGROUPNAME	VARCHAR (128)		Nombre no calificado del grupo de columnas.
COLGROUPID	INTEGER		Identificador del grupo de columnas.
COLGROUPCARD	BIGINT		Cardinalidad del grupo de columnas.
NUMFREQ_VALUES	SMALLINT		Número de valores frecuentes recopilados para el grupo de columnas.
NUMQUANTILES	SMALLINT		Número de valores cuantiles recopilados para el grupo de columnas.

SYSCAT.COLIDENTATTRIBUTES

Cada fila representa una columna de identidad definida para una tabla.

Tabla 94. Vista de catálogo SYSCAT.COLIDENTATTRIBUTES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o vista que contiene la columna.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o vista que contiene la columna.
COLNAME	VARCHAR (128)		Nombre de la columna.
START	DECIMAL(31,0)	S	Valor inicial de la secuencia. El valor es nulo si la secuencia es un alias.
INCREMENT	DECIMAL(31,0)	S	Valor de incremento. El valor es nulo si la secuencia es un alias.
MINVALUE	DECIMAL(31,0)	S	Valor mínimo de la secuencia.El valor es nulo si la secuencia es un alias.
MAXVALUE	DECIMAL(31,0)	S	Valor máximo de la secuencia. El valor es nulo si la secuencia es un alias.
CYCLE	CHAR (1)		Indica si la secuencia puede o no continuar generando valores después de alcanzar su valor máximo o mínimo. • N = La secuencia no puede continuar • Y = La secuencia puede continuar • Blanco = La secuencia es un alias.
CACHE	INTEGER		Número de valores de secuencia que se deben preasignar en memoria para el acceso más rápido. 0 indica que los valores de la secuencia no se deben preasignar. En una base de datos particionada, este valor se aplica a cada partición de base de datos. -1 si la secuencia es un alias.
ORDER	CHAR (1)		Indica si los números de secuencia se deben o no generar en orden de petición. • N = No es necesario que los números de secuencia se generen en orden de petición • Y = Los números de secuencia se deben generar en orden de petición • Blanco = La secuencia es un alias.
NEXTCACHEFIRSTVALUE	DECIMAL(31,0)	S	Primer valor que estará disponible para asignarlo en el bloque de memoria intermedia siguiente. Si no hay almacenamiento en antememoria, el siguiente valor que estará disponible para asignar.
SEQID	INTEGER		Identificador de la secuencia o el alias.

SYSCAT.COLOPTIONS

Cada fila contiene valores de opciones específicos de columna.

Tabla 95. Vista de catálogo SYSCAT.COLOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema del apodo.
TABNAME	VARCHAR (128)		Apodo de la columna para la que se establecen opciones.
COLNAME	VARCHAR (128)		Nombre de columna local.
OPTION	VARCHAR (128)		Nombre de la opción de columna.
SETTING	CLOB (32K)		Valor.

SYSCAT.COLUMNS

Cada fila representa una columna definida para una tabla, vista o apodo.

Tabla 96. Vista de catálogo SYSCAT.COLUMNS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla, vista o apodo que contiene la columna.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla, vista o apodo que contiene la columna.
COLNAME	VARCHAR (128)		Nombre de la columna.
COLNO	SMALLINT		Número de esta columna en la tabla (empezando por 0).
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos de la columna.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos de la columna.
LENGTH	INTEGER		Longitud máxima de los datos; 0 para tipos diferenciados. La columna LENGTH indica la precisión para campos DECIMAL e indica el número de bytes de almacenamiento necesarios para las columnas de coma flotante decimal; es decir, 8 y 16 para DECFLOAT(16) y DECFLOAT(34), respectivamente.
SCALE	SMALLINT		Escala si el tipo de columna es DECIMAL o un número de dígitos de segundos fraccionarios si el tipo de columna es TIMESTAMP; 0 en caso contrario.
DEFAULT[1]	VARCHAR (254)	S	El valor por omisión de la columna de una tabla expresado como una constante, un registro especial o una función de conversión adecuada para el tipo de datos de la columna. También puede ser la palabra clave NULL. Los valores se pueden convertir a partir de lo que se ha especificado como valor por omisión. Por ejemplo, las constantes de fecha y hora se muestran en formato ISO, los nombres de funciones de conversión se califican con nombres de esquema y los identificadores se delimitan. El valor nulo si no se ha especificado una cláusula DEFAULT o la columna es una columna de vista.

Tabla 96. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
NULLS[2]	CHAR (1)		Atributo de capacidad de nulos para la columna. • N = La columna no puede contener nulos • Y = La columna puede contener nulos El valor puede ser 'N' para una columna de vista derivada de una expresión o función. No obstante, estas columnas permiten valores nulos cuando la sentencia que utiliza la vista se procesa con avisos para errores aritméticos.
CODEPAGE	SMALLINT		Página de códigos utilizada para los datos de esta columna; 0 si la columna está definida como FOR BIT DATA o no es de tipo serie.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para la columna; en caso contrario, valor nulo.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para la columna; de lo contrario, valor nulo.
LOGGED	CHAR (1)		Sólo se aplica a las columnas cuyo tipo es LOB o diferenciado en base a LOB; blanco en caso contrario. • N = La columna no se anota cronológicamente • Y = La columna se anota cronológicamente
COMPACT	CHAR (1)		Sólo se aplica a las columnas cuyo tipo es LOB o diferenciado en base a LOB; blanco en caso contrario. • N = La columna no se compacta • Y = La columna se compacta en el almacenamiento
COLCARD	BIGINT		Número de valores diferenciados de la columna; -1 se no se recopilan estadísticas; -2 para columnas heredadas y columnas de tablas de jerarquía.
HIGH2KEY[3]	VARCHAR (254)	S	El segundo valor de datos más alto. Representación de datos numéricos modificados por literales de caracteres. Vacío si no se recopilan estadísticas. Vacío para columnas heredadas y columnas de tablas jerárquicas.
LOW2KEY[3]	VARCHAR (254)	S	El segundo valor de datos más bajo. Representación de datos numéricos modificados por literales de caracteres. Vacío si no se recopilan estadísticas. Vacío para columnas heredadas y columnas de tablas jerárquicas.

Tabla 96. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AVGCOLLEN	INTEGER		Espacio promedio en bytes si la columna se guarda en la memoria de la base de datos o en una tabla temporal. En los tipos de datos LOB que no están en línea, los tipos de datos LONG y los documentos XML, el valor utilizado para calcular el promedio de longitud de columna es la longitud del descriptor de datos. Se requiere un byte adicional si la columna es anulable; -1 si no se han recopilado estadísticas; -2 para las columnas heredadas y las columnas de las tablas de jerarquía. Nota: el espacio promedio necesario para almacenar la columna en el disco puede ser diferente del valor representado por esta estadística.
KEYSEQ	SMALLINT	S	La posición numérica de la columna dentro de la clave primaria de la tabla. El valor es nulo para columnas de subtablas y tablas jerárquicas.
PARTKEYSEQ	SMALLINT	S	La posición numérica de la columna dentro de la clave de distribución de la tabla; 0 o el valor nulo si la columna no forma parte de la clave de distribución. El valor es nulo para columnas de subtablas y tablas jerárquicas.
NQUANTILES	SMALLINT		Número de valores cuantiles registrados en SYSCAT.COLDIST para esta columna; -1 si no se recopilan estadísticas; -2 para columnas heredadas y columnas de tablas jerárquicas.
NMOSTFREQ	SMALLINT		Número de valores más frecuentes registrados en SYSCAT.COLDIST para esta columna; -1 si no se recopilan estadísticas; -2 para columnas heredadas y columnas de tablas jerárquicas.
NUMNULLS	BIGINT		Número de valores nulos de la columna; -1 si no se recopilan estadísticas.
TARGET_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de fila de destino, si el tipo de esta columna es REFERENCE; valor nulo en caso contrario.
TARGET_TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo de fila de destino, si el tipo de esta columna es REFERENCE; valor nulo en caso contrario.
SCOPE_TABSCHEMA	VARCHAR (128)	S	Nombre de esquema del ámbito (valor de destino), si el tipo de esta columna es REFERENCE; valor nulo en caso contrario.
SCOPE_TABNAME	VARCHAR (128)	S	Nombre no calificado del ámbito (tabla de destino), si el tipo de esta columna es REFERENCE; valor nulo en caso contrario.

Tabla 96. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SOURCE_TABSCHEMA	VARCHAR (128)	S	Para columnas de tablas o vistas con tipo, el nombre de esquema de la tabla o vista en la que se ha incorporado la columna por primera vez. Para columnas no heredadas, es igual que TABSCHEMA. El valor es nulo para columnas de tablas y vistas sin tipo.
SOURCE_TABNAME	VARCHAR (128)	S	Para columnas de tablas y vistas con tipo, el nombre no calificado de la tabla o vista en la que se ha incorporado la columna por primera vez. Para columnas no heredadas, es igual que TABNAME. El valor es nulo para columnas de tablas y vistas sin tipo.
DL_FEATURES	CHAR(10)	S	Esta columna ya no se utiliza y se eliminará en un próximo release.
SPECIAL_PROPS	CHAR(8)	S	Sólo se aplica a columnas de tipo REFERENCE; blancos en caso contrario. Cada posición de byte se define de la manera siguiente: • 1 = Columna identificador de objeto (OID) ('Y' para sí; 'N' para no) • 2 = Generado por el usuario o generado por el sistema ('U' para usuario; 'S' para sistema) Los bytes del 3 al 8 están reservados para su utilización en el futuro.
HIDDEN	CHAR (1)		Tipo de columna oculta. • I = La columna se define como IMPLICITLY HIDDEN • S = Columna oculta gestionada por el sistema • Blanco = La columna no está oculta
INLINE_LENGTH	INTEGER		Tamaño máximo en bytes de la representación interna de una instancia de un documento XML, un tipo estructurado o un tipo de datos LOB, que puede almacenarse en la tabla base; 0 cuando no es de aplicación.
PCTINLINED	SMALLINT		Porcentaje de datos LOB o documentos XML en línea. -1 si no se han recopilado estadísticas.
IDENTITY	CHAR (1)		• N = No es una columna de identidad • T = Columna de indicación de fecha y hora de cambio de fila • Y = Columna de identidad
ROWCHANGETIMESTAMP	CHAR (1)		• N = No es una columna de indicación de fecha y hora de cambio de fila • Y = Columna de indicación de fecha y hora de cambio de fila

Tabla 96. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GENERATED	CHAR (1)		Tipo de columna generada. • A = Siembre se genera un valor de columna • D = Por omisión se genera un valor de columna • Blanco = No se genera ninguna columna
TEXT	CLOB(2M)	S	Para columnas definidas como generadas como expresión, este campo contiene el texto de la expresión de columna generada, comenzando por la palabra clave AS.
COMPRESS	CHAR (1)		• O = Compresión desactivada • S = Comprimir valores por omisión del sistema
AVGDISTINCTPERPAGE	DOUBLE	S	Para su utilización en el futuro.
PAGEVARIANCERATIO	DOUBLE	S	Para su utilización en el futuro.
SUB_COUNT	SMALLINT		Número medio de subelementos de la columna. Sólo se aplica a columnas de tipo serie de caracteres.
SUB_DELIM_LENGTH	SMALLINT		Longitud media de los delimitadores que separan cada subelemento de la columna. Sólo se aplica a columnas de tipo serie de caracteres.
AVGCOLLENCHAR	INTEGER		Promedio del número de caracteres (basándose en la clasificación en vigor para la columna) necesarios para la columna; -1 si el tipo de datos de la columna es long, LOB o XML o si no se han recopilado estadísticas; -2 par columnas heredadas y columnas de tablas jerárquicas.
IMPLICITVALUE[4]	VARCHAR (254)	S	Para una columna que se ha añadido a una tabla después de que se creara la tabla, almacena el valor por omisión en el momento en que se añadió la columna. Para una columna definida cuando se creó la tabla, almacena un valor nulo.
SECLABELNAME	VARCHAR(128)	S	Nombre de la etiqueta de seguridad asociada a la columna si se trata de una columna protegida; en caso contrario, valor nulo.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Tabla 96. Vista de catálogo SYSCAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción

Nota:

1. Para la Versión 2.1.0, los nombres de función de conversión no se han delimitado y pueden seguir apareciendo de esta manera en la columna DEFAULT. También, algunas columnas de vistas incluyen valores por omisión que todavía aparecen en la columna DEFAULT.
2. Empezando en la Versión 2, el valor D (que indica ningún nulo con valor por omisión) ya no se utiliza. En su lugar, la utilización de WITH DEFAULT se indica por un valor no nulo en la columna DEFAULT.
3. En la vista de catálogo, los valores de HIGH2KEY y de LOW2KEY siempre se muestran en la página de códigos de la base de datos y pueden contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.
4. Se permite enlazar una partición de datos a no ser que IMPLICITVALUE para una columna específica sea un valor no nulo tanto para la columna fuente como para la columna de destino, y los valores no coincidan. En este caso, debe eliminar la tabla fuente y volverla a crear. Una columna puede tener un valor no nulo en el campo IMPLICITVALUE si se cumplen una de las siguientes condiciones:
   - La columna se crea como resultado de una sentencia ALTER TABLE...ADD COLUMN.
   - El valor IMPLICITVALUE se propaga desde una fuente durante la función de enlazar
   - El valor IMPLICITVALUE se hereda de la tabla fuente durante el desenlace
   - El campo IMPLICITVALUE se establece durante la actualización de la base de datos de la Versión 8 a la Versión 9, donde se determina que es una columna añadida o que puede ser una columna añadida. Si la base de datos no sabe con certeza si la columna es o no añadida, se trata como añadida. Una columna añadida es una columna creada como resultado de una sentencia ALTER TABLE...ADD COLUMN.

   Para evitar estas incoherencias durante escenarios que no sean de migración, se recomienda crear siempre las tablas que se van a enlazar a todas las columnas ya definidas. Es decir, se recomienda no utilizar nunca la sentencia ALTER TABLE para añadir columnas a una tabla antes de enlazarla.

SYSCAT.COLUSE

Cada fila representa una columna a la que se hace referencia en la cláusula DIMENSIONS de una sentencia CREATE TABLE.

Tabla 97. Vista de catálogo SYSCAT.COLUSE

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla que contiene la columna.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla que contiene la columna.
COLNAME	VARCHAR (128)		Nombre de la columna.
DIMENSION	SMALLINT		Número de dimensiones, basado en el orden de las dimensiones especificadas en la cláusula DIMENSIONS (la posición inicial es 0). Para una dimensión compuesta, este valor será el mismo para cada componente de la dimensión.
COLSEQ	SMALLINT		Posición numérica de la columna en la dimensión a la que pertenece (la posición inicial es 0). El valor es 0 para una sola columna en una dimensión no compuesta.
TYPE	CHAR (1)		Tipo de dimensión. • C = Clúster o clúster de varias dimensiones • P = Particionamiento

SYSCAT.CONDITIONS

Cada fila representa una condición definida en un paquete.

Tabla 98. Vista de catálogo SYSCAT.CONDITIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONDSCHEMA	VARCHAR (128)		Nombre de esquema de la condición.
CONDMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece la condición.
CONDNAME	VARCHAR (128)		Nombre no calificado de la condición.
CONDID	INTEGER		Identificador para la condición.
CONDMODULEID	INTEGER	S	Identificador del módulo al que pertenece la condición.
SQLSTATE	CHAR(5)	S	Valor SQLSTATE asociado con la condición.
OWNER	VARCHAR (128)		ID de autorización del propietario de la condición.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la condición.
REMARKS	VARCHAR (254)		Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.CONSTDEP

Cada fila representa una dependencia de una restricción sobre algún otro objeto. La restricción depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta a la restricción.

Tabla 99. Vista de catálogo SYSCAT.CONSTDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre no calificado de la restricción.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla a la que se aplica la restricción.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla a la que se aplica la restricción.
BTYPE	CHAR (1)		Tipo de objeto del que depende la restricción. Los valores posibles son: • F = Rutina • I = Índice • R = Tipo estructurado definido por el usuario • u = Alias de módulo
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto del que depende la restricción.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto del que depende la restricción.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto del que depende la restricción.

SYSCAT.CONTEXTATTRIBUTES

Cada fila representa un atributo de contexto fiable.

Tabla 100. Vista de catálogo SYSCAT.CONTEXTATTRIBUTES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONTEXTNAME	VARCHAR (128)		Nombre del contexto fiable.
ATTR_NAME	VARCHAR (128)		Nombre del atributo. Uno de los siguientes: • ADDRESS • ENCRYPTION
ATTR_VALUE	VARCHAR (128)		Valor del atributo.
ATTR_OPTIONS	VARCHAR (128)	S	Si ATTR_NAME es 'ADDRESS', especifica el nivel de cifrado necesario para esta dirección específica. Un valor nulo indica que se aplica el atributo ENCRYPTION global.

SYSCAT.CONTEXTS

Cada fila representa un contexto fiable.

Tabla 101. Vista de catálogo SYSCAT.CONTEXTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONTEXTNAME	VARCHAR (128)		Nombre del contexto fiable.
CONTEXTID	INTEGER		Identificador del contexto fiable.
SYSTEMAUTHID	VARCHAR (128)		ID de autorización del sistema asociado al contexto fiable.
DEFAULTCONTEXTROLE	VARCHAR (128)	S	El rol por omisión para el contexto.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el contexto fiable.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez el contexto fiable.
ENABLED	CHAR (1)		Estado de contexto fiable. • N = Inhabilitado • Y = Habilitada
AUDITPOLICYID	INTEGER	S	Identificador de la política de comprobación.
AUDITPOLICYNAME	VARCHAR (128)	S	Nombre de la política de comprobación.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.DATAPARTITIONEXPRESSION

Cada fila representa una expresión correspondiente a esta parte de la clave de particionamiento de tabla.

Tabla 102. Vista de catálogo SYSCAT.DATAPARTITIONEXPRESSION

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla particionada.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla particionada.
DATAPARTITIONKEYSEQ	INTEGER		ID de secuencia de la parte de clave de expresión, empezando por 1.
DATAPARTITIONEXPRESSION	CLOB (32K)		Expresión correspondiente a esta entrada en la secuencia, en sintaxis de SQL.
NULLSFIRST	CHAR (1)		• N = Los valores nulos de esta expresión se comparan altos • Y = Los valores nulos de esta expresión se comparan bajos

SYSCAT.DATAPARTITIONS

Cada fila representa una partición de datos. Nota:

- Las estadísticas de partición de datos representan a una partición de base de datos si la tabla se crea en varias particiones de base de datos.

Tabla 103. Vista de catálogo SYSCAT.DATAPARTITIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DATAPARTITIONNAME	VARCHAR (128)		Nombre de la partición de datos.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla a la que pertenece esta partición de datos.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla a la que pertenece esta partición de datos.
DATAPARTITIONID	INTEGER		Identificador de la partición de datos.
TBSPACEID	INTEGER	S	Identificador del espacio de tablas en el que está almacenada esta partición de datos. El valor es nulo cuando STATUS es 'I'.
PARTITIONOBJECTID	INTEGER	S	Identificador de la partición de datos dentro del espacio de tablas.
LONG_TBSPACEID	INTEGER	S	Identificador del espacio de tablas en el que se almacenan los datos largos. El valor es nulo cuando STATUS es 'I'.
ACCESS_MODE	CHAR (1)		Estado de restricción de acceso de la partición de datos. Estos estados sólo se aplican a objetos que están en estado pendiente de establecer integridad o a objetos procesados por una sentencia SET INTEGRITY. Los valores posibles son: • D = No hay movimiento de datos • F = Acceso completo • N = No hay acceso • R = Acceso de sólo lectura

Tabla 103. Vista de catálogo SYSCAT.DATAPARTITIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
STATUS	VARCHAR (32)		• A = La partición de datos se acaba de enlazar • D = La partición de datos se ha desenlazado y los elementos dependientes desenlazados deben mantenerse de forma incremental con respecto del contenido de esta partición • I = Partición de datos desenlazada cuya entrada en el catálogo se mantiene únicamente durante la limpieza asíncrona del índice; las filas con el valor de STATUS 'I' se eliminan cuando todos los registros de índice que hacen referencia a la partición desenlazada se han suprimido • L = La partición de datos se ha desenlazado lógicamente • Serie vacía = La partición de datos está visible (estado normal) Los bytes comprendidos entre el 2 y el 32 están reservados para su utilización en el futuro.
SEQNO	INTEGER		Número de secuencia de la partición de datos (empezando por 0).
LOWINCLUSIVE	CHAR (1)		• N = No incluye el valor de clave bajo • Y = Incluye el valor de clave bajo
LOWVALUE	VARCHAR (512)		Valor de clave bajo (una representación de serie de un valor de SQL) correspondiente a esta partición de datos.
HIGHINCLUSIVE	CHAR (1)		• N = No incluye el valor de clave alto • Y = Incluye el valor de clave alto
HIGHVALUE	VARCHAR (512)		Valor de clave alto (una representación de serie de un valor de SQL) correspondiente a esta partición de datos.
CARD	BIGINT		Número total de filas de la partición de datos; -1 si no se recopilan estadísticas.
OVERFLOW	BIGINT		Número total de registros de desbordamiento de la partición de datos; -1 si no se recopilan estadísticas.
NPAGES	BIGINT		Número total de páginas en las que hay filas de la partición de datos; -1 si no se recopilan estadísticas.
FPAGES	BIGINT		Número total de páginas de la partición de datos; -1 si no se recopilan estadísticas.
ACTIVE_BLOCKS	BIGINT		Número total de bloques activos en la partición de datos, o -1. Sólo se aplica a las tablas que están en un clúster de varias dimensiones (MDC).

Tabla 103. Vista de catálogo SYSCAT.DATAPARTITIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDEX_TBSPACEID	INTEGER		Identificador para el espacio de tablas que contiene todos los índices particionados de esta partición de datos.
AVGROWSIZE	SMALLINT		Longitud media (en bytes) de filas comprimidas y no comprimidas de esta partición de datos; -1 si no se recopilan estadísticas.
PCTROWSCOMPRESSED	REAL		Filas comprimidas como porcentaje del número total de filas de la partición de datos; -1 si no se recopilan estadísticas.
PCTPAGESAVED	SMALLINT		Porcentaje aproximado de páginas guardadas en la partición de datos como resultado de la compresión de filas. Este valor incluye bytes de actividad general para cada fila de datos de usuario de la partición de datos, pero no incluye el espacio que consume la actividad general del diccionario; es -1 si no se recopilan estadísticas.
AVGCOMPRESSEDROWSIZE	SMALLINT		Longitud media (en bytes) de filas comprimidas en esta partición de datos; -1 si no se recopilan estadísticas.
AVGROWCOMPRESSIONRATIO	REAL		Para filas comprimidas en la partición de datos, es la proporción media de compresión por fila; es decir, la longitud media de las filas no comprimidas dividida entre la longitud media de las filas comprimidas; -1 si no se recopilan estadísticas.
STATS_TIME	TIMESTAMP	S	Hora a la que se ha hecho por última vez un cambio en las estadísticas registradas correspondientes a este objeto. Nulo si no se recopilan estadísticas.
LASTUSED	DATE		Fecha de última utilización de la partición de datos por parte de una sentencia DML o mandato LOAD. El valor por omisión es '0001-01-01'. Este valor se actualiza de forma asíncrona.

SYSCAT.DATATYPEDEP

Cada fila representa una dependencia de un tipo de datos definidos por el usuario sobre algún otro objeto.

Tabla 104. Vista de catálogo SYSCAT.DATATYPEDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos.
TYPEMODULENAME	VARCHAR (128)	S	Nombre de módulo del tipo de datos.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos.
TYPEMODULEID	INTEGER	S	Identificador para el módulo del tipo de datos.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • G = Tabla temporal global • H = Tabla de jerarquía • N = Apodo • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR (128)	S	Nombre del objeto sobre el que hay una dependencia.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.
TABAUTH	SMALLINT	S	Si BTYPE = 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita el tipo de datos dependientes; en caso contrario, valor nulo.

SYSCAT.DATATYPES

Cada fila representa un tipo de datos incorporado o definido por el usuario.

Tabla 105. Vista de catálogo SYSCAT.DATATYPES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos si TYPEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece el tipo de datos.
TYPEMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el tipo definido por el usuario. El valor es nulo si no es un tipo de módulo definido por el usuario.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos.
OWNER	VARCHAR (128)		ID de autorización del propietario del tipo.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
SOURCESCHEMA	VARCHAR (128)	S	Para los tipos diferenciados o los tipos de matriz, nombre de esquema del tipo de datos fuente. Para los tipos estructurados definidos por el usuario, nombre de esquema del tipo incorporado del tipo de representación de referencia. Valor nulo para el resto de tipos de datos.
SOURCEMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos fuente. El valor es nulo si no es un tipo de datos de fuente de módulo.
SOURCENAME	VARCHAR (128)	S	Para los tipos diferenciados o los tipos de matriz, nombre no calificado del tipo de datos fuente. Para los tipos estructurados definidos por el usuario, nombre de tipo incorporado no calificado del tipo de representación de referencia. Valor nulo para el resto de tipos de datos.
METATYPE	CHAR (1)		• A = Tipo de matriz definido por el usuario • C = Tipo de cursor definido por el usuario • F = Tipo de fila definido por el usuario • L = Tipo de matriz asociativa definido por el usuario • R = Tipo estructurado definido por el usuario • S = Tipo predefinido por el sistema • T = Tipo diferenciado definido por el usuario
TYPEID	SMALLINT		Identificador para el tipo de datos

Tabla 105. Vista de catálogo SYSCAT.DATATYPES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPEMODULEID	INTEGER	S	Identificador para el módulo al que pertenece el tipo definido por el usuario. El valor es nulo si no es un tipo de módulo definido por el usuario.
SOURCETYPEID	SMALLINT	S	Identificador para el tipo de fuente (el valor es nulo para tipos incorporados). Para los tipos estructurados definidos por el usuario, es el identificador del tipo de representación de referencia.
SOURCEMODULEID	INTEGER	S	Identificador para el módulo al que pertenece el tipo de datos de fuente. El valor es nulo si no es un tipo de datos de fuente de módulo.
PUBLISHED	CHAR (1)		Indica si se puede hacer referencia al tipo de módulo definido por el usuario desde fuera de su módulo. • N = El tipo de módulo definido por el usuario no está publicado • Y = El tipo de módulo definido por el usuario está publicado • Blanco = No se aplica
LENGTH	INTEGER		Longitud máxima del tipo. 0 para tipos con parámetros incorporados (por ejemplo, DECIMAL y VARCHAR). Para tipos estructurados definidos por el usuario, es la longitud del tipo de representación de referencia.
SCALE	SMALLINT		Escala para los tipos diferenciados o tipos de representación de referencia basados en el tipo DECIMAL incorporado; el número de dígitos de segundos fraccionarios para los tipos diferenciados basados en el tipo TIMESTAMP incorporado; 6 para el tipo TIMESTAMP incorporado; o 0 para los demás tipos (incluido el propio DECIMAL).
CODEPAGE	SMALLINT		Página de códigos de base de datos para tipos de serie, tipos diferenciados basados en tipos de serie o tipos de representación de referencia; 0 en caso contrario.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, nombre de esquema de la clasificación para el tipo de datos; valor nulo en caso contrario.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, nombre no calificado de la clasificación para el tipo de datos; valor nulo en caso contrario.
ARRAY_LENGTH	INTEGER	S	Cardinalidad máxima de la matriz. El valor es nulo si METATYPE no es 'A'.
ARRAYINDEXTYPESCHEMA	VARCHAR(128)	S	Esquema del tipo de datos del índice de matriz. El valor es nulo si METATYPE no es 'L'.

Tabla 105. Vista de catálogo SYSCAT.DATATYPES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ARRAYINDEXTYPENAME	VARCHAR (128)	S	Nombre del tipo de datos del índice de matriz. El valor es nulo si METATYPE no es 'L'.
ARRAYINDEXTYPEID	SMALLINT	S	Identificador para el tipo de índice de matriz. El valor es nulo si METATYPE no es 'L'.
ARRAYINDEXTYPELENGTH	INTEGER	S	Longitud máxima del tipo de datos de índice de matriz. El valor es nulo si METATYPE no es 'L'.
CREATE_TIME	TIMESTAMP		Tiempo de creación del tipo de datos.
VALID	CHAR (1)		• N = El tipo de datos no es válido • Y = El tipo de datos es válido
ATTRCOUNT	SMALLINT		Número de atributos del tipo de datos.
INSTANTIABLE	CHAR (1)		• N = No se pueden crear instancias del tipo • Y = Se pueden crear instancias del tipo
WITH_FUNC_ACCESS	CHAR (1)		• N = Los métodos correspondientes a este tipo no se pueden invocar utilizando notación de función. • Y = Todos los métodos correspondientes a este tipo se pueden invocar utilizando notación de función.
FINAL	CHAR (1)		• N = El tipo definido por el usuario puede tener subtipos. • Y = El tipo definido por el usuario no puede tener subtipos.
INLINE_LENGTH	INTEGER		Longitud máxima de un tipo estructurado que se puede conservar con una fila de tabla base; 0 en caso contrario.
NATURAL_ INLINE_LENGTH	INTEGER	S	Longitud en línea natural generada por el sistema de una instancia de un tipo estructurado. El valor es nulo si este tipo no es un tipo estructurado.
JARSCHEMA	VARCHAR (128)	S	Nombre de esquema del JAR_ID que identifica el archivo Jar que contiene la clase Java que implementa el tipo de SQL. El valor es nulo si no se especifica la cláusula EXTERNAL NAME.
JAR_ID	VARCHAR (128)	S	Identificador del archivo Jar que contiene la clase Java que implementa el tipo de SQL. El valor es nulo si no se especifica la cláusula EXTERNAL NAME.
CLASS	VARCHAR (384)	S	Clase Java que implementa el tipo de SQL. El valor es nulo si no se especifica la cláusula EXTERNAL NAME.

Tabla 105. Vista de catálogo SYSCAT.DATATYPES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SQLJ_REPRESENTATION	CHAR (1)	S	SQLJ "espec_representación" de la clase Java que implementa el tipo de SQL. El valor es nulo si no se especifica la cláusula EXTERNAL NAME ... LANGUAGE JAVA REPRESENTATION SPEC. • D = datos SQL • S = Serializable
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez el tipo de datos.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario del tipo.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.DBAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado una o más autorizaciones del nivel de la base de datos.

Tabla 106. Vista de catálogo SYSCAT.DBAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado la autorización.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene la autorización.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
BINDADDAUTH	CHAR (1)		Autorización para crear paquetes. • N = No se mantiene • Y = Se mantiene
CONNECTAUTH	CHAR (1)		Autorización para conectar con la base de datos. • N = No se mantiene • Y = Se mantiene
CREATETABAUTH	CHAR (1)		Autorización para crear tablas. • N = No se mantiene • Y = Se mantiene
DBADMAUTH	CHAR (1)		Autorización DBADM. • N = No se mantiene • Y = Se mantiene
EXTERNALROUTINEAUTH	CHAR (1)		Autorización para crear rutinas externas. • N = No se mantiene • Y = Se mantiene
IMPLSCHEMAAUTH	CHAR (1)		Autorización para crear esquemas de forma implícita, creando objetos en esquemas no existentes. • N = No se mantiene • Y = Se mantiene
LOADAUTH	CHAR (1)		Autorización para utilizar el programa de utilidad de carga DB2. • N = No se mantiene • Y = Se mantiene
NOFENCEAUTH	CHAR (1)		Autorización para crear funciones definidas por el usuario no delimitadas. • N = No se mantiene • Y = Se mantiene

Tabla 106. Vista de catálogo SYSCAT.DBAUTH (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
QUIESCECONNECTAUTH	CHAR (1)		Autorización para acceder a la base de datos mientras está inactiva. • N = No se mantiene • Y = Se mantiene
LIBRARYADMAUTH	CHAR (1)		Reservado para una utilización futura.
SECURITYADMAUTH	CHAR (1)		Autorización para administrar la seguridad de la base de datos. • N = No se mantiene • Y = Se mantiene
SQLADMAUTH	CHAR (1)		Autorización para supervisar y ajustar las sentencias de SQL. • N = No se mantiene • Y = Se mantiene
WLMADMAUTH	CHAR (1)		Autorización para gestionar objetos WLM. • N = No se mantiene • Y = Se mantiene
EXPLAINAUTH	CHAR (1)		Autorización para explicar sentencias de SQL sin necesidad de privilegios reales sobre los objetos de la sentencia. • N = No se mantiene • Y = Se mantiene
DATAACCESSAUTH	CHAR (1)		Autorización para acceder a los datos. • N = No se mantiene • Y = Se mantiene
ACCESSCTRLAUTH	CHAR (1)		Autorización para otorgar y revocar privilegios de objetos de base de datos. • N = No se mantiene • Y = Se mantiene

SYSCAT.DBPARTITIONGROUPDEF

Cada fila representa una partición de base de datos que está contenida en un grupo de particiones de base de datos.

Tabla 107. Vista de catálogo SYSCAT.DBPARTITIONGROUPDEF

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DBPGNAME	VARCHAR (128)		Nombre del grupo de particiones de base de datos que contiene la partición de base de datos.
DBPARTITIONNUM	SMALLINT		Número de partición de una partición de base de datos que está contenida en el grupo de particiones de base de datos. Un número de partición válido está entre 0 y 999 inclusive.
IN_USE	CHAR (1)		Estado de la partición de base de datos. • A = La partición de base de datos recién añadida no está en la correlación de distribución, pero los contenedores correspondientes a los espacios de tablas del grupo de particiones de base de datos se han creado; la partición de base de datos se añade a la correlación de distribución cuando la operación de redistribución del grupo de particiones de base de datos finaliza satisfactoriamente. • D = La partición de base de datos se eliminará cuando la operación de redistribución del grupo de particiones de base de datos finalice satisfactoriamente. • T = La partición de base de datos recién añadida no está en la correlación de distribución, y se ha añadido mediante la cláusula WITHOUT TABLESPACES; los contenedores se deben añadir a los espacios de tablas del grupo de particiones de base de datos. • Y = La partición de base de datos está en la correlación de distribución.

SYSCAT.DBPARTITIONGROUPS

Cada fila representa un grupo de particiones de base de datos.

Tabla 108. Vista de catálogo SYSCAT.DBPARTITIONGROUPS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DBPGNAME	VARCHAR (128)		Nombre del grupo de particiones de base de datos.
OWNER	VARCHAR (128)		ID de autorización del propietario del grupo de particiones de base de datos.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
PMAP_ID	SMALLINT		Identificador de la correlación de distribución en la vista de catálogo SYSCAT.PARTITIONMAPS.
REDISTRIBUTE_PMAP_ID	SMALLINT		Identificador de la correlación de distribución que se utiliza actualmente para la redistribución; -1 si actualmente no se está realizando ninguna redistribución.
CREATE_TIME	TIMESTAMP		Hora de creación del grupo de particiones de base de datos.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario del grupo de particiones de base de datos.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.EVENTMONITORS

Cada fila representa un supervisor de sucesos.

Tabla 109. Vista de catálogo SYSCAT.EVENTMONITORS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
EVMONNAME	VARCHAR (128)		Nombre del supervisor de sucesos.
OWNER	VARCHAR (128)		ID de autorización del propietario del supervisor de sucesos.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TARGET_TYPE	CHAR (1)		Tipo del destino en el que se graban los datos del suceso. • F = Archivo • P = Conducto • T = Tabla • U = Tabla de sucesos sin formato
TARGET	VARCHAR (762)		Nombre del destino en el que se escriben los datos del supervisor de sucesos de archivo o de conexión. Para archivos, puede ser un nombre de vía de acceso absoluto o un nombre de vía de acceso relativo (relativo a la vía de acceso de la base de datos para la base de datos; se puede ver utilizando el mandato LIST ACTIVE DATABASES). Para conexiones, puede ser un nombre de vía de acceso absoluto.
MAXFILES	INTEGER	S	El número máximo de archivos de sucesos que permite este supervisor de sucesos en una vía de acceso de sucesos. El valor es nulo si no hay ningún máximo o si TARGET_TYPE no es 'F' (archivo).
MAXFILESIZE	INTEGER	S	Tamaño máximo (en páginas de 4K) que cada archivo de sucesos puede alcanzar antes de que el supervisor de sucesos cree un nuevo archivo. El valor es nulo si no hay ningún máximo o si TARGET_TYPE no es 'F' (archivo).
BUFFERSIZE	INTEGER	S	Tamaño del almacenamiento intermedio (en páginas de 4K) utilizado por los supervisores de sucesos con los destinos de archivo; nulo en caso contrario.
IO_MODE	CHAR (1)	S	Modalidad de entrada/salida (I/O) de archivo. • B = Bloqueado • N = No bloqueado • Valor nulo = TARGET_TYPE no es 'F' (file) ni 'T' (tabla)

Tabla 109. Vista de catálogo SYSCAT.EVENTMONITORS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRITE_MODE	CHAR (1)	S	Indica cómo maneja este supervisor de sucesos los datos de sucesos existentes cuando se activa el supervisor. • A = Adjuntar • R = Sustituir • Valor nulo = TARGET_TYPE no es 'F' (archivo)
AUTOSTART	CHAR (1)		Indica si este supervisor de sucesos se debe activar automáticamente cuando se inicia la base de datos. • N = No • Y = Sí
DBPARTITIONNUM	SMALLINT		Número de la partición de base de datos en la que el supervisor de sucesos ejecuta y anota cronológicamente los sucesos.
MONSCOPE	CHAR (1)		Ámbito de supervisión. • G = Global • L = Local • T = Cada partición de base de datos en la que haya espacio de tablas • Blanco = Supervisor de sucesos WRITE TO TABLE
EVMON_ACTIVATES	INTEGER		Número de veces que se ha activado el supervisor de sucesos.
NODENUM[1]	SMALLINT		Número de la partición de base de datos en la que el supervisor de sucesos ejecuta y anota cronológicamente los sucesos.
DEFINER[2]	VARCHAR (128)		ID de autorización del propietario del supervisor de sucesos.
REMARKS	VARCHAR (254)	S	Reservado para una utilización futura.

Nota:

1. La columna NODENUM se incluye por razones de compatibilidad con versiones anteriores. Consulte DBPARTITIONNUM.
2. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.EVENTS

Cada fila representa un suceso que se está supervisando. En general, un supervisor de sucesos supervisa varios sucesos.

Tabla 110. Vista de catálogo SYSCAT.EVENTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
EVMONNAME	VARCHAR (128)		Nombre del supervisor de sucesos que está supervisando este suceso.
TYPE	VARCHAR (128)		Tipo del suceso que se supervisa. Los valores posibles son: • ACTIVITIES • CONNECTIONS • DATABASE • DEADLOCKS • DETAILDEADLOCKS • LOCKING • PKGCACHEBASE • PKGCACHEDETAILED • STATEMENTS • TABLES • TABLESPACES • THRESHOLDVIOLATIONS • TRANSACTIONS • STATISTICS • UOW
FILTER	CLOB (64K)	S	Texto completo de la cláusula WHERE que se aplica a este suceso.

SYSCAT.EVENTTABLES

Cada fila representa la tabla de destino de un supervisor de sucesos que escribe en tablas SQL.

Tabla 111. Vista de catálogo SYSCAT.EVENTTABLES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
EVMONNAME	VARCHAR (128)		Nombre del supervisor de sucesos.
LOGICAL_GROUP	VARCHAR (128)		Nombre del grupo de datos lógicos. Los valores posibles son: • ACTIVITYHISTORY • BUFFERPOOL • CONN • CONNHEADER • CONTROL • DATAVAL • DB • DEADLOCK • DLCONN • DLLOCK • LOCKING • PKGCACHEBASE • PKGCACHEDETAILED • SCSTATS • STMT • STMTHIST • STMTVALS • SUBSECTION • TABLE • TABLESPACE • THRESHOLDVIOLATIONS • UOW • WCSTATS • WLSTATS • XACT
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla de destino.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla de destino.
PCTDEACTIVATE	SMALLINT		Un valor de porcentaje que especifica hasta qué punto debe estar lleno un espacio de tablas DMS para que un supervisor de sucesos lo desactive de forma automática. Establecido en 100 para los espacios de tablas SMS.

SYSCAT.FULLHIERARCHIES

Cada fila representa la relación entre una subtabla y una supertabla, un subtipo y un supertipo o una subvista y una supervista. Todas las relaciones jerárquicas, incluyendo las inmediatas, se incluyen en esta vista.

Tabla 112. Vista de catálogo SYSCAT.FULLHIERARCHIES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
METATYPE	CHAR (1)		Tipo de relación. • R = Entre tipos estructurados • U = Entre tablas con tipo • W = Entre vistas con tipo
SUB_SCHEMA	VARCHAR (128)		Nombre de esquema de subtipo, subtabla o subvista.
SUB_NAME	VARCHAR (128)		Nombre no calificado de subtipo, subtabla o subvista.
SUPER_SCHEMA	VARCHAR (128)	S	Nombre de esquema de supertipo, supertabla o supervista.
SUPER_NAME	VARCHAR (128)	S	Nombre no calificado de supertipo, supertabla o supervista.
ROOT_SCHEMA	VARCHAR (128)		Nombre de esquema de la tabla, vista o tipo que está en la raíz de la jerarquía.
ROOT_NAME	VARCHAR (128)		Nombre no calificado de la tabla, vista o tipo que está en la raíz de la jerarquía.

SYSCAT.FUNCMAPOPTIONS

Cada fila representa un valor de opción de correlación de funciones.

Tabla 113. Vista de catálogo SYSCAT.FUNCMAPOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCTION_MAPPING	VARCHAR (128)		Nombre de la correlación de funciones.
OPTION	VARCHAR (128)		Nombre de la opción de correlación de funciones.
SETTING	VARCHAR (2048)		Valor de la opción de correlación de funciones.

SYSCAT.FUNCMAPPARMOPTIONS

Cada fila representa un valor de opción de parámetro de correlación de funciones.

Tabla 114. Vista de catálogo SYSCAT.FUNCMAPPARMOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCTION_MAPPING	VARCHAR (128)		Nombre de la correlación de funciones.
ORDINAL	SMALLINT		Posición del parámetro
LOCATION	CHAR (1)		Ubicación del parámetro • L = Parámetro local • R = Parámetro remoto
OPTION	VARCHAR (128)		Nombre de la opción de parámetro de correlación de funciones.
SETTING	VARCHAR (2048)		Valor de la opción de parámetro de correlación de funciones.

SYSCAT.FUNCMAPPINGS

Cada fila representa una correlación de funciones.

Tabla 115. Vista de catálogo SYSCAT.FUNCMAPPINGS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCTION_MAPPING	VARCHAR (128)		Nombre de la correlación de funciones (puede ser generada por el sistema).
FUNCSCHEMA	VARCHAR (128)	S	Nombre de esquema de la función. Si el valor es nulo, se supone que la función es una función incorporada.
FUNCNAME	VARCHAR(1024)	S	Nombre no calificado de la función definida por el usuario o incorporada.
FUNCID	INTEGER	S	Identificador de la función.
SPECIFICNAME	VARCHAR (128)	S	Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
OWNER	VARCHAR (128)		ID de autorización del propietario de la correlación. 'SYSIBM' indica que se trata de una función incorporada.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
WRAPNAME	VARCHAR (128)	S	Derivador al que se aplica esta correlación.
SERVERNAME	VARCHAR (128)	S	Nombre de la fuente de datos.
SERVERTYPE	VARCHAR (30)	S	Tipo de fuente de datos al que se aplica esta correlación.
SERVERVERSION	VARCHAR (18)	S	Versión del tipo de servidor al que se aplica ésta correlación.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la correlación.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la correlación. 'SYSIBM' indica que se trata de una función incorporada.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.HIERARCHIES

Cada fila representa la relación entre una subtabla y su supertabla inmediata, un subtipo y su supertipo inmediato o una subvista y su supervista inmediata. Sólo las relaciones jerárquicas inmediatas se incluyen en esta vista.

Tabla 116. Vista de catálogo SYSCAT.HIERARCHIES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
METATYPE	CHAR (1)		Tipo de relación. • R = Entre tipos estructurados • U = Entre tablas con tipo • W = Entre vistas con tipo
SUB_SCHEMA	VARCHAR (128)		Nombre de esquema de subtipo, subtabla o subvista.
SUB_NAME	VARCHAR (128)		Nombre no calificado de subtipo, subtabla o subvista.
SUPER_SCHEMA	VARCHAR (128)		Nombre de esquema de supertipo, supertabla o supervista.
SUPER_NAME	VARCHAR (128)		Nombre no calificado de supertipo, supertabla o supervista.
ROOT_SCHEMA	VARCHAR (128)		Nombre de esquema de la tabla, vista o tipo que está en la raíz de la jerarquía.
ROOT_NAME	VARCHAR (128)		Nombre no calificado de la tabla, vista o tipo que está en la raíz de la jerarquía.

SYSCAT.HISTOGRAMTEMPLATEBINS

Cada fila representa un binario de plantilla de histograma.

Tabla 117. Vista de catálogo SYSCAT.HISTOGRAMTEMPLATEBINS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TEMPLATENAME	VARCHAR (128)	S	Nombre de la plantilla de histograma.
TEMPLATEID	INTEGER		Identificador de la plantilla de histograma.
BINID	INTEGER		Identificador del binario de plantilla de histograma.
BINUPPERVALUE	BIGINT		El valor superior de un único binario de la plantilla de histograma.

SYSCAT.HISTOGRAMTEMPLATES

Cada fila representa una plantilla de histograma.

Tabla 118. Vista de catálogo SYSCAT.HISTOGRAMTEMPLATES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TEMPLATEID	INTEGER		Identificador de la plantilla de histograma.
TEMPLATENAME	VARCHAR (128)		Nombre de la plantilla de histograma.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la plantilla de histograma.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez la plantilla de histograma.
NUMBINS	INTEGER		Número de binarios de la plantilla de histograma, incluido el último binario con valor superior no vinculado.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.HISTOGRAMTEMPLATEUSE

Cada fila representa una relación entre un objeto de gestión de carga de trabajo que puede utilizar plantillas de histograma y una plantilla de histograma.

Tabla 119. Vista de catálogo SYSCAT.HISTOGRAMTEMPLATEUSE

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TEMPLATENAME	VARCHAR (128)	S	Nombre de la plantilla de histograma.
TEMPLATEID	INTEGER		Identificador de la plantilla de histograma.
HISTOGRAMTYPE	CHAR (1)		El tipo de información recopilada por histogramas basados en esta plantilla. • C = Histograma de coste estimado de la actividad • E = Histograma de tiempo de ejecución de la actividad • I = Histograma de tiempo hasta la llegada de la actividad • L = Histograma de tiempo de vida de la actividad • Q = Histograma de tiempo de cola de la actividad • R = Histograma de tiempo de ejecución de la actividad
OBJECTTYPE	CHAR (1)		El tipo de objeto WLM. • b = Clase de servicio • k = Acción de trabajo • w = Carga de trabajo
OBJECTID	INTEGER		Identificador del objeto WLM.
SERVICECLASSNAME	VARCHAR (128)	S	Nombre de la clase de servicio.
PARENTSERVICECLASSNAME	VARCHAR (128)	S	Nombre de la clase de servicio padre de la subclase de servicio que emplea la plantilla de histograma.
WORKACTIONNAME	VARCHAR (128)	S	Nombre de la acción de trabajo que emplea la plantilla de histograma.
WORKACTIONSETNAME	VARCHAR (128)	S	Nombre del conjunto de acciones de trabajo en el que se encuentra la acción de trabajo que emplea la plantilla de histograma.
WORKLOADNAME	VARCHAR (128)	S	Nombre de la carga de trabajo que emplea la plantilla de histograma.

SYSCAT.INDEXAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado el privilegio CONTROL sobre un índice.

Tabla 120. Vista de catálogo SYSCAT.INDEXAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
CONTROLAUTH	CHAR (1)		Privilegio CONTROL. • N = No se mantiene • Y = Se mantiene

SYSCAT.INDEXCOLUSE

Cada fila representa una columna que interviene en un índice.

Tabla 121. Vista de catálogo SYSCAT.INDEXCOLUSE

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
COLNAME	VARCHAR (128)		Nombre de la columna.
COLSEQ	SMALLINT		Posición numérica de la columna en el índice (la posición inicial es 1).
COLORDER	CHAR (1)		Orden de los valores de esta columna del índice. Los valores posibles son: • A = Ascendente • D = Descendente • I = Columna INCLUDE (se pasa por alto el orden)
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para la columna; en caso contrario, valor nulo.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para la columna; de lo contrario, valor nulo.

SYSCAT.INDEXDEP

Cada fila representa una dependencia de un índice respecto de algún otro objeto. El índice depende de un objeto de tipo BTYPE y nombre BNAME, de modo que un cambio en el objeto afecta al índice.

Tabla 122. Vista de catálogo SYSCAT.INDEXDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • K = Paquete • L = Tabla desenlazada • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • Q = Secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • X = Extensión de índice • Z = Objeto XSR • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.

Tabla 122. Vista de catálogo SYSCAT.INDEXDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita el índice dependiente; en caso contrario, valor nulo.

SYSCAT.INDEXES

Cada fila representa un índice. Los índices en tablas con tipo se representan mediante dos filas: una para el ″índice lógico″ de la tabla con tipo y otro para el ″índice-H″ de la tabla de jerarquía.

Tabla 123. Vista de catálogo SYSCAT.INDEXES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
OWNER	VARCHAR (128)		ID de autorización del propietario del índice.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o apodo en el que se define el índice.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o apodo en el que se define el índice.
COLNAMES	VARCHAR(640)		Esta columna ya no se utiliza y se eliminará en el próximo release. Utilice SYSCAT.INDEXCOLUSE para obtener esta información.
UNIQUERULE	CHAR (1)		Norma de unicidad. • D = Permite duplicados • U = Unicidad • P = Implanta clave primaria
MADE_UNIQUE	CHAR (1)		• N = El índice permanece tal como se ha creado • Y = El índice era de no unicidad originalmente, pero se ha convertido en un índice de unicidad para dar soporte a una restricción de clave primaria o de unicidad. Si se elimina la restricción, el índice vuelve a ser de no unicidad.
COLCOUNT	SMALLINT		Número de columnas de la clave, más el número de columnas de inclusión, si hay alguna.
UNIQUE_COLCOUNT	SMALLINT		Número de columnas necesarias para una clave de unicidad. Siempre es <= COLCOUNT, y < COLCOUNT sólo si hay columnas de inclusión; -1 si el índice no tiene clave exclusiva (es decir, permite duplicados).

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDEXTYPE[5]	CHAR(4)		Tipo de índice. • BLOK = Índice de bloques • CLUS = Índice de clúster (controla la ubicación física de las filas recién insertadas) • DIM = Índice de bloques de dimensión • REG = Índice regular • XPTH = Índice de vías de acceso XML • XRGN = Índice de regiones XML • XVIL = Índice sobre columna XML (lógico) • XVIP = Índice sobre columna XML (físico)
ENTRYTYPE	CHAR (1)		• H = Esta fila representa un índice en una tabla de jerarquía • L = Esta fila representa un índice lógico en una tabla con tipo • Blanco = Esta fila representa un índice en una tabla sin tipo
PCTFREE	SMALLINT		Porcentaje de cada página de índice que se debe reservar durante la creación inicial del índice. Este espacio está disponible para inserciones de datos después de que se cree el índice.
IID	SMALLINT		Identificador del índice.
NLEAF	BIGINT		Número de páginas de índice; -1 si no se recopilan estadísticas.
NLEVELS	SMALLINT		Número de niveles de índice; -1 si no se recopilan estadísticas.
FIRSTKEYCARD	BIGINT		Número de valores de primera clave diferenciada; -1 si no se recopilan estadísticas.
FIRST2KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las dos primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST3KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las tres primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST4KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FULLKEYCARD	BIGINT		Número de valores de clave completa diferenciada; -1 si no se recopilan estadísticas.
CLUSTERRATIO[3]	SMALLINT		Grado de clúster de datos con el índice; -1 si no se recopilan estadísticas o si se recopilan estadísticas detalladas de índice (en este caso se utilizará CLUSTERFACTOR en su lugar).

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CLUSTERFACTOR[3]	DOUBLE		Mejor medición del grado de clúster; -1 si no se recopilan estadísticas o si el índice se define sobre un apodo.
SEQUENTIAL_PAGES	BIGINT		Número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas; -1 si no se recopilan estadísticas.
DENSITY	INTEGER		Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100; -1 si no se recopilan estadísticas.
USER_DEFINED	SMALLINT		1 si un usuario ha definido este índice y no se ha eliminado; 0 en caso contrario.
SYSTEM_REQUIRED	SMALLINT		• 1 si se cumple una de las condiciones siguientes: – Este índice es necesario para una restricción de clave primaria o de unicidad o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla con clústeres de varias dimensiones (MDC). - Se trata de un índice en la columna identificador de objeto (OID) de una tabla con tipo. • 2 si se cumplen estas dos condiciones: – Este índice es necesario para una restricción de clave primaria o exclusiva o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla MDC. - Se trata del índice en la columna OID de una tabla con tipo. • De lo contrario, 0.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado el índice.
STATS_TIME	TIMESTAMP	S	Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. El valor es nulo si hay estadísticas disponibles.
PAGE_FETCH_PAIRS[3]	VARCHAR(520)		Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. Serie de longitud cero si no hay datos disponibles.

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
MINPCTUSED	SMALLINT		Un valor entero distinto de cero indica que el índice está habilitado para la desfragmentación en línea y representa el porcentaje mínimo de espacio utilizado en una página antes de que se pueda intentar una fusión de página. El valor cero indica que no se intenta más fusiones de página.
REVERSE_SCANS	CHAR (1)		• N = El índice no da soporte a las exploraciones invertidas • Y = El índice da soporte a las exploraciones invertidas
INTERNAL_FORMAT	SMALLINT		Los valores posibles son: • 1 = El índice no tiene punteros que apunten hacia atrás • 2 o mayor = El índice tiene punteros que apuntan hacia atrás • 6 = El índice es un índice de bloques compuestos
COMPRESSION	CHAR (1)		Especifica si la compresión de índice está habilitada. • N = No activada • Y = Activada
IESCHEMA	VARCHAR (128)	S	Nombre de esquema de la extensión del índice. El valor es nulo para índices ordinarios.
IENAME	VARCHAR (128)	S	Nombre no calificado de la extensión del índice. El valor es nulo para índices ordinarios.
IEARGUMENTS	CLOB (64K)	S	Información externa del parámetro especificado cuando se crea el índice. El valor es nulo para índices ordinarios.
INDEX_OBJECTID	INTEGER		Identificador del objeto de índice.
NUMRIDS	BIGINT		Número total de identificadores de fila (RID) o identificadores de bloque (BID) del índice; -1 si no se conoce.
NUMRIDS_DELETED	BIGINT		Número total de identificadores de filas (o identificadores de bloque) del índice que están marcados como suprimidos, excluidos los identificadores de páginas de hojas en las que todos los identificadores de filas están marcados como suprimidos.
NUM_EMPTY_LEAFS	BIGINT		Número total de páginas de hoja del índice que tienen todos los identificadores de filas (o identificadores de bloqueo) marcados como suprimidos.
AVERAGE_RANDOM_FETCH_PAGES[1,2]	DOUBLE		Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice; -1 si no se conoce.

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AVERAGE_RANDOM_PAGES[2]	DOUBLE		Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales; -1 si no se conoce.
AVERAGE_SEQUENCE_GAP[2]	DOUBLE		Espacio entre las secuencias de páginas del índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas del índice que deben captarse de forma aleatoria entre las secuencias de las páginas del índice; -1 si no se conoce.
AVERAGE_SEQUENCE_FETCH_GAP[1,2]	DOUBLE		Espacio entre las secuencias de páginas de la tabla al captar utilizando el índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas de la tabla que deben captarse de forma aleatoria entre las secuencias de las páginas de la tabla; -1 si no se conoce.
AVERAGE_SEQUENCE_PAGES[2]	DOUBLE		Promedio de páginas del índice accesibles en secuencia (es decir, el número de páginas del índice que la captación previa detectaría que forman una secuencia); -1 si no se conoce.
AVERAGE_SEQUENCE_FETCH_PAGES[1,2]	DOUBLE		Promedio de páginas de la tabla accesibles en secuencia (es decir, el número de páginas de la tabla que la captación previa detectaría que forman una secuencia) al captar utilizando el índice; -1 si no se conoce.
TBSPACEID	INTEGER		Identificador del espacio de tablas del índice.
LEVEL2PCTFREE	SMALLINT		Porcentaje de cada página de nivel 2 de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice.
PAGESPLIT	CHAR (1)		Comportamiento de división de páginas del índice. • H = Alto • L = Bajo • S = Simétrico
AVGPARTITION_CLUSTERRATIO[3]	SMALLINT		Nivel de clúster de los datos dentro de una sola partición de datos. -1 si la tabla no está particionada, si no se recopilan estadísticas o si se recopilan estadísticas detalladas (en cuyo caso se utiliza AVGPARTITION_CLUSTERFACTOR en su lugar).
AVGPARTITION_CLUSTERFACTOR[3]	DOUBLE		Mejor medición del nivel de clúster dentro de una sola partición de datos. -1 si la tabla no está particionada, si no se recopilan estadísticas o si el índice está definido sobre un apodo.

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AVGPARTITION_PAGE_FETCH_PAIRS[3]	VARCHAR(520)		Una lista de pares de enteros en formato de caracteres. Cada par representa un tamaño potencial de la agrupación de almacenamientos intermedios en las captaciones de páginas correspondientes necesarias para acceder a una sola partición de datos desde la tabla. Serie de longitud cero si no hay datos disponibles o si la tabla no está particionada.
PCTPAGESSAVED	SMALLINT		Porcentaje aproximado de páginas guardadas en el índice como un resultado de la compresión de índice. -1 si no se recopilan estadísticas.
DATAPARTITION_CLUSTERFACTOR	DOUBLE		Una medición estadística del ″clúster″ de claves de índices con respecto a particiones de datos. Es un número comprendido entre 0 y 1; 1 representa clúster perfecto y 0 representa que no hay clúster.
INDCARD	BIGINT		Cardinalidad del índice. Puede ser distinta de la cardinalidad de la tabla para índices que no tienen una relación de uno a uno entre las filas de la tabla y las entradas de índice.
AVGLEAFKEYSIZE	INTEGER		Promedio del tamaño de clave de índice para las claves en páginas hojas del índice.
AVGNLEAFKEYSIZE	INTEGER		Promedio del tamaño de clave de índice para las claves en páginas no hojas del índice.
OS_PTR_SIZE	INTEGER		Tamaño de palabra de autorización bajo el cual se ha creado el índice. • 32 = 32 bits • 64 = 64 bits
COLLECTSTATISTICS	CHAR (1)		Especifica el modo en que se reúnen las estadísticas al crear índices. • D = Reunir estadísticas de índice detalladas • S = Reunir estadísticas de índice detalladas muestreadas • Y = Reunir estadísticas de índice básicas • Blanco = No reunir estadísticas de índice
DEFINER[4]	VARCHAR (128)		ID de autorización del propietario del índice.
LASTUSED	DATE		Fecha de última utilización del índice por parte de una sentencia DML o en que se ha utilizado para aplicar las restricciones de integridad referenciales. El valor por omisión es ’0001-01-01’. Este valor se actualiza de forma asíncrona.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Tabla 123. Vista de catálogo SYSCAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción

Nota:

1. Cuando se utilizan espacios de tablas DMS, no puede calcularse esta estadística.
2. No se recopilan estadísticas de captación previa durante una operación LOAD...STATISTICS YES o CREATE INDEX...COLLECT STATISTICS ni cuando el parámetro de configuración *seqdetect* está desactivado.
3. AVGPARTITION_CLUSTERRATIO, AVGPARTITION_CLUSTERFACTOR y AVGPARTITION_PAGE_FETCH_PAIRS miden el grado de clúster dentro de una sola partición de datos (clúster local). CLUSTERRATIO, CLUSTERFACTOR y PAGE_FETCH_PAIRS miden el grado de clúster en la tabla entera (clúster global). Los valores de clúster global y de clúster local pueden diferir significativamente si la clave de particionamiento de la tabla no es un prefijo de la clave de índice o cuando la clave de particionamiento de la tabla y la clave de índice son independientes, en términos lógicos, una de otra.
4. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.
5. Los índices XPTH, XRGN y XVIP no son reconocidos por ninguna interfaz de programación de aplicaciones que devuelvan metadatos de índice.

SYSCAT.INDEXEXPLOITRULES

Cada fila representa una norma de explotación de índice.

Tabla 124. Vista de catálogo SYSCAT.INDEXEXPLOITRULES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCID	INTEGER		Identificador de la función.
SPECID	SMALLINT		Número de la especificación de predicado.
IESCHEMA	VARCHAR (128)		Nombre de esquema de la extensión del índice.
IENAME	VARCHAR (128)		Nombre no calificado de la extensión del índice.
RULEID	SMALLINT		Identificador de la norma de explotación.
SEARCHMETHODID	SMALLINT		Identificador del método de búsqueda en la extensión de índice específica.
SEARCHKEY	VARCHAR(640)		Clave utilizada para explotar el índice.
SEARCHARGUMENT	VARCHAR (2700)		Argumentos de búsqueda utilizados para explotar el índice.
EXACT	CHAR (1)		• N = La búsqueda en el índice no es exacta en términos de evaluación del predicado • Y = La búsqueda en el índice es exacta en términos de evaluación del predicado

SYSCAT.INDEXEXTENSIONDEP

Cada fila representa una dependencia de una extensión de índice respecto de algún otro objeto. La extensión de índice depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta a la extensión de índice.

Tabla 125. Vista de catálogo SYSCAT.INDEXEXTENSIONDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
IESCHEMA	VARCHAR (128)		Nombre de esquema de la extensión del índice.
IENAME	VARCHAR (128)		Nombre no calificado de la extensión del índice.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • K = Paquete • L = Tabla desenlazada • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • Q = Secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • X = Extensión de índice • Z = Objeto XSR • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.

Tabla 125. Vista de catálogo SYSCAT.INDEXEXTENSIONDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita la extensión de índice dependiente; en caso contrario, valor nulo.

SYSCAT.INDEXEXTENSIONMETHODS

Cada fila representa un método de búsqueda. Una extensión de índice puede contener más de un método de búsqueda.

Tabla 126. Vista de catálogo SYSCAT.INDEXEXTENSIONMETHODS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
METHODNAME	VARCHAR (128)		Nombre del método de búsqueda.
METHODID	SMALLINT		El número del método en la extensión de índice.
IESCHEMA	VARCHAR (128)		Nombre de esquema de la extensión de índice en la que está definido este método.
IENAME	VARCHAR (128)		Nombre no calificado de la extensión de índice en la que está definido este método.
RANGEFUNCSCHEMA	VARCHAR (128)		Nombre de esquema de la función del rango.
RANGEFUNCNAME	VARCHAR (128)		Nombre no calificado de la función del rango.
RANGESPECIFICNAME	VARCHAR (128)		Nombre específico de la función de la función del rango.
FILTERFUNCSCHEMA	VARCHAR (128)	S	Nombre de esquema de la función de filtro.
FILTERFUNCNAME	VARCHAR (128)	S	Nombre no calificado de la función de filtro.
FILTERSPECIFICNAME	VARCHAR (128)	S	Nombre específico de la función de la función de filtro.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.INDEXEXTENSIONPARMS

Cada fila representa un parámetro de instancia de la extensión de índice o una columna de la clave fuente.

Tabla 127. Vista de catálogo SYSCAT.INDEXEXTENSIONPARMS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
IESCHEMA	VARCHAR (128)		Nombre de esquema de la extensión del índice.
IENAME	VARCHAR (128)		Nombre no calificado de la extensión del índice.
ORDINAL	SMALLINT		Número de secuencia del parámetro o de la columna clave.
PARMNAME	VARCHAR (128)		Nombre del parámetro o de la columna clave.
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos del parámetro o de la columna clave.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos del parámetro o de la columna clave.
LENGTH	INTEGER		Longitud del tipo de datos del parámetro o de la columna clave.
SCALE	SMALLINT		Escala del tipo de datos del parámetro o de la columna clave; 0 si no se aplica.
PARMTYPE	CHAR (1)		• K = Columna de clave fuente • P = Parámetro de instancia de extensión de índice
CODEPAGE	SMALLINT		Página de códigos del parámetro de la instancia de extensión del índice; 0 si no es de tipo serie.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para el parámetro; valor nulo en caso contrario.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para el parámetro; valor nulo en caso contrario.

SYSCAT.INDEXEXTENSIONS

Cada fila representa una extensión de índice.

Tabla 128. Vista de catálogo SYSCAT.INDEXEXTENSIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
IESCHEMA	VARCHAR (128)		Nombre de esquema de la extensión del índice.
IENAME	VARCHAR (128)		Nombre no calificado de la extensión del índice.
OWNER	VARCHAR (128)		ID de autorización del propietario de la extensión de índice.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
CREATE_TIME	TIMESTAMP		Hora en la que se definió la extensión de índice.
KEYGENFUNCSCHEMA	VARCHAR (128)		Nombre de esquema de la función de generación de claves.
KEYGENFUNCNAME	VARCHAR (128)		Nombre calificado de la función de generación de claves.
KEYGENSPECIFICNAME	VARCHAR (128)		Nombre de la instancia de generación de claves (puede ser un nombre generado por el sistema).
TEXT	CLOB(2M)		Texto completo de la sentencia CREATE INDEX EXTENSION.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la extensión de índice.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.INDEXOPTIONS

Cada fila representa un valor de opción específico del índice.

Tabla 129. Vista de catálogo SYSCAT.INDEXOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
OPTION	VARCHAR (128)		Nombre de la opción de índice.
SETTING	VARCHAR (2048)		Valor de la opción de índice.

SYSCAT.INDEXPARTITIONS

Cada fila representa un fragmento de índice particionado ubicado en una partición de datos. Nota:

- Las estadísticas de partición de índice representan una partición de base de datos si la tabla se crea en varias particiones de base de datos.

Tabla 130. Vista de catálogo SYSCAT.INDEXPARTITIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice.
INDNAME	VARCHAR (128)		Nombre no calificado del índice.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o apodo en el que se define el índice.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o apodo en el que se define el índice.
IID	SMALLINT		Identificador del índice.
INDPARTITIONTBSPACEID	INTEGER		Identificador para el espacio de tablas de la partición de índice.
INDPARTITIONOBJECTID	INTEGER		Identificador para el objeto de la partición de índice.
DATAPARTITIONID	INTEGER		Se corresponde con el DATAPARTITIONID encontrado en la vista SYSCAT.DATAPARTITIONS.
INDCARD	BIGINT		Cardinalidad de la partición de índice. Puede ser distinta de la cardinalidad de partición de datos correspondiente para índices particionados que no tienen una relación de uno a uno entre las filas de la partición de tabla y las entradas de índice.
NLEAF	BIGINT		Número de páginas hojas de la partición de índice; -1 si no se recopilan estadísticas.
NUM_EMPTY_LEAFS	BIGINT		Número total de páginas hojas de índice de la partición de índice que tienen todos los identificadores de fila (RID) o identificadores de bloqueo (BID) marcados como suprimidos.
NUMRIDS	BIGINT		Número total de identificadores de fila (RID) o identificadores de bloque (BID) de la partición de índice; -1 si no se conoce.
NUMRIDS_DELETED	BIGINT		Número total de identificadores de filas (RID) o identificadores de bloque (BID) de la partición de índice que están marcados como suprimidos, excluidos los identificadores de páginas hojas en las que todos los identificadores de filas están marcados como suprimidos.
FULLKEYCARD	BIGINT		Número de valores de clave completa diferenciada de la partición de índice; -1 si no se recopilan estadísticas.
NLEVELS	SMALLINT		Número de niveles de índice de la partición de índice; -1 si no se recopilan estadísticas.

Tabla 130. Vista de catálogo SYSCAT.INDEXPARTITIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CLUSTERRATIO	SMALLINT		Nivel de clúster de los datos con la partición de índice; -1 en cualquier de las situaciones siguientes: • No se recopilan estadísticas. • Se recopilan estadísticas detalladas de índice. En esta situación, se utilizará CLUSTERFACTOR en su lugar.
CLUSTERFACTOR	DOUBLE		Mejor medición del grado de clúster; -1 si no se recopilan estadísticas.
FIRSTKEYCARD	BIGINT		Número de valores de primera clave diferenciada; -1 si no se recopilan estadísticas.
FIRST2KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las dos primeras columnas de la clave de índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST3KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las tres primeras columnas de la clave de índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST4KEYCARD	BIGINT		Número de claves diferenciadas que utilizan las cuatro primeras columnas de la clave de índice; -1 si no se recopilan estadísticas o si no se aplica.
AVGLEAFKEYSIZE	INTEGER		Promedio del tamaño de clave de índice de las claves en las páginas hojas de la partición de índice; -1 si no se recopilan estadísticas.
AVGNLEAFKEYSIZE	INTEGER		Promedio del tamaño de clave de índice de las claves en las páginas que no son hojas de la partición de índice; -1 si no se recopilan estadísticas.
PCTFREE	SMALLINT		Porcentaje de cada página de índice que se debe reservar durante la creación inicial de la partición de índice. Este espacio está disponible para inserciones de datos después de que se cree la partición de índice.
PAGE_FETCH_PAIRS	VARCHAR(520)		Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la partición de índice con este índice utilizando dicho almacenamiento intermedio hipotético. Serie de longitud cero si no hay datos disponibles.
SEQUENTIAL_PAGES	BIGINT		Número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas; -1 si no se recopilan estadísticas.

Tabla 130. Vista de catálogo SYSCAT.INDEXPARTITIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DENSITY	INTEGER		Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por la partición de índice, expresada como un porcentaje (entero entre 0 y 100); -1 si no se recopilan estadísticas.
AVERAGE_SEQUENCE_GAP	DOUBLE		Espacio entre las secuencias de páginas del índice dentro de la partición de índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas del índice que deben captarse de forma aleatoria entre las secuencias de las páginas del índice; -1 si no se conoce.
AVERAGE_SEQUENCE_ FETCH_GAP	DOUBLE		Espacio entre las secuencias de páginas de la tabla al captar utilizando la partición de índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas de la partición de índice que deben captarse de forma aleatoria entre las secuencias de las páginas de la partición de índice; -1 si no se conoce.
AVERAGE_SEQUENCE_ PAGES	DOUBLE		Promedio de páginas del índice accesibles en secuencia (es decir, el número de páginas del índice que la captación previa detectaría que forman una secuencia); -1 si no se conoce.
AVERAGE_SEQUENCE_ FETCH_PAGES	DOUBLE		Promedio de páginas de la partición de datos accesibles en secuencia (es decir, el número de páginas de la partición de datos que la captación previa detectaría que forman una secuencia) al captar utilizando el índice; -1 si no se conoce.
AVERAGE_RANDOM_PAGES	DOUBLE		Promedio de páginas de la partición de datos aleatorias entre los accesos a páginas secuenciales; -1 si no se conoce.
AVERAGE_RANDOM_FETCH_ PAGES	DOUBLE		Promedio de páginas de la partición de datos aleatorias entre los accesos a páginas secuenciales al captar utilizando la partición de índice; -1 si no se conoce.
STATS_TIME	TIMESTAMP	S	Última vez que se ha realizado un cambio en las estadísticas registradas para esta partición de índice. El valor es nulo si hay estadísticas disponibles.
COMPRESSION	CHAR (1)		Especifica si la compresión de índice está habilitada. • N = No activada • Y = Activada
PCTPAGESSAVED	SMALLINT		Porcentaje aproximado de páginas guardadas en el índice como un resultado de la compresión de índice. -1 si no se recopilan estadísticas.

SYSCAT.INDEXXMLPATTERNS

Cada fila representa una cláusula de patrón en un índice sobre una columna XML.

Tabla 131. Vista de catálogo SYSCAT.INDEXXMLPATTERNS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INDSCHEMA	VARCHAR (128)		Nombre de esquema del índice lógico.
INDNAME	VARCHAR (128)		Nombre no calificado del índice lógico.
PINDNAME	VARCHAR (128)		Nombre no calificado del índice físico.
PINDID	SMALLINT		Identificador del índice físico.
TYPEMODEL	CHAR (1)		• Q = SQL DATA TYPE (Ignorar valores no válidos) • R = SQL DATA TYPE (Rechazar valores no válidos)
DATATYPE	VARCHAR (128)		Nombre del tipo de datos.
HASHED	CHAR (1)		Indica si el valor se ha generado o no de forma aleatoria. • N = No generado de forma aleatoria • Y = Generado de forma aleatoria
LENGTH	SMALLINT		Longitud VARCHAR(*n*); 0 en caso contrario.
PATTERNID	SMALLINT		Identificador del patrón.
PATTERN	CLOB(2M)	S	Definición del patrón.

Nota:

1. Cuando se crean índices sobre columnas XML, se crean índices lógicos que utilizan la información del patrón XML, lo que da lugar a la creación de índices de árbol-B con columnas clave generadas por DB2 para dar soporte a los índices lógicos. Se crea un índice físico para dar soporte al tipo de datos especificado en la cláusula xmltype de la sentencia CREATE INDEX.

SYSCAT.INVALIDOBJECTS

Cada fila representa un objeto no válido.

Tabla 132. Vista de catálogo SYSCAT.INVALIDOBJECTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto que se está creando o validando de nuevo.
OBJECTMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el objeto que se está creando o validando de nuevo. El valor es nulo si el objeto no pertenece a ningún módulo.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto que se está creando o validando de nuevo. Para las rutinas (OBJECTTYPE = 'F'), es el nombre específico.
ROUTINENAME	VARCHAR (128)	S	Nombre no calificado de la rutina.
OBJECTTYPE	CHAR (1)		Tipo de objeto que se está creando o validando de nuevo. Los valores posibles son: • B = Activador • F = Rutina • R = Tipo de datos definido por el usuario • V = Vista • v = Variable global
SQLCODE	INTEGER	S	SQLCODE devuelto en CREATE con errores de revalidación. El valor es nulo si el objeto no se ha revalidado nunca.
SQLSTATE	CHAR(5)	S	SQLSTATE devuelto en CREATE con errores de revalidación. El valor es nulo si el objeto no se ha revalidado nunca.
ERRORMESSAGE	VARCHAR(70)	S	Texto corto del mensaje asociado con SQLCODE.El valor es nulo si el objeto no se ha revalidado nunca.
LINENUMBER	INTEGER	S	Número de la línea en la que se produjo el error en objetos compilados. El valor es nulo si el objeto no es un objeto compilado.
INVALIDATE_TIME	TIMESTAMP		Hora a la que se ha invalidado por última vez el objeto.
LAST_REGEN_TIME	TIMESTAMP	S	Hora a la que se ha vuelto a validar por última vez el objeto. El valor es nulo si el objeto no se ha revalidado nunca.

SYSCAT.KEYCOLUSE

Cada fila representa una columna que participa en una clave definida por una restricción de unicidad, de clave primaria o de clave foránea.

Tabla 133. Vista de catálogo SYSCAT.KEYCOLUSE

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre de la restricción.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla que contiene la columna.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla que contiene la columna.
COLNAME	VARCHAR (128)		Nombre de la columna.
COLSEQ	SMALLINT		Posición numérica de la columna en la clave para la restricción (la posición inicial es 1). Si una restricción utiliza un índice existente, este valor es la posición numérica de la columna en el índice.

SYSCAT.MODULEAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado un privilegio sobre un módulo.

Tabla 134. Vista de catálogo SYSCAT.MODULEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		Otorgante de un privilegio
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene un privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
MODULEID	INTEGER		Identificador para el módulo al que se aplica este privilegio.
MODULESCHEMA	VARCHAR (128)		Nombre de esquema del módulo al que se aplica este privilegio.
MODULENAME	VARCHAR (128)		Nombre no calificado del módulo al que se aplica este privilegio.
EXECUTEAUTH	CHAR (1)		Privilegio para ejecutar objetos en el módulo identificado. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.MODULEOBJECTS

Cada fila representa una función, procedimiento, variable global, condición o tipo definido por el usuario que pertenece a un módulo.

Tabla 135. Vista de catálogo SYSCAT.MODULEOBJECTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTSCHEMA	VARCHAR(128)	N	Nombre de esquema del módulo.
OBJECTMODULENAME	VARCHAR(128)	N	Nombre no calificado del módulo al que pertenece el objeto.
OBJECTNAME	VARCHAR(128)	N	Nombre no calificado del objeto.
OBJECTTYPE	VARCHAR(9)	N	• CONDITION = El objeto es una condición • FUNCTION = El objeto es una función • PROCEDURE = El objeto es un procedimiento • TYPE = El objeto es un tipo de datos • VARIABLE = El objeto es una variable
PUBLISHED	CHAR(1)	N	Indica si se puede hacer referencia al objeto fuera de su módulo. • N = El objeto no está publicado • Y = El objeto está publicado
SPECIFICNAME	VARCHAR(128)	N	Nombre específico de rutina si OBJECTTYPE es 'FUNCTION', 'METHOD' o 'PROCEDURE'; en caso contrario, el valor es nulo.
USERDEFINED	CHAR(1)	N	Indica si el objeto es generado por el sistema o definido por un usuario. • N = El objeto es generado por el sistema • Y = El objeto es definido por un usuario

SYSCAT.MODULES

Cada fila representa un módulo.

Tabla 136. Vista de catálogo SYSCAT.MODULES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
MODULESCHEMA	VARCHAR (128)		Nombre de esquema del módulo.
MODULENAME	VARCHAR (128)		Nombre no calificado del módulo.
MODULEID	INTEGER		Identificador del módulo.
DIALECT	VARCHAR(10)		Dialecto fuente del módulo de SQL. Los valores posibles son: • DB2 SQL PL • PL/SQL • Blanco = No se aplica para un alias
OWNER	VARCHAR (128)		ID de autorización del propietario del módulo.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
MODULETYPE	CHAR (1)		Tipo de módulo. • A = Alias • M = Módulo • P = Paquete PL/SQL
BASE_MODULESCHEMA	VARCHAR (128)	S	Si MODULETYPE es 'A', contiene el nombre de esquema del módulo o el alias al que hace referencia este alias; el valor es nulo en caso contrario.
BASE_MODULENAME	VARCHAR (128)	S	Si MODULETYPE es 'A', contiene el nombre no calificado del módulo o el alias al que hace referencia este alias; el valor es nulo en caso contrario.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el módulo.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.NAMEMAPPINGS

Cada fila representa la correlación entre un objeto ″lógico″ (tabla o vista con tipo y sus columnas e índices, incluidas las columnas heredadas) y el objeto de ″implantación″ correspondiente (tabla de jerarquía o vista de jerarquía y sus columnas e índices) que implanta el objeto lógico.

Tabla 137. Vista de catálogo SYSCAT.NAMEMAPPINGS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPE	CHAR (1)		• C = Columna • I = Índice • U = Tabla con tipo
LOGICAL_SCHEMA	VARCHAR (128)		Nombre de esquema del objeto lógico.
LOGICAL_NAME	VARCHAR (128)		Nombre no calificado del objeto lógico.
LOGICAL_COLNAME	VARCHAR (128)	S	Nombre de la columna lógica si TYPE = 'C'; valor nulo en caso contrario.
IMPL_SCHEMA	VARCHAR (128)		Nombre de esquema del objeto de implantación que implanta el objeto lógico.
IMPL_NAME	VARCHAR (128)		Nombre no calificado del objeto de implantación que implanta el objeto lógico.
IMPL_COLNAME	VARCHAR (128)	S	Nombre de la columna de implantación si TYPE = 'C'; valor nulo en caso contrario.

SYSCAT.NICKNAMES

Cada fila representa un apodo.

Tabla 138. Vista de catálogo SYSCAT.NICKNAMES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema del apodo.
TABNAME	VARCHAR (128)		Nombre no calificado del apodo.
OWNER	VARCHAR (128)		ID de autorización del propietario de la tabla, vista, alias o apodo.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
STATUS	CHAR (1)		Estado del objeto. • C = Pendiente de establecer integridad • N = Normal • X = No operativo
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el objeto.
STATS_TIME	TIMESTAMP	S	Hora a la que se ha hecho por última vez un cambio en las estadísticas registradas correspondientes a este objeto. El valor es nulo si no se recopilan estadísticas.
COLCOUNT	SMALLINT		Número de columnas, incluidas las columnas heredadas (si las hay).
TABLEID	SMALLINT		Identificador interno del objeto lógico.
TBSPACEID	SMALLINT		Identificador lógico interno del espacio de tablas primario correspondiente a este objeto.
CARD	BIGINT		Número total de filas de la tabla; -1 si no se recopilan estadísticas.
NPAGES	BIGINT		Número total de páginas en las que hay filas del apodo; -1 si no se recopilan estadísticas.
FPAGES	BIGINT		Número total de páginas; -1 si no se recopilan estadísticas.
OVERFLOW	BIGINT		Número total de registros de desbordamiento; -1 si no se recopilan estadísticas.
PARENTS	SMALLINT	S	Número de tablas padre de este objeto; es decir, el número de restricciones de referencia de las que depende este objeto.
CHILDREN	SMALLINT	S	Número de tablas dependientes de este objeto; es decir, el número de restricciones de referencia de las que depende este objeto.
SELFREFS	SMALLINT	S	Número de restricciones de referencia propia para este objeto; es decir, el número de restricciones de referencia en las que este objeto es tanto padre como dependiente.
KEYCOLUMNS	SMALLINT	S	Número de columnas de la clave primaria.

Tabla 138. Vista de catálogo SYSCAT.NICKNAMES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
KEYINDEXID	SMALLINT	S	Identificador de índice correspondiente al índice de clave primaria; 0 o el valor nulo si no hay clave primaria.
KEYUNIQUE	SMALLINT		Número de restricciones de clave de unicidad (que no sean la restricción de clave primara) definidas en este objeto.
CHECKCOUNT	SMALLINT		Número de restricciones de comprobación definidas en este objeto.
DATACAPTURE	CHAR (1)		• L = El apodo interviene en la replicación de datos, incluida la replicación de las columnas LONG VARCHAR y LONG VARGRAPHIC • N = El apodo no interviene en la replicación de datos • Y = El apodo participa en la replicación de datos
CONST_CHECKED	CHAR(32)		• El byte 1 representa restricción de clave foránea. • El byte 2 representa restricción de comprobación. • El byte 5 representa la tabla de consultas materializadas. • El byte 6 representa columna generada. • El byte 7 representa la tabla dispuesta con antelación. • El byte 8 representa restricción de particionamiento de datos. • Otros bytes están reservados para su utilización en el futuro. Los valores posibles son: • F = En el byte 5, la tabla de consultas materializadas no puede renovarse de forma incremental. En el byte 7, el contenido de la tabla dispuesta con antelación es incompleto y no puede utilizarse para renovaciones incrementales de la tabla de consultas materializadas asociada. • N = No comprobado • U = Comprobado por el usuario • W = Estaba en el estado 'U' cuando la tabla se colocó en el estado de pendiente de establecer integridad • Y = Comprobado por el sistema
PARTITION_MODE	CHAR (1)		Reservado para una utilización futura.
STATISTICS_PROFILE	CLOB(10M)	S	Mandato RUNSTATS utilizado para registrar un perfil estadístico para el objeto.

Tabla 138. Vista de catálogo SYSCAT.NICKNAMES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ACCESS_MODE	CHAR (1)		Estado de restricción de acceso del objeto. Estos estados sólo se aplican a objetos que están en estado pendiente de establecer integridad o a objetos procesados por una sentencia SET INTEGRITY. Los valores posibles son: • D = No hay movimiento de datos • F = Acceso completo • N = No hay acceso • R = Acceso de sólo lectura
CODEPAGE	SMALLINT		Página de códigos del objeto. Es la página de códigos por omisión utilizada para todas las columnas de caracteres, activadores, restricciones de comprobación y columnas generadas por expresión.
REMOTE_TABLE	VARCHAR (128)	S	Nombre no calificado del objeto de fuente de datos específico (como, por ejemplo, una tabla o una vista) para el que se ha creado el apodo.
REMOTE_SCHEMA	VARCHAR (128)	S	Nombre de esquema del objeto de fuente de datos específico (como, por ejemplo, una tabla o una vista) para el que se ha creado el apodo.
SERVERNAME	VARCHAR (128)	S	Nombre de la fuente de datos que contiene la tabla o la vista para la cual se ha creado el apodo.
REMOTE_TYPE	CHAR (1)	S	Tipo de objeto en la fuente de datos. • A = Alias • N = Apodo • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • V = Vista (sin tipo)
CACHINGALLOWED	VARCHAR(1)		• N = No se permite el almacenamiento en antememoria • Y = Se permite el almacenamiento en antememoria
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la tabla, vista, alias o apodo.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.PACKAGEAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado uno o más privilegios sobre un paquete.

Tabla 139. Vista de catálogo SYSCAT.PACKAGEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
PKGSCHEMA	VARCHAR (128)		Nombre de esquema del paquete.
PKGNAME	VARCHAR (128)		Nombre no calificado del paquete.
CONTROLAUTH	CHAR (1)		Privilegio CONTROL. • N = No se mantiene • Y = Se mantiene
BINDAUTH	CHAR (1)		Privilegio para vincular el paquete. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
EXECUTEAUTH	CHAR (1)		Privilegio para ejecutar el paquete. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.PACKAGEDEP

Cada fila representa una dependencia de un paquete sobre algún otro objeto. El paquete depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta al paquete.

Tabla 140. Vista de catálogo SYSCAT.PACKAGEDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PKGSCHEMA	VARCHAR (128)		Nombre de esquema del paquete.
PKGNAME	VARCHAR (128)		Nombre no calificado del paquete.
BINDER	VARCHAR (128)		Vinculador del paquete.
BINDERTYPE	CHAR (1)		• U = El vinculador es un usuario individual
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • D = Definición de servidor • F = Rutina • G = Tabla temporal de usuario • I = Índice • M = Correlación de funciones • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • P = Tamaño de página • Q = Objeto de secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • Z = Objeto XSR • m = Módulo • n = Grupo de particiones de base de datos • q = Alias de secuencia • u = Alias de módulo • v = Variable global
BSCHEMA	VARCHAR (128)		Nombre de esquema de un objeto del que depende el paquete.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado de un objeto del que depende el paquete.

Tabla 140. Vista de catálogo SYSCAT.PACKAGEDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.
TABAUTH	SMALLINT	S	Si BTYPE es 'O', 'S', 'T', 'U', 'V' o 'W', codifica los privilegios que necesita este paquete (SELECT, INSERT, UPDATE, o DELETE). El vector de bits se define en SQL.H.
VARAUTH	SMALLINT	S	Si BTYPE es 'v', codifica los privilegios que necesita este paquete (READ o WRITE). El vector de bits se define en SQL.H.
UNIQUE_ID	CHAR(8) FOR BIT DATA		Identificador de un paquete específico cuando existen varios paquetes con el mismo nombre.
PKGVERSION	VARCHAR(64)	S	Identificador de versión del paquete.

Nota:

1. Si se elimina una instancia de función con dependencias, el paquete se coloca en un estado ″no operativo″ y se debe volver a vincular de forma explícita. Si se elimina cualquier otro objeto con dependencias, el paquete se coloca en un estado ″no válido″ y el sistema intentará volver a vincular el paquete automáticamente cuando se haga referencia al mismo por primera vez.

SYSCAT.PACKAGES

Cada fila representa un paquete que se ha creado mediante la vinculación de un programa de aplicación.

Tabla 141. Vista de catálogo SYSCAT.PACKAGES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PKGSCHEMA	VARCHAR (128)		Nombre de esquema del paquete.
PKGNAME	VARCHAR (128)		Nombre no calificado del paquete.
BOUNDBY	VARCHAR (128)		ID de autorización del vinculador y propietario del paquete.
BOUNDBYTYPE	CHAR (1)		• U = El vinculador y propietario es un usuario individual
OWNER	VARCHAR (128)		ID de autorización del vinculador y propietario del paquete.
OWNERTYPE	CHAR (1)		• U = El vinculador y propietario es un usuario individual
DEFAULT_SCHEMA	VARCHAR (128)		Nombre de esquema por omisión para nombres no calificados en sentencias de SQL estático.
VALID[1]	CHAR (1)		• N = Necesita revinculación • V = Validar en el momento de la ejecución • X = El paquete no está operativo porque alguna instancia de función de la que depende se ha eliminado; se tiene que revincular de forma explícita • Y = Válido
UNIQUE_ID	CHAR(8) FOR BIT DATA		Identificador de un paquete específico cuando existen varios paquetes con el mismo nombre.
TOTAL_SECT	SMALLINT		Número de secciones del paquete.
FORMAT	CHAR (1)		Formato de fecha y hora asociado al paquete. • 0 = Formato asociado con el código territorial del cliente • 1 = USA • 2 = EUR • 3 = ISO • 4 = JIS • 5 = LOCAL
ISOLATION	CHAR (2)	S	Nivel de aislamiento. • CS = Estabilidad del cursor • RR = Lectura repetible • RS = Estabilidad de lectura • UR = Lectura no confirmada
CONCURRENTACCESSRESOLUTION	CHAR (1)	S	Valor de la opción de vinculación CONCURRENTACCESSRESOLUTION: • U = USE CURRENTLY COMMITTED • W = WAIT FOR OUTCOME • Blanco = Sin especificar
BLOCKING	CHAR (1)	S	Opción de bloqueo del cursor. • B = Bloqueo de todos los cursores • N = Sin bloqueo • U = Bloqueo de cursores no ambiguos

Tabla 141. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INSERT_BUF	CHAR (1)		Establecimiento de la opción de vinculación INSERT (se aplica a sistemas de bases de datos particionadas). • N = Las inserciones no se colocan en almacenamiento intermedio • Y = Las inserciones se colocan en almacenamiento intermedio en la partición de base de datos del coordinador para minimizar el tráfico entre particiones de base de datos
LANG_LEVEL	CHAR (1)	S	Establecimiento de la opción de vinculación LANGLEVEL. • 0 = SAA1 • 1 = MIA • 2 = SQL92E
FUNC_PATH	CLOB (2K)		Vía de acceso de SQL en vigor en el momento en que se vinculó el paquete.
QUERYOPT	INTEGER		La clase de optimización bajo la que se ha vinculado este paquete. Utilizado para operaciones de revinculación.
EXPLAIN_LEVEL	CHAR (1)		Indica si se ha invocado Explain utilizando la opción de vinculación EXPLAIN o EXPLSNAP. • P = Nivel de selección del paquete • Blanco = No se invoca Explain
EXPLAIN_MODE	CHAR (1)		Valor de la opción de vinculación EXPLAIN. • A = Todo • N = No • R = REOPT • Y = Sí
EXPLAIN_SNAPSHOT	CHAR (1)		Valor de la opción de vinculación EXPLSNAP. • A = Todo • N = No • R = REOPT • Y = Sí
SQLWARN	CHAR (1)		Indica si se devuelven o no a la aplicación los SQLCODE positivos resultantes de las sentencias de SQL dinámico. • N = No, se suprimen • Y = Sí
SQLMATHWARN	CHAR (1)		Valor del parámetro de configuración de base de datos *dft_sqlmathwarn* en el momento de la vinculación. Indica si los errores aritméticos y de conversión de recuperación devuelven avisos y valores nulos (indicador -2), lo que permite que el proceso de consulta continúe siempre que sea posible. • N = No, se devuelven errores • Y = Sí, se devuelven avisos
CREATE_TIME	TIMESTAMP		Hora a la que se ha vinculado el paquete por primera vez.
EXPLICIT_BIND_TIME	TIMESTAMP		Hora a la que este paquete se modificó por última vez: • BIND • REBIND (explícito)
LAST_BIND_TIME	TIMESTAMP		Hora a la que el paquete se modificó por última vez: • BIND • REBIND (explícito) • REBIND (implícito)
ALTER_TIME	TIMESTAMP		Hora a la que este paquete se modificó por última vez: • BIND • REBIND (explícito) • REBIND (implícito) • ALTER PACKAGE

Tabla 141. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CODEPAGE	SMALLINT		Página de códigos de la aplicación en el momento de la vinculación; -1 si no se conoce.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para el paquete.
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para el paquete.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en el paquete.
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en el paquete.
DEGREE	CHAR(5)		Grado de paralelismo intrapartición que se especificó cuando se vinculó el paquete. • 1 = Sin paralelismo • 2-32767 = Límite especificado por el usuario • ANY = Grado determinado por el sistema (no se especifica límite)
MULTINODE_PLANS	CHAR (1)		• N = El paquete no se ha vinculado en un entorno de bases de datos particionadas • Y = El paquete se ha vinculado en un entorno de bases de datos particionadas
INTRA_PARALLEL	CHAR (1)		Utilización del paralelismo intrapartición por las sentencias de SQL estático dentro del paquete. • F = Una o varias sentencias de SQL estático del paquete pueden utilizar paralelismo intrapartición; se ha inhabilitado este paralelismo para su uso en un sistema que no está configurado para paralelismo intrapartición. • N = Ninguna sentencia de SQL estático utiliza paralelismo intrapartición • Y = Una o más sentencias de SQL estático del paquete utilizan paralelismo intrapartición
VALIDATE	CHAR (1)		Indica que la comprobación de validez se puede diferir hasta el momento de la ejecución. • B = Toda comprobación debe realizarse en el momento de la vinculación • R = La validación de tablas, vistas y privilegios que no existan en el momento de la vinculación se realiza en el momento de la ejecución

Tabla 141. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DYNAMICRULES	CHAR (1)		• B = BIND; las sentencias de SQL dinámico se ejecutan con comportamiento DYNAMICRULES BIND • D = DEFINERBIND; cuando el paquete se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento DEFINE; cuando el paquete no se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento BIND • E = DEFINERRUN; cuando el paquete se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento DEFINE; cuando el paquete no se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento RUN • H = INVOKEBIND; cuando el paquete se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento INVOKE; cuando el paquete no se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento BIND • I = INVOKERUN; cuando el paquete se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento INVOKE; cuando el paquete no se ejecuta dentro de un contexto de rutina, las sentencias de SQL dinámico del paquete se ejecutan con comportamiento RUN • R = RUN; las sentencias de SQL dinámico se ejecutan con comportamiento RUN; es el valor por omisión
SQLERROR	CHAR (1)		Opción SQLERROR en el submandato más reciente que ha vinculado o revinculado el paquete. • C = CONTINUE; crea un paquete, aunque se produzcan errores al vincular sentencias de SQL • N = NOPACKAGE; no se crea un paquete o un archivo de vinculación si se produce un error
REFRESHAGE	DECIMAL(20,6)		Duración de la indicación de fecha y hora que indica el intervalo máximo de tiempo entre la ejecución de una sentencia REFRESH TABLE para una tabla de consultas materializadas (MQT) y el momento en que la MQT se utiliza en lugar de una tabla base.
FEDERATED	CHAR (1)		• N = La opción de vinculación o de preparación FEDERATED está desactivada • U = La opción de vinculación o de preparación FEDERATED no se ha especificado • Y = La opción de vinculación o de preparación FEDERATED está activada
TRANSFORMGROUP	VARCHAR(1024)	S	Valor de la opción de vinculación TRANSFORM GROUP; el valor es nulo si no se especifica ningún grupo de transformación.
REOPTVAR	CHAR (1)		Indica si la vía de acceso se vuelve a determinar en el momento de la ejecución utilizando valores de variables de entrada. • A = La vía de acceso se vuelve a optimizar para cada petición OPEN o EXECUTE • N = La vía de acceso se determina en el momento de la vinculación • O = La vía de acceso sólo se vuelve a optimizar en la primera petición OPEN o EXECUTE; luego se coloca en antememoria
OS_PTR_SIZE	INTEGER		Tamaño de las palabras para la plataforma en la que se ha creado el paquete: • 32 = El paquete es un paquete de 32 bits • 64 = El paquete es un paquete de 64 bits

Tabla 141. Vista de catálogo SYSCAT.PACKAGES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PKGVERSION	VARCHAR(64)		Identificador de versión del paquete.
STATICREADONLY	CHAR (1)		Indica si los cursores estáticos se van a tratar o no como READ ONLY. Los valores posibles son: • N = Los cursores estáticos adoptan los atributos que se generarían normalmente para el texto de sentencia dado y el valor de la opción de precompilación LANGLEVEL • Y = Cualquier cursor estático que no contenga la cláusula FOR UPDATE o FOR READ ONLY se considera READ ONLY
FEDERATED_ASYNCHRONY	INTEGER		Indica el límite de asincronía (el número de ATQ del plan) como una opción de vinculación cuando se vinculó el paquete. • 0 = Sin asincronía • *n* = Límite especificado por el usuario (32.767 máximo) • -1 = Grado de asincronía determinado por el sistema • -2 = Grado de asincronía no especificado Para un sistema no federado, el valor es 0.
ANONBLOCK	CHAR (1)		• N = El paquete no está asociado con un bloque anónimo • Y = El paquete está asociado con un bloque anónimo
OPTPROFILESCHEMA	VARCHAR (128)	S	Valor del esquema de perfil de optimización como parte de la opción de vinculación OPTPROFILE.
OPTPROFILENAME	VARCHAR (128)	S	Valor del nombre de perfil de optimización como parte de la opción de vinculación OPTPROFILE.
PKGID	BIGINT		Identificador del paquete.
DBPARTITIONNUM	SMALLINT		Número de la partición de base de datos a la que estaba vinculado el paquete.
DEFINER[2]	VARCHAR (128)		ID de autorización del vinculador y propietario del paquete.
PKG_CREATE_TIME[3]	TIMESTAMP		Hora a la que se ha vinculado el paquete por primera vez.
APREUSE	CHAR (1)		• El compilador de consultas no intentará reutilizar planes de acceso. • Y = Los planes de acceso de este paquete deberían reutilizarse, es decir, en el momento de volver a vincular, el compilador de consultas intentará seleccionar planes iguales a los que se encuentran actualmente en el paquete
EXTENDEDINDICATOR	CHAR (1)		• N = No se reconocen los valores de la variable de indicador ampliada • Y = Se reconocen los valores de la variable de indicador ampliada
LASTUSED	DATE		Fecha en la que se ejecutó por última vez una sentencia del paquete. Esta columna no se actualiza en el caso de un paquete asociado con un bloque asíncrono. El valor por omisión es '0001-01-01'. Este valor se actualiza de forma asíncrona.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. Si se elimina una instancia de función con dependencias, el paquete se coloca en un estado ″no operativo″ y se debe volver a vincular de forma explícita. Si se elimina cualquier otro objeto con dependencias, el paquete se coloca en un estado ″no válido″ y el sistema intentará volver a vincular el paquete automáticamente cuando se haga referencia al mismo por primera vez.
2. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.
3. La columna PKG_CREATE_TIME se incluye por razones de compatibilidad con versiones anteriores. Consulte CREATE_TIME.

SYSCAT.PARTITIONMAPS

Cada fila representa una correlación de distribución que se utiliza para distribuir las filas de una tabla entre las particiones de base de datos de un grupo de particiones de base de datos, en función de la generación aleatoria de la distribución de la tabla.

Tabla 142. Vista de catálogo SYSCAT.PARTITIONMAPS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PMAP_ID	SMALLINT		Identificador de la correlación de distribución.
PARTITIONMAP	BLOB (65536)		Correlación de distribución, un vector de 32.768 enteros de dos bytes para un grupo de particiones de base de datos de varias particiones. Para un grupo de particiones de base de datos de una sola partición, hay una entrada que indica el número de partición de la partición individual.

SYSCAT.PASSTHRUAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado autorización de paso para consultar una fuente de datos.

Tabla 143. Vista de catálogo SYSCAT.PASSTHRUAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SERVERNAME	VARCHAR (128)		Nombre de la fuente de datos a la que se otorga autorización.

SYSCAT.PREDICATESPECS

Cada fila representa una especificación del predicado.

Tabla 144. Vista de catálogo SYSCAT.PREDICATESPECS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCSCHEMA	VARCHAR (128)		Nombre de esquema de la función.
FUNCNAME	VARCHAR (128)		Nombre no calificado de la función.
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la función.
FUNCID	INTEGER		Identificador de la función.
SPECID	SMALLINT		Número de esta especificación de predicado.
CONTEXTOP	CHAR(8)		Operador de comparación, uno de los operadores relacionales incorporados (=, <, >, >=, etc.)
CONTEXTEXP	CLOB(2M)		Constante o una expresión SQL.
FILTERTEXT	CLOB (32K)	S	Texto de la expresión de filtro de datos.

SYSCAT.REFERENCES

Cada fila representa una restricción de integridad referencial (clave foránea).

Tabla 145. Vista de catálogo SYSCAT.REFERENCES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre de la restricción.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla dependiente.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla dependiente.
OWNER	VARCHAR (128)		ID de autorización del propietario de la restricción.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
REFKEYNAME	VARCHAR (128)		Nombre de la clave padre.
REFTABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla padre.
REFTABNAME	VARCHAR (128)		Nombre no calificado de la tabla padre.
COLCOUNT	SMALLINT		El número de columnas de la clave foránea.
DELETERULE	CHAR (1)		Norma de supresión. • A = NO ACTION • C = CASCADE • N = SET NULL • R = RESTRICT
UPDATERULE	CHAR (1)		Norma de actualización. • A = NO ACTION • R = RESTRICT
CREATE_TIME	TIMESTAMP		Hora a la que se ha definido la restricción.
FK_COLNAMES	VARCHAR(640)		Esta columna ya no se utiliza y se eliminará en un próximo release. Utilice SYSCAT.KEYCOLUSE para obtener esta información.
PK_COLNAMES	VARCHAR(640)		Esta columna ya no se utiliza y se eliminará en un próximo release. Utilice SYSCAT.KEYCOLUSE para obtener esta información.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la restricción.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.ROLEAUTH

Cada fila representa un rol otorgado a un usuario, grupo, rol o PUBLIC.

Tabla 146. Vista de catálogo SYSCAT.ROLEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		ID de autorización que otorgó el rol.
GRANTORTYPE	CHAR (1)		• U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		ID de autorización al que se le ha otorgado el rol.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
ROLENAME	VARCHAR (128)		Nombre del rol.
ROLEID	INTEGER		Identificador del rol.
ADMIN	CHAR (1)		Privilegio para otorgar o revocar el rol a otros o para comentar el rol. • N = No se mantiene • Y = Se mantiene

SYSCAT.ROLES

Cada fila representa un rol.

Tabla 147. Vista de catálogo SYSCAT.ROLES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROLENAME	VARCHAR (128)		Nombre del rol.
ROLEID	INTEGER		Identificador del rol.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado el rol.
AUDITPOLICYID	INTEGER	S	Identificador de la política de comprobación.
AUDITPOLICYNAME	VARCHAR (128)	S	Nombre de la política de comprobación.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.ROUTINEAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado el privilegio EXECUTE sobre:

- una rutina específica (función, método o procedimiento) en la base de datos que no está definida en un módulo
- todas las rutinas de un esquema específico en la base de datos que no están definidas en un módulo

Tabla 148. Vista de catálogo SYSCAT.ROUTINEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que otorga el privilegio. 'SYSIBM' si el privilegio lo ha otorgado el sistema.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SCHEMA	VARCHAR (128)		Nombre de esquema de la rutina.
SPECIFICNAME	VARCHAR (128)	S	Nombre específico de la rutina. Si SPECIFICNAME es el valor nulo y ROUTINETYPE no es 'M', el privilegio se aplica a todas las rutinas del tipo especificado en ROUTINETYPE en el esquema especificado en SCHEMA. Si SPECIFICNAME es el valor nulo y ROUTINETYPE es 'M', el privilegio se aplica a todos los métodos correspondientes al tipo de asunto especificado por TYPENAME en el esquema especificado por TYPESCHEMA. Si SPECIFICNAME es el valor nulo, ROUTINETYPE es 'M' y tanto TYPENAME como TYPESCHEMA son nulos, el privilegio se aplica a todos los métodos correspondientes a todos los tipos del esquema.
TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo correspondiente al método. El valor es nulo si ROUTINETYPE no es 'M'.
TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo correspondiente al método. El valor es nulo si ROUTINETYPE no es 'M'. Si TYPENAME es el valor nulo y ROUTINETYPE es 'M', el privilegio se aplica a todos los métodos correspondientes a cualquier tipo de asunto si están en el esquema especificado por SCHEMA.

Tabla 148. Vista de catálogo SYSCAT.ROUTINEAUTH (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINETYPE	CHAR (1)		Tipo de la rutina. • F = Función • M = Método • P = Procedimiento
EXECUTEAUTH	CHAR (1)		Privilegio para ejecutar la rutina. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
GRANT_TIME	TIMESTAMP		Hora a la que se ha otorgado el privilegio.

SYSCAT.ROUTINEDEP

Cada fila representa una dependencia de una rutina sobre algún otro objeto. La rutina depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta a la rutina.

Tabla 149. Vista de catálogo SYSCAT.ROUTINEDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)		Nombre de esquema de la rutina que tiene dependencias sobre otro objeto.
ROUTINEMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo.
SPECIFICNAME	VARCHAR (128)		Nombre específico de la rutina que tiene dependencias sobre otro objeto.
ROUTINEMODULEID	INTEGER	S	Identificador para el módulo del objeto que tiene dependencias sobre otro objeto.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • K = Paquete • L = Tabla desenlazada • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • Q = Secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • X = Extensión de índice • Z = Objeto XSR • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.

Tabla 149. Vista de catálogo SYSCAT.ROUTINEDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita la rutina dependiente; en caso contrario, valor nulo.
ROUTINENAME	VARCHAR (128)		Esta columna ya no se utiliza y se eliminará en un próximo release. Consulte SPECIFICNAME.

SYSCAT.ROUTINEOPTIONS

Cada fila representa un valor de opción específico de la rutina.

Tabla 150. Vista de catálogo SYSCAT.ROUTINEOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)		Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINENAME	VARCHAR (128)		Nombre no calificado de la rutina.
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
OPTION	VARCHAR (128)		Nombre de la opción de rutina federada.
SETTING	VARCHAR (2048)		Valor de la opción de rutina federada.

SYSCAT.ROUTINEPARMOPTIONS

Cada fila representa un valor de opción específico del parámetro de rutina.

Tabla 151. Vista de catálogo SYSCAT.ROUTINEPARMOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)		Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINENAME	VARCHAR (128)		Nombre no calificado de la rutina.
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
ORDINAL	SMALLINT		Posición del parámetro dentro de la signatura de la rutina.
OPTION	VARCHAR (128)		Nombre de la opción de rutina federada.
SETTING	VARCHAR (2048)		Valor de la opción de rutina federada.

SYSCAT.ROUTINEPARMS

Cada fila representa un parámetro o el resultado de una rutina definida en SYSCAT.ROUTINES.

Tabla 152. Vista de catálogo SYSCAT.ROUTINEPARMS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)	S	Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece la rutina. El valor es nulo si no es una rutina de módulo.
ROUTINENAME	VARCHAR (128)	S	Nombre no calificado de la rutina.
ROUTINEMODULEID	INTEGER	S	Identificador para el módulo al que pertenece la rutina. El valor es nulo si no es una rutina de módulo.
SPECIFICNAME	VARCHAR (128)	S	Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
PARMNAME	VARCHAR (128)	S	Nombre del parámetro o columna de resultado, o valor nulo si no existe ningún nombre.
ROWTYPE	CHAR (1)		• B = Tanto parámetro de entrada como de salida • C = Resultado después de conversión • O = Parámetro de salida • P = Parámetro de entrada • R = Resultado antes de conversión
ORDINAL	SMALLINT		Si ROWTYPE = 'B', 'O' o 'P', posición numérica del parámetro dentro de la signatura de la rutina, empezando por 1; si ROWTYPE = 'R' y la rutina devuelve una tabla, posición numérica de una columna nombrada en la tabla de resultados, empezando por 1; 0 en caso contrario.
TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de datos si TYPEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece el tipo de datos.
TYPEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos del parámetro o resultado. El valor es nulo si no es un tipo de datos de módulo.
TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo de datos.
LOCATOR	CHAR (1)		• N = El parámetro o resultado no se pasa en forma de localizador • Y = El parámetro o resultado se pasa en forma de localizador

Tabla 152. Vista de catálogo SYSCAT.ROUTINEPARMS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
LENGTH[1]	INTEGER		Longitud del parámetro o resultado; 0 si el parámetro o resultado es un tipo de datos definido por el usuario.
SCALE[1]	SMALLINT		Escala si el tipo de datos del parámetro o resultado es DECIMAL; el número de dígitos de segundos fraccionarios si el tipo de datos del parámetro o resultado es TIMESTAMP; 0 en caso contrario.
CODEPAGE	SMALLINT		Página de códigos de este parámetro o resultado; o indica no aplicable o un parámetro o resultado para datos de caracteres declarado con el atributo FOR BIT DATA.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para el parámetro; valor nulo en caso contrario.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para el parámetro; valor nulo en caso contrario.
CAST_FUNCSCHEMA	VARCHAR (128)	S	Nombre de esquema de la función utilizada para convertir un argumento o un resultado. Se aplica a funciones con fuente y externas; valor nulo en caso contrario.
CAST_FUNCSPECIFIC	VARCHAR (128)	S	Nombre no calificado de la función utilizada para convertir un argumento o resultado. Se aplica a funciones con fuente y externas; valor nulo en caso contrario.
TARGET_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de destino si el tipo del parámetro o resultado es REFERENCE. Valor nulo si el tipo del parámetro o resultado no es REFERENCE.
TARGET_TYPEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el tipo de destino si el tipo del parámetro o resultado es REFERENCE. El valor es nulo si el tipo de parámetro o resultado no es REFERENCE o si el tipo de destino no es un tipo de datos de módulo.
TARGET_TYPENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el tipo de destino si el tipo del parámetro o resultado es REFERENCE. El valor es nulo si el tipo de parámetro o resultado no es REFERENCE o si el tipo de destino no es un tipo de datos de módulo.
SCOPE_TABSCHEMA	VARCHAR (128)	S	Nombre de esquema del ámbito (valor de destino), si el tipo del parámetro es REFERENCE; valor nulo en caso contrario.
SCOPE_TABNAME	VARCHAR (128)	S	Nombre no calificado del ámbito (valor de destino), si el tipo del parámetro es REFERENCE; valor nulo en caso contrario.

Tabla 152. Vista de catálogo SYSCAT.ROUTINEPARMS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TRANSFORMGRPNAME	VARCHAR (128)	S	Nombre del grupo de transformación correspondiente a un parámetro o resultado de tipo estructurado.
DEFAULT	CLOB (64K)	S	Expresión utilizada para calcular el valor por omisión del parámetro. El valor es nulo si no se especificó la cláusula DEFAULT para el parámetro.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. LENGTH y SCALE se establecen en 0 para las funciones con fuente (funciones definidas con una referencia a otra función), porque heredan la longitud y escala de los parámetros de su fuente.

SYSCAT.ROUTINES

Cada fila representa una rutina definida por el usuario (función escalar, función de tabla, función fuente, método o procedimiento). No incluye las funciones incorporadas.

Tabla 153. Vista de catálogo SYSCAT.ROUTINES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)		Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINEMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece la rutina. El valor es nulo si no es una rutina de módulo.
ROUTINENAME	VARCHAR (128)		Nombre no calificado de la rutina.
ROUTINETYPE	CHAR (1)		Tipo de rutina. • F = Función • M = Método • P = Procedimiento
OWNER	VARCHAR (128)		ID de autorización del propietario de la rutina.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
ROUTINEID	INTEGER		Identificador de la rutina.
ROUTINEMODULEID	INTEGER	S	Identificador para el módulo al que pertenece la rutina. El valor es nulo si no es una rutina de módulo.
RETURN_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de retorno correspondiente a una función escalar o a un método.
RETURN_TYPEMODULE	VARCHAR (128)	S	Nombre de módulo del tipo de retorno; el valor es nulo si el tipo de retorno no pertenece a ningún módulo.
RETURN_TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo de retorno correspondiente a una función escalar o a un método.
ORIGIN	CHAR (1)		• B = Incorporado • E = Definido por el usuario, externo • M = Función de plantilla • F = Procedimiento federado • Q = Con cuerpo de SQL[1] • R = Rutina incorporada al SQL generada por el sistema • S = Generado por el sistema • T = Función de transformación generada por el sistema (no se puede invocar directamente) • U = Definido por el usuario, basado en una fuente

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
FUNCTIONTYPE	CHAR (1)		• C = Columna o agregado • R = Fila • S = Escalar • T = Tabla • Blanco = Procedimiento
PARM_COUNT	SMALLINT		Número de parámetros de la rutina.
LANGUAGE	CHAR(8)		Lenguaje de implantación del cuerpo de la rutina (o del cuerpo de la función fuente, si esta función es la fuente de otra función). Los valores posibles son 'C', 'COBOL', 'JAVA', 'OLE', 'OLEDB' o 'SQL'. Espacios en blanco si ORIGIN no es 'E', 'Q' ni 'R'.
DIALECT	VARCHAR(10)		Dialecto fuente del cuerpo de la rutina de SQL: • DB2 SQL PL • PL/SQL • Blanco = No una rutina de SQL
SOURCESCHEMA	VARCHAR (128)	S	Si ORIGIN = 'U' y la función fuente es una función definida por el usuario, contiene el nombre de esquema del nombre específico de la función fuente. Si ORIGIN = 'U' y la función fuente es una función incorporada, contiene el valor 'SYSIBM'. El valor es nulo si ORIGIN no es 'U'.
SOURCESPECIFIC	VARCHAR (128)	S	Si ORIGIN = 'U' y la función fuente es una función definida por el usuario, contiene el nombre específico no calificado de la función fuente. Si ORIGIN = 'U' y la función fuente es una función incorporada, contiene el valor 'N/A para valor incorporado'. El valor es nulo si ORIGIN no es 'U'.
PUBLISHED	CHAR (1)		Indica si la rutina de módulo puede invocarse fuera de su módulo. • N = La rutina de módulo no está publicada • Y = La rutina de módulo está publicada • Blanco = No se aplica
DETERMINISTIC	CHAR (1)		• N = Los resultados no son determinantes (algunos parámetros pueden tener distintos resultados si llama otra rutina) • Y = Los resultados son determinantes • Blanco = ORIGIN no es 'E', 'F', 'Q' ni 'R'
EXTERNAL_ACTION	CHAR (1)		• E = La función tiene efectos adicionales externos (por lo tanto el número de invocaciones es importante) • N = Ningún efecto adicional • Blanco = ORIGIN no es 'E', 'F', 'Q' ni 'R'
NULLCALL	CHAR (1)		• N = RETURNS NULL ON NULL INPUT (el resultado de la función es de forma implícita el valor nulo si uno o más operandos son nulos) • Y = CALLED ON NULL INPUT • Blanco = ORIGIN no es 'E', 'Q' ni 'R'
CAST_FUNCTION	CHAR (1)		• N = No es una función de conversión • Y = Función de conversión • Blanco = ROUTINETYPE no es 'F'

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ASSIGN_FUNCTION	CHAR (1)		• N = No es una función de asignación • Y = Función de asignación implícita • Blanco = ROUTINETYPE no es 'F'
SCRATCHPAD	CHAR (1)		• N = La rutina no tiene área reutilizable • Y = La rutina tiene un área reutilizable • Blanco = ORIGIN no es 'E' o ROUTINETYPE es 'P'
SCRATCHPAD_LENGTH	SMALLINT		Tamaño (en bytes) del área reutilizable de la rutina. • -1 = LANGUAGE es 'OLEDB' y SCRATCHPAD es 'Y' • 0 = SCRATCHPAD no es 'Y'
FINALCALL	CHAR (1)		• N = No se realiza ninguna llamada final • Y = Se realiza una llamada final a esta rutina al ejecutar el fin de sentencia • Blanco = ORIGIN no es 'E' o ROUTINETYPE es 'P'
PARALLEL	CHAR (1)		• N = La rutina no se puede ejecutar en paralelo • Y = La rutina se puede ejecutar en paralelo • Blanco = ORIGIN no es 'E'
PARAMETER_STYLE	CHAR(8)		Estilo de parámetro que se ha declarado al crearse la rutina. Los valores posibles son: • DB2DARI • DB2GENRL • DB2SQL • GENERAL • GNRLNULL • JAVA • SQL • Blanco si ORIGIN no es 'E'
FENCED	CHAR (1)		• N = No delimitado • Y = Delimitado • Blanco = ORIGIN no es 'E'
SQL_DATA_ACCESS	CHAR (1)		Indica qué tipo de sentencias de SQL, si las hay, debe dar por supuesto el gestor de bases de datos que están contenidas en la rutina. • C = Contiene SQL (sólo expresiones simples sin subconsultas) • M = Contiene sentencias de SQL que modifican datos • N = No contiene sentencias de SQL • R = Contiene sentencias de SQL de sólo lectura • Blanco = ORIGIN no es 'E', 'F', 'Q' ni 'R'
DBINFO	CHAR (1)		Indica si se pasa un parámetro DBINFO a una rutina externa. • N = No se pasa DBINFO • Y = Se pasa DBINFO • Blanco = ORIGIN no es 'E'

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PROGRAMTYPE	CHAR (1)		Indica cómo se ha invocado la rutina externa. • M = Principal • S = Subrutina • Blanco = ORIGIN es 'F'
COMMIT_ON_RETURN	CHAR (1)		Indica se la transacción se confirma tras un retorno satisfactorio por parte de este procedimiento. • N = La unidad de trabajo no se confirma • Y = La unidad de trabajo se confirma • Blanco = ROUTINETYPE no es 'P'
AUTONOMOUS	CHAR (1)		Indica si la rutina se ejecuta de forma autónoma o no. • N = Rutina que no se ejecuta de forma autónoma respecto a la transacción que realiza la invocación • Y = Rutina que se ejecuta de forma autónoma respecto a la transacción que realiza la invocación • Blanco = ROUTINETYPE no es 'P'
RESULT_SETS	SMALLINT		Número máximo estimado de conjuntos de resultados.
SPEC_REG	CHAR (1)		Indica los valores de los registros especiales que se utilizan cuando se llama a la rutina. • I = Registros especiales heredados • Blanco = PARAMETER_STYLE es 'DB2DARI' u ORIGIN no es 'E', 'Q' ni 'R'
FEDERATED	CHAR (1)		Indica si se puede acceder o no a los objetos federados desde la rutina. • Y = Se puede acceder a los objetos federados • Blanco = ORIGIN no es 'F'
THREADSAFE	CHAR (1)		Indica si la rutina se puede ejecutar o no en el mismo proceso que otras rutinas. • N = La rutina no está protegida de hebras • Y = La rutina está protegida de hebras • Blanco = ORIGIN no es 'E'
VALID	CHAR (1)		Se aplica a LANGUAGE = 'SQL' y a las rutinas que tienen parámetros con valores por omisión; blanco en caso contrario. • N = La rutina que tiene que volver a vincular • X = La rutina no está operativa y se tiene que volver a crear • Y = La rutina es válida
MODULEROUTINEIMPLEMENTED	CHAR (1)		• N = El cuerpo de la rutina del módulo no se ha implantado • Y = El cuerpo de la rutina del módulo se ha implantado • Blanco = ROUTINEMODULENAME tiene un valor nulo
METHODIMPLEMENTED	CHAR (1)		• N = El cuerpo del método no se ha implantado • Y = El cuerpo del método se ha implantado • Blanco = ROUTINETYPE no es 'M' o ROUTINEMODULENAME no es un valor nulo

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
METHODEFFECT	CHAR (2)		• CN = Método constructor • MU = Método mutador • OB = Método observador • Blancos = No es un método del sistema
TYPE_PRESERVING	CHAR (1)		• N = El tipo de retorno es el tipo de retorno declarado del método • Y = El tipo de retorno se controla mediante un parámetro de "mantenimiento-tipo"; todos los métodos del mutador generados por el sistema son de mantenimiento del tipo • Blanco = ROUTINETYPE no es 'M'
WITH_FUNC_ACCESS	CHAR (1)		• N = Este método no se puede invocar utilizando una notación funcional • Y = Este método se puede invocar utilizando una notación funcional; es decir, se especifica el atributo "WITH FUNCTION ACCESS" • Blanco = ROUTINETYPE no es 'M'
OVERRIDDEN_METHODID	INTEGER	S	Identificador del método alterado temporalmente cuando se especifica la opción OVERRIDING para un método definido por el usuario. El valor es nulo si ROUTINETYPE no es 'M'.
SUBJECT_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de sujeto para el método definido por el usuario. El valor es nulo si ROUTINETYPE no es 'M'.
SUBJECT_TYPENAME	VARCHAR (128)	S	Nombre no calificado del tipo de sujeto para el método definido por el usuario. El valor es nulo si ROUTINETYPE no es 'M'.
CLASS	VARCHAR (384)	S	Para LANGUAGE JAVA, CLR o OLE, ésta es la clase que implementa esta rutina; valor nulo en caso contrario.
JAR_ID	VARCHAR (128)	S	Para LANGUAGE JAVA, éste es el JAR_ID del archivo jar instalado que implementa esta rutina si se especificó un archivo jar en el momento de creación de la rutina; valor nulo en caso contrario. Para LANGUAGE CLR, éste es el archivo de ensamblaje que implementa esta rutina.
JARSCHEMA	VARCHAR (128)	S	Para LANGUAGE JAVA, cuando hay un JAR_ID, éste es el nombre del esquema del archivo jar que implementa esta rutina; valor nulo en caso contrario.
JAR_SIGNATURE	VARCHAR (2048)	S	Para LANGUAGE JAVA, es la signatura del método Java especificado de esta rutina. Para LANGUAGE CLR, éste es un campo de referencia para la rutina CLR. Valor nulo en caso contrario.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la rutina.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez la rutina.
FUNC_PATH	CLOB (2K)	S	Vía de acceso de SQL en vigor en el momento en que se definió la rutina. El valor es nulo si LANGUAGE no es 'SQL'.
QUALIFIER	VARCHAR (128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
IOS_PER_INVOC	DOUBLE		Número estimado de entradas/salidas (I/O) por invocación; 0 es el valor por omisión; -1 si no se conoce.
INSTS_PER_INVOC	DOUBLE		Número estimado de instrucciones por invocación; 450 es el valor por omisión; -1 si no se conoce.
IOS_PER_ARGBYTE	DOUBLE		Número estimado de I/O por byte de argumento de entrada; 0 es el valor por omisión; -1 si no se conoce.
INSTS_PER_ARGBYTE	DOUBLE		Número estimado de instrucciones por byte de argumento de entrada; 0 es el valor por omisión; -1 si no se conoce.
PERCENT_ARGBYTES	SMALLINT		Porcentaje medio estimado de bytes de argumento de entrada que leerá la rutina realmente; 100 es el valor por omisión; -1 si no se conoce.
INITIAL_IOS	DOUBLE		Número estimado de I/O realizadas la primera vez que se ha invocado la rutina; 0 es el valor por omisión; -1 si no se conoce.
INITIAL_INSTS	DOUBLE		Número estimado de instrucciones ejecutadas la primera vez que se ha invocado la rutina; 0 es el valor por omisión; -1 si no se conoce.
CARDINALITY	BIGINT		Cardinalidad prevista de una función de tabla; -1 si no se conoce o si la rutina no es una función de tabla.
SELECTIVITY[2]	DOUBLE		Para predicados definidos por el usuario; -1 si no hay ningún predicado definido por el usuario.
RESULT_COLS	SMALLINT		Para una función de tabla (ROUTINETYPE = 'F' y FUNCTIONTYPE = 'T'), contiene el número de columnas de la tabla de resultados; para un procedimiento (ROUTINETYPE = 'P'), contiene 0; en caso contrario, contiene 1.
IMPLEMENTATION	VARCHAR (762)	S	Si ORIGIN ='E', identifica la vía de acceso/módulo/función que implanta esta función. Si ORIGIN = 'U' y la función fuente es incorporada, esta columna contiene el nombre y signatura de la función fuente. Valor nulo en caso contrario.
LIB_ID	INTEGER	S	Reservado para una utilización futura.
TEXT_BODY_OFFSET	INTEGER		Si LANGUAGE = 'SQL', el desplazamiento respecto al inicio del cuerpo de la rutina de SQL compilada en el texto completo de la sentencia CREATE; -1 si LANGUAGE no es 'SQL' o la rutina de SQL es en línea.
TEXT	CLOB(2M)	S	Si LANGUAGE = 'SQL', el texto completo de la sentencia CREATE FUNCTION, CREATE METHOD o CREATE PROCEDURE; valor nulo en caso contrario.
NEWSAVEPOINTLEVEL	CHAR (1)		Indica si la rutina inicia un nivel de punto de salvaguarda nuevo cuando se invoca. • N = No se inicia un nivel de punto de salvaguarda nuevo cuando se invoca la rutina; la rutina utiliza el nivel de punto de salvaguarda existente • Y = Se inicia un nivel de punto de salvaguarda nuevo cuando se invoca la rutina • Blanco = No se aplica

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DEBUG_MODE[3]	VARCHAR(8)		Indica si la rutina se puede depurar o no, utilizando el depurador de DB2. • DISALLOW = La rutina no se puede depurar • ALLOW = La rutina se puede depurar y puede participar en una sesión de depuración de cliente con el depurador de DB2 • DISABLE = La rutina no se puede depurar y este valor no se puede modificar sin eliminar y volver a crear la rutina • Blanco = El depurador de DB2 no soporta actualmente el tipo de rutina
TRACE_LEVEL	VARCHAR(1)	S	Reservado para una utilización futura.
DIAGNOSTIC_LEVEL	VARCHAR(1)	S	Reservado para una utilización futura.
CHECKOUT_USERID	VARCHAR (128)	S	ID del usuario que ha realizando una extracción del objeto; el valor es nulo si el objeto no se ha extraído.
PRECOMPILE_OPTIONS	VARCHAR(1024)	S	Opciones de precompilación y vinculación que estaban en vigor cuando se creó la rutina de SQL compilada. El valor nulo si LANGUAGE no es 'SQL' o si la rutina de SQL no está compilada.
COMPILE_OPTIONS	VARCHAR(1024)	S	Valor del registro especial SQL_CCFLAGS que estaba en vigor cuando la rutina de SQL compilada se creó y había presentes directivas de consulta. Serie vacía si no había presentes directivas de consulta en la rutina de SQL compilada. El valor nulo si LANGUAGE no es 'SQL' o si la rutina de SQL no está compilada.
EXECUTION_CONTROL	CHAR (1)		Modalidad de control de ejecución de una rutina de tipo CLR (tiempo de ejecución de lenguaje común) Los valores posibles son: • N = Red • R = Lectura de archivo • S = Seguro • U = No seguro • W = Escritura de archivo • Blanco = LANGUAGE no es 'CLR'
CODEPAGE	SMALLINT		Página de códigos de la rutina, que especifica la página de códigos por omisión utilizada para todos los tipos de parámetros de caracteres, tipos de resultados y variables locales dentro del cuerpo de la rutina.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para la rutina.
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para la rutina.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en la rutina.
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en la rutina.

Tabla 153. Vista de catálogo SYSCAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ENCODING_SCHEME	CHAR (1)		Esquema de codificación de la rutina, según lo especificado en la cláusula PARAMETER CCSID. Los valores posibles son: • A = ASCII • U = UNICODE • Blanco = No se ha especificado la cláusula PARAMETER CCSID
LAST_REGEN_TIME	TIMESTAMP		Hora a la que se ha vuelto a generar por última vez el descriptor empaquetado de la rutina SQL.
INHERITLOCKREQUEST	CHAR (1)		• N = Esta función o método 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 • Y = Esta función o método hereda el nivel de aislamiento de la sentencia que lo ha invocado; también hereda la cláusula de petición de bloqueo especificada • Blanco = LANGUAGE no es 'SQL' o ROUTINETYPE es 'P'
DEFINER[4]	VARCHAR (128)		ID de autorización del propietario de la rutina.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. Para procedimientos de SQL creados antes de la versión 8.2 y actualizados a la versión 9, 'E' (en lugar de 'Q').
2. Durante la actualización de la base de datos, la columna SELECTIVITY se establecerá en -1 en el descriptor empaquetado y en los catálogos del sistema para todas las rutinas definidas por el usuario. Para un predicado definido por el usuario, la selectividad es -1 en el catálogo del sistema. En este caso, el valor de selectividad utilizado por el optimizador es 0.01.
3. Para rutinas Java, el valor de DEBUG_MODE no indica si la rutina Java se ha compilado realmente en modalidad de depuración o si se ha instalado un archivo Jar de depuración en el servidor.
4. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.ROUTINESFEDERATED

Cada fila representa un procedimiento federado.

Tabla 154. Vista de catálogo SYSCAT.ROUTINESFEDERATED

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ROUTINESCHEMA	VARCHAR (128)		Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINENAME	VARCHAR (128)		Nombre no calificado de la rutina.
ROUTINETYPE	CHAR (1)		Tipo de rutina. • P = Procedimiento
OWNER	VARCHAR (128)		ID de autorización del propietario de la rutina.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
ROUTINEID	INTEGER		Identificador de la rutina.
PARM_COUNT	SMALLINT		Número de parámetros de la rutina.
DETERMINISTIC	CHAR (1)		• N = Los resultados no son determinantes (algunos parámetros pueden tener distintos resultados si llama otra rutina) • Y = Los resultados son determinantes
EXTERNAL_ACTION	CHAR (1)		• E = La rutina tiene efectos adicionales externos (por lo tanto el número de invocaciones es importante) • N = Ningún efecto adicional
SQL_DATA_ACCESS	CHAR (1)		Indica qué tipo de sentencias de SQL, si las hay, debe dar por supuesto el gestor de bases de datos que están contenidas en la rutina. • C = Contiene SQL (sólo expresiones simples sin subconsultas) • M = Contiene sentencias de SQL que modifican datos • N = No contiene sentencias de SQL • R = Contiene sentencias de SQL de sólo lectura
COMMIT_ON_RETURN	CHAR (1)		Indica se la transacción se confirma tras un retorno satisfactorio por parte de este procedimiento. • N = La unidad de trabajo no se confirma • Y = La unidad de trabajo se confirma • Blanco = ROUTINETYPE no es 'P'
RESULT_SETS	SMALLINT		Número máximo estimado de conjuntos de resultados.

Tabla 154. Vista de catálogo SYSCAT.ROUTINESFEDERATED (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la rutina.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez la rutina.
QUALIFIER	VARCHAR (128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.
RESULT_COLS	SMALLINT		Para un procedimiento (ROUTINETYPE = 'P'), contiene 0; en caso contrario, contiene 1.
CODEPAGE	SMALLINT		Página de códigos de la rutina, que especifica la página de códigos por omisión utilizada para todos los tipos de parámetros de caracteres, tipos de resultados y variables locales dentro del cuerpo de la rutina.
LAST_REGEN_TIME	TIMESTAMP		Hora a la que se ha vuelto a generar por última vez el descriptor empaquetado de la rutina SQL.
REMOTE_PROCEDURE	VARCHAR (128)	S	Nombre no calificado del procedimiento fuente para el que se ha creado la rutina federada.
REMOTE_SCHEMA	VARCHAR (128)	S	Nombre del esquema del procedimiento fuente para el que se ha creado la rutina federada.
SERVERNAME	VARCHAR (128)	S	Nombre de la fuente de datos que contiene el procedimiento fuente para el cual se ha creado la rutina federada.
REMOTE_PACKAGE	VARCHAR (128)	S	Nombre del paquete al que pertenece el procedimiento fuente (se aplica únicamente a los derivadores para fuentes de datos Oracle).
REMOTE_PROCEDURE_ ALTER_TIME	VARCHAR (128)	S	Reservado para una utilización futura.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.ROWFIELDS

Cada fila representa un campo que está definido para un tipo de datos de fila definido por el usuario.

Tabla 155. Vista de catálogo SYSCAT.ROWFIELDS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos de fila que incluye el campo.
TYPEMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos de fila. El valor es nulo si no es un tipo de datos de fila de módulo.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos de fila que incluye el campo.
FIELDNAME	VARCHAR (128)		Nombre de campo.
FIELDTYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos del campo.
FIELDTYPEMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el tipo de datos del campo. El valor es nulo si el tipo de datos del campo no es un tipo de datos de módulo definido por el usuario.
FIELDTYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos del campo.
ORDINAL	SMALLINT		Posición del campo en la definición del tipo de datos de fila, empezando por 0.
LENGTH	INTEGER		Longitud del tipo de datos de campo. Para los tipos decimales, contiene la precisión.
SCALE	SMALLINT		Para los tipos decimales, contiene la escala del tipo de datos de campo; para valores de indicación de fecha y hora, contiene la precisión de indicación de fecha y hora del tipo de datos de campo; 0 en caso contrario.
CODEPAGE	SMALLINT		Para tipos de serie, indica la página de códigos; 0 indica FOR BIT DATA; 0 para tipos que no sean de serie.
COLLATIONSCHEMA	VARCHAR (128)	S	Para tipos de serie, el nombre de esquema de la clasificación para el campo; un valor nulo en caso contrario.
COLLATIONNAME	VARCHAR (128)	S	Para tipos de serie, el nombre no calificado de la clasificación para el campo; de lo contrario, valor nulo.

SYSCAT.SCHEMAAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado uno o más privilegios sobre un esquema.

Tabla 156. Vista de catálogo SYSCAT.SCHEMAAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado un privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene un privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SCHEMANAME	VARCHAR (128)		Nombre del esquema al que se aplica este privilegio.
ALTERINAUTH	CHAR (1)		Privilegio para modificar o realizar comentarios sobre los objetos del esquema nombrado. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
CREATEINAUTH	CHAR (1)		Privilegio para crear objetos en el esquema nombrado. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
DROPINAUTH	CHAR (1)		Privilegio para eliminar objetos del esquema nombrado. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.SCHEMATA

Cada fila representa un esquema.

Tabla 157. Vista de catálogo SYSCAT.SCHEMATA

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SCHEMANAME	VARCHAR (128)		Nombre del esquema.
OWNER	VARCHAR (128)		ID de autorización del propietario del esquema.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
DEFINER	VARCHAR (128)		ID de autorización del definidor del esquema o ID de autorización del propietario del esquema si se ha transferido la propiedad del mismo.
DEFINERTYPE	CHAR (1)		• S = El sistema es el definidor • U = El definidor es un usuario individual
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el esquema.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SECURITYLABELACCESS

Cada fila representa la etiqueta de seguridad que se ha otorgado al ID de autorización de base de datos.

Tabla 158. Vista de catálogo SYSCAT.SECURITYLABELACCESS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que otorga la etiqueta de seguridad.
GRANTEE	VARCHAR (128)		El que mantiene la etiqueta de seguridad.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SECLABELID	INTEGER		Identificador de la etiqueta de seguridad. Como nombre de la etiqueta de seguridad, seleccione la columna SECLABELNAME del valor correspondiente de SECLABELID en la vista de catálogo SYSCAT.SECURITYLABELS.
SECPOLICYID	INTEGER		Identificador de la política de seguridad asociada a la etiqueta de seguridad. Como nombre de la política de seguridad, seleccione la columna SECPOLICYNAME del valor correspondiente de SECPOLICYID en la vista de catálogo SYSCAT.SECURITYPOLICIES.
ACCESSTYPE	CHAR (1)		• B = Tanto acceso de lectura como de grabación • R = Acceso de lectura • W = Acceso de grabación
GRANT_TIME	TIMESTAMP		Hora a la que se ha otorgado la etiqueta de seguridad.

SYSCAT.SECURITYLABELCOMPONENTELEMENTS

Cada fila representa un valor de elemento de un componente de etiqueta de seguridad.

Tabla 159. Vista de catálogo SYSCAT.SECURITYLABELCOMPONENTELEMENTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COMPID	INTEGER		Identificador del componente de etiqueta de seguridad.
ELEMENTVALUE	VARCHAR (32)		Valor de elemento del componente de etiqueta de seguridad.
ELEMENTVALUEENCODING	CHAR(8) FOR BIT DATA		Formato codificado del valor del elemento.
PARENTELEMENTVALUE	VARCHAR (32)	S	Nombre del padre de un elemento para componentes de árbol; el valor es nulo para los componentes de conjunto y de matriz y para el nodo ROOT de un componente de árbol.

SYSCAT.SECURITYLABELCOMPONENTS

Cada fila representa un componente de etiqueta de seguridad.

Tabla 160. Vista de catálogo SYSCAT.SECURITYLABELCOMPONENTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COMPNAME	VARCHAR (128)		Nombre del componente de etiqueta de seguridad.
COMPID	INTEGER		Identificador del componente de etiqueta de seguridad.
COMPTYPE	CHAR (1)		Tipo de componente de etiqueta de seguridad. • A = Matriz • S = Conjunto • T = Árbol
NUMELEMENTS	INTEGER		Número de elementos del componente de etiqueta de seguridad.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el componente de etiqueta de seguridad.
REMARKS	VARCHAR (254)		Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SECURITYLABELS

Cada fila representa una etiqueta de seguridad.

Tabla 161. Vista de catálogo SYSCAT.SECURITYLABELS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SECLABELNAME	VARCHAR (128)		Nombre de la etiqueta de seguridad.
SECLABELID	INTEGER		Identificador de la etiqueta de seguridad.
SECPOLICYID	INTEGER		Identificador de la política de seguridad a la que pertenece la etiqueta de seguridad.
SECLABEL	SYSPROC. DB2SECURITYLABEL		Representación interna de la etiqueta de seguridad.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la etiqueta de seguridad.
REMARKS	VARCHAR (254)	Y	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SECURITYPOLICIES

Cada fila representa una política de seguridad.

Tabla 162. Vista de catálogo SYSCAT.SECURITYPOLICIES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SECPOLICYNAME	VARCHAR (128)		Nombre de la política de seguridad.
SECPOLICYID	INTEGER		Identificador de la política de seguridad.
NUMSECLABELCOMP	INTEGER		Número de componentes de etiqueta de seguridad de la política de seguridad.
RWSECLABELREL	CHAR (1)		Relación entre las etiquetas de seguridad correspondientes al acceso de lectura y de grabación otorgado al mismo ID de autorización. • S = La etiqueta de seguridad correspondiente al acceso de grabación otorgado a un usuario es un subconjunto de la etiqueta de seguridad correspondiente al acceso de lectura otorgado al mismo usuario
NOTAUTHWRITESECLABEL	CHAR (1)		Acción que se debe emprender cuando un usuario no tiene autorización para escribir la etiqueta de seguridad especificada en la sentencia INSERT o UPDATE. • O = Alterar temporalmente • R = Restringir
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la política de seguridad.
GROUPAUTHS	CHAR (1)		Indica si se utilizarán o ignorarán las autorizaciones de etiquetas de seguridad y exenciones otorgadas a un ID de autorización que representa a un grupo. • I = Ignorada • U = Utilizada
ROLEAUTHS	CHAR (1)		Indica si se utilizarán o ignorarán las autorizaciones de etiquetas de seguridad y exenciones otorgadas a un ID de autorización que representa a un rol. • I = Ignorada • U = Utilizada
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SECURITYPOLICYCOMPONENTRULES

Cada fila representa las normas de acceso de lectura y grabación correspondientes a un componente de etiqueta de seguridad de la política de seguridad.

Tabla 163. Vista de catálogo SYSCAT.SECURITYPOLICYCOMPONENTRULES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SECPOLICYID	INTEGER		Identificador de la política de seguridad.
COMPID	INTEGER		Identificador del componente de etiqueta de seguridad de la política de seguridad.
ORDINAL	INTEGER		Posición del componente de etiqueta de seguridad tal como aparece en la política de seguridad, empezando por 1.
READACCESSRULENAME	VARCHAR (128)		Nombre de la norma de acceso de lectura asociada al componente de etiqueta de seguridad.
READACCESSRULETEXT	VARCHAR (512)		Texto de la norma de acceso de lectura asociada al componente de etiqueta de seguridad.
WRITEACCESSRULENAME	VARCHAR (128)		Nombre de la norma de acceso de grabación asociada al componente de etiqueta de seguridad.
WRITEACCESSRULETEXT	VARCHAR (512)		Texto de la norma de acceso de grabación asociada al componente de etiqueta de seguridad.

SYSCAT.SECURITYPOLICYEXEMPTIONS

Cada fila representa una exención de la política de seguridad que se ha otorgado a un ID de autorización de base de datos.

Tabla 164. Vista de catálogo SYSCAT.SECURITYPOLICYEXEMPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que otorga la exención.
GRANTEE	VARCHAR (128)		El que mantiene la exención.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SECPOLICYID	INTEGER		Identificador de la política de seguridad a la que se ha otorgado la exención. Como nombre de la política de seguridad, seleccione la columna SECPOLICYNAME del valor correspondiente de SECPOLICYID en la vista de catálogo SYSCAT.SECURITYPOLICIES.
ACCESSRULENAME	VARCHAR (128)		Nombre de la norma de acceso para la que se ha otorgado la exención.
ACCESSTYPE	CHAR (1)		Tipo de acceso al que se aplica la norma. • R = Acceso de lectura • W = Acceso de grabación
ORDINAL	INTEGER		Posición del componente de etiqueta de seguridad en la política de seguridad a la que se aplica la norma.
ACTIONALLOWED	CHAR (1)		Si la norma es DB2LBACWRITEARRAY, entonces: • D = Escritura hacia abajo • U = Escritura hacia arriba Blanco en caso contrario.
GRANT_TIME	TIMESTAMP		Hora a la que se ha otorgado la exención.

SYSCAT.SEQUENCEAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado uno o más privilegios sobre una secuencia.

Tabla 165. Vista de catálogo SYSCAT.SEQUENCEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado un privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene un privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
SEQSCHEMA	VARCHAR (128)		Nombre de esquema de la secuencia.
SEQNAME	VARCHAR (128)		Nombre no calificado de la secuencia.
ALTERAUTH	CHAR (1)		Privilegio para modificar la secuencia. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
USAGEAUTH	CHAR (1)		Privilegio para hacer referencia a la secuencia. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.SEQUENCES

Cada fila representa una secuencia o alias.

Tabla 166. Vista de catálogo SYSCAT.SEQUENCES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SEQSCHEMA	VARCHAR (128)		Nombre de esquema de la secuencia.
SEQNAME	VARCHAR (128)		Nombre no calificado de la secuencia.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la secuencia.
DEFINERTYPE	CHAR (1)		• S = El sistema es el definidor • U = El definidor es un usuario individual
OWNER	VARCHAR (128)		ID de autorización del propietario de la secuencia.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
SEQID	INTEGER		Identificador de la secuencia o el alias.
SEQTYPE	CHAR (1)		Tipo de secuencia. • A = Alias • I = Secuencia de identificación • S = Secuencia
BASE_SEQSCHEMA	VARCHAR (128)	S	Si SEQTYPE es 'A', contiene el nombre de esquema de la secuencia o el alias al que hace referencia este alias; el valor es nulo en caso contrario.
BASE_SEQNAME	VARCHAR (128)	S	Si SEQTYPE es 'A', contiene el nombre no calificado de la secuencia o el alias al que hace referencia este alias; el valor es nulo en caso contrario.
INCREMENT	DECIMAL(31,0)	S	Valor de incremento. El valor es nulo si la secuencia es un alias.
START	DECIMAL(31,0)	S	Valor inicial de la secuencia. El valor es nulo si la secuencia es un alias.
MAXVALUE	DECIMAL(31,0)	S	Valor máximo de la secuencia. El valor es nulo si la secuencia es un alias.
MINVALUE	DECIMAL(31,0)	S	Valor mínimo de la secuencia.El valor es nulo si la secuencia es un alias.
NEXTCACHEFIRSTVALUE	DECIMAL(31,0)	S	Primer valor que estará disponible para asignarlo en el bloque de memoria intermedia siguiente. Si no hay almacenamiento en antememoria, el siguiente valor que estará disponible para asignar.

Tabla 166. Vista de catálogo SYSCAT.SEQUENCES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CYCLE	CHAR (1)		Indica si la secuencia puede o no continuar generando valores después de alcanzar su valor máximo o mínimo. • N = La secuencia no puede continuar • Y = La secuencia puede continuar • Blanco = La secuencia es un alias.
CACHE	INTEGER		Número de valores de secuencia que se deben preasignar en memoria para el acceso más rápido. 0 indica que los valores de la secuencia no se deben preasignar. En una base de datos particionada, este valor se aplica a cada partición de base de datos. -1 si la secuencia es un alias.
ORDER	CHAR (1)		Indica si los números de secuencia se deben o no generar en orden de petición. • N = No es necesario que los números de secuencia se generen en orden de petición • Y = Los números de secuencia se deben generar en orden de petición • Blanco = La secuencia es un alias.
DATATYPEID	INTEGER		Para tipos incorporados, el identificador interno del tipo incorporado. Para tipos diferenciados, el identificador interno del tipo diferenciado. 0 si la secuencia es un alias.
SOURCETYPEID	INTEGER		Para un tipo incorporado o si la secuencia es un alias, tiene un valor de 0. Para un tipo diferenciado, es el identificador interno del tipo incorporado que es el tipo fuente del tipo diferenciado.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la secuencia.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez la secuencia.
PRECISION	SMALLINT		Precisión del tipo de datos de la secuencia. Los valores posibles son: • 5 = SMALLINT • 10 = INTEGER • 19 = BIGINT Para DECIMAL, es la precisión del tipo de datos DECIMAL especificado.0 si la secuencia es un alias.
ORIGIN	CHAR (1)		Origen de la secuencia. • S = Secuencia generada por el sistema • U = Secuencia generada por el usuario
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.SERVEROPTIONS

Cada fila representa un valor de opción específico del servidor.

Tabla 167. Vista de catálogo SYSCAT.SERVEROPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRAPNAME	VARCHAR (128)	S	Nombre del derivador.
SERVERNAME	VARCHAR (128)	S	Nombre en mayúsculas del servidor.
SERVERTYPE	VARCHAR (30)	S	Tipo de servidor.
SERVERVERSION	VARCHAR (18)	S	Versión de servidor.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la entrada.
OPTION	VARCHAR (128)		Nombre de la opción de servidor.
SETTING	VARCHAR (2048)		Valor de la opción de servidor.
SERVEROPTIONKEY	VARCHAR (18)		Identifica una fila de forma exclusiva.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SERVERS

Cada fila representa una fuente de datos.

Tabla 168. Vista de catálogo SYSCAT.SERVERS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRAPNAME	VARCHAR (128)		Nombre del derivador.
SERVERNAME	VARCHAR (128)		Nombre en mayúsculas del servidor.
SERVERTYPE	VARCHAR (30)	S	Tipo de servidor.
SERVERVERSION	VARCHAR (18)	S	Versión de servidor.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.SERVICECLASSES

Cada fila representa una clase de servicio.

Tabla 169. Vista de catálogo SYSCAT.SERVICECLASSES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
SERVICECLASSNAME	VARCHAR (128)		Nombre de la clase de servicio.
PARENTSERVICECLASSNAME	VARCHAR (128)	S	Nombre de clase de servicio de la superclase de servicio padre.
SERVICECLASSID	SMALLINT		Identificador de la clase de servicio.
PARENTID	SMALLINT		Identificador de la clase de servicio padre para esta clase de servicio. 0 si esta clase de servicio es una super clase de servicio.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado la clase de servicio.
ALTER_TIME	TIMESTAMP		Hora en que se modificó por última vez la clase de servicio.
ENABLED	CHAR (1)		Estado de la clase de servicio. • N = Inhabilitado • Y = Habilitada
AGENTPRIORITY	SMALLINT		Prioridad de hebra de los agentes de la clase de servicio relacionados con la prioridad normal de las hebras de DB2. • De -20 a 20 (Linux y UNIX) • De -6 a 6 (Windows) • -32768 = no establecido
PREFETCHPRIORITY	CHAR (1)		Prioridad de los agentes en la clase de servicio. • H = Alto • L = Bajo • M = Medio • Blanco = no establecido
BUFFERPOOLPRIORITY	CHAR (1)		Prioridad de agrupación de almacenamientos intermedios de los agentes en la clase de servicio. • H = Alto • L = Bajo • M = Medio • Blanco = No establecido
INBOUNDCORRELATOR	VARCHAR (128)	S	Para su utilización en el futuro.
OUTBOUNDCORRELATOR	VARCHAR (128)	S	Serie utilizada para asociar la clase de servicio a una clase de servicio del gestor de carga de trabajo del sistema operativo.

Tabla 169. Vista de catálogo SYSCAT.SERVICECLASSES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLECTAGGACTDATA	CHAR (1)		Especifica los datos de actividad agregados que deben capturarse para la clase de servicio por medio del supervisor de sucesos aplicable. • B = Recopilar datos de actividad agregados básicos • E = Recopilar datos de actividad agregados ampliados • N = Ninguno
COLLECTAGGREQDATA	CHAR (1)		Especifica los datos de actividad agregados que deben capturarse para la clase de servicio por medio del supervisor de sucesos aplicable. • B = Recopilar datos de petición agregados básicos • N = Ninguno
COLLECTACTDATA	CHAR (1)		Especifica los datos de actividad que deben recopilarse por medio del supervisor de sucesos aplicable. • D = Datos de actividad con detalles • N = Ninguno • S = Datos de actividad con detalles y entorno de sección • V = Datos de actividad con detalles y valores • W = Datos de actividad sin detalles • X = Datos de actividad con detalles, entorno de sección y valores
COLLECTACTPARTITION	CHAR (1)		Especifica el lugar en el que se recopilarán los datos de actividad. • C = Partición de base de datos del coordinador de la actividad • D = Todas las particiones de base de datos
COLLECTREQMETRICS	CHAR (1)		Especifica el nivel de supervisión de las peticiones enviadas por una conexión que está asociada con la superclase de servicio. • B = Recopilar métricas de peticiones base • E = Recopilar métricas de peticiones ampliadas • N = Ninguno
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.STATEMENTS

Cada fila representa una sentencia de SQL de un paquete.

Tabla 170. Vista de catálogo SYSCAT.STATEMENTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
PKGSCHEMA	VARCHAR (128)		Nombre de esquema del paquete.
PKGNAME	VARCHAR (128)		Nombre no calificado del paquete.
STMTNO	INTEGER		El número de línea de la sentencia de SQL del módulo fuente del programa de aplicación.
SECTNO	SMALLINT		El número de la sección del paquete que contiene la sentencia de SQL.
SEQNO	INTEGER		Siempre 1.
TEXT	CLOB(2M)		Texto de la sentencia de SQL.
UNIQUE_ID	CHAR(8) FOR BIT DATA		Identificador de un paquete específico cuando existen varios paquetes con el mismo nombre.
VERSION	VARCHAR(64)	S	Identificador de versión del paquete.

SYSCAT.SURROGATEAUTHIDS

Cada fila representa un usuario o un grupo al que se ha otorgado el privilegio SETSESSIONUSER sobre un usuario o PUBLIC.

Tabla 171. Vista de catálogo SYSCAT.SURROGATEAUTHIDS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		ID de autorización que ha otorgado a TRUSTEDID la capacidad para actuar como un sustituto. Cuando TRUSTEDID representa un objeto de contexto fiable, este campo representa el ID de autorización que ha creado o modificado el objeto de contexto fiable.
TRUSTEDID	VARCHAR (128)		Identificador de la entidad en la que se confía para que actúe como un sustituto.
TRUSTEDIDTYPE	CHAR (1)		• C = Contexto fiable • G = Grupo • U = Usuario
SURROGATEAUTHID	VARCHAR (128)		ID de autorización del sustituto que puede adoptar TRUSTEDID. 'PUBLIC' indica que TRUSTEDID puede adoptar cualquier ID de autorización.
SURROGATEAUTHIDTYPE	CHAR (1)		• G = Grupo • U = Usuario
AUTHENTICATE	CHAR (1)		• N = No se requiere ninguna autentificación. • Y = Se necesita un símbolo de autentificación con el ID de autorización para autentificar al usuario antes de que pueda asumirse el ID de autorización • Blanco = TRUSTEDIDTYPE no es 'C'
CONTEXTROLE	VARCHAR (128)	S	Ha de asignarse un rol específico para el ID de autorización asumido, que reemplaza el rol por omisión, si lo hay, definida por el contexto fiable. Valor nulo cuando TRUSTEDIDTYPE no es 'C'.
GRANT_TIME	TIMESTAMP		Hora a la que se ha efectuado el otorgamiento.

SYSCAT.TABAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado uno o más privilegios sobre una tabla o vista.

Tabla 172. Vista de catálogo SYSCAT.TABAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o vista.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o vista.
CONTROLAUTH	CHAR (1)		Privilegio CONTROL. • N = No se mantiene • Y = Se mantiene pero no se puede otorgar
ALTERAUTH	CHAR (1)		Privilegio para modificar la tabla; permitir que una tabla padre de esta tabla elimine su restricción de clave primaria o de unidad; permitir que una tabla se convierta en una tabla de consultas materializadas que haga referencia a esta tabla o vista en la consulta materializada; o permitir que una tabla que haga referencia a esta tabla o vista en su consulta materializada deje de ser una tabla de consultas materializadas. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
DELETEAUTH	CHAR (1)		Privilegio para suprimir filas de una tabla o vista actualizable. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
INDEXAUTH	CHAR (1)		Privilegio para crear un índice sobre una tabla. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

Tabla 172. Vista de catálogo SYSCAT.TABAUTH (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
INSERTAUTH	CHAR (1)		Privilegio para insertar en una tabla o vista actualizable o para ejecutar el programa de utilidad de importación sobre una tabla o vista. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
REFAUTH	CHAR (1)		Privilegio para crear y eliminar una clave foránea que hace referencia a una tabla como la tabla padre. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
SELECTAUTH	CHAR (1)		Privilegio para recuperar filas de una tabla o vista, crear vistas en una tabla o ejecutar el programa de utilidad de exportación sobre una tabla o vista. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
UPDATEAUTH	CHAR (1)		Privilegio para ejecutar la sentencia UPDATE sobre una tabla o vista actualizable. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.TABCONST

Cada fila representa una restricción de tabla del tipo CHECK, UNIQUE, PRIMARY KEY o FOREIGN KEY. Para jerarquías de tablas, cada restricción se registra sólo a nivel de la jerarquía en la que se ha creado la restricción.

Tabla 173. Vista de catálogo SYSCAT.TABCONST

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CONSTNAME	VARCHAR (128)		Nombre de la restricción.
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla a la que se aplica esta restricción.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla a la que se aplica esta restricción.
OWNER	VARCHAR (128)		ID de autorización del propietario de la restricción.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TYPE	CHAR (1)		Indica el tipo de restricción. • F = Clave foránea • I = Dependencia funcional • K = Comprobación • P = Clave primaria • U = Unicidad
ENFORCED	CHAR (1)		• N = No imponer la restricción • Y = Imponer la restricción
CHECKEXISTINGDATA	CHAR (1)		• D = Diferir la comprobación de los datos existentes • I = Comprobación inmediata de los datos existentes • N = No comprobar nunca los datos existentes
ENABLEQUERYOPT	CHAR (1)		• N = Optimización de consulta desactivada • Y = Optimización de consulta activada
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la restricción.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.TABDEP

Cada fila representa una dependencia de una vista o de una tabla de consultas materializadas sobre algún otro objeto. La vista o tabla de consultas materializadas depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta a la vista o a la tabla de consultas materializadas. También codifica la forma en que los privilegios sobre vistas dependen de los privilegios sobre las tablas y vistas subyacentes.

Tabla 174. Vista de catálogo SYSCAT.TABDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la vista o tabla de consultas materializadas.
TABNAME	VARCHAR (128)		Nombre no calificado de la vista o tabla de consultas materializadas.
DTYPE	CHAR (1)		Tipo de objeto dependiente. • S = Tabla de consultas materializadas • T = Tabla (sólo de etapas) • V = Vista (sin tipo) • W = Vista con tipo
OWNER	VARCHAR (128)		ID de autorización del creador de la vista o tabla de consultas materializadas.
OWNERTYPE	CHAR (1)		• U = El propietario es un usuario individual
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • F = Rutina • I = Índice si se registra una dependencia de una tabla base • G = Tabla temporal global • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • R = Tipo estructurado definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • Z = Objeto XSR • u = Alias de módulo • v = Variable global
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto del que depende la vista o tabla de consultas materializadas.

Tabla 174. Vista de catálogo SYSCAT.TABDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
BMODULENAME	VARCHAR (128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto del que depende la vista o tabla de consultas materializadas.
BMODULEID	INTEGER	S	Identificador del módulo del objeto del que depende la vista o tabla de consulta materializada.
TABAUTH	SMALLINT	S	Si BTYPE es 'N', 'O', 'S', 'T', 'U', 'V' o 'W', codifica los privilegios de la tabla o vista subyacente de la que depende esta vista o tabla de consultas materializadas; valor nulo en caso contrario.
VARAUTH	SMALLINT	S	Si BTYPE es 'v', codifica los privilegios de la variable global subyacente de la que depende esta vista o tabla de consulta materializada; valor nulo en caso contrario.
DEFINER[1]	VARCHAR (128)		ID de autorización del creador de la vista o tabla de consultas materializadas.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.TABDETACHEDDEP

Cada fila representa una dependencia desenlazada entre una tabla dependiente desenlazada y una tabla desenlazada.

Tabla 175. Vista de catálogo SYSCAT.TABDETACHEDDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla desenlazada.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla desenlazada.
DEPTABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla dependiente desenlazada..
DEPTABNAME	VARCHAR (128)		Nombre no calificado de la tabla dependiente desenlazada.

SYSCAT.TABLES

Cada fila representa una tabla, vista, alias o apodo. Cada jerarquía de tabla o de vista tiene una fila adicional que representa la tabla de jerarquía o la vista de jerarquía que implanta la jerarquía. Se incluyen las tablas de catálogo y las vistas.

Tabla 176. Vista de catálogo SYSCAT.TABLES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema del objeto.
TABNAME	VARCHAR (128)		Nombre no calificado del objeto.
OWNER	VARCHAR (128)		ID de autorización del propietario de la tabla, vista, alias o apodo.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TYPE	CHAR (1)		Tipo de objeto. • A = Alias • G = Tabla temporal creada • H = Tabla de jerarquía • L = Tabla desenlazada • N = Apodo • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo
STATUS	CHAR (1)		Estado del objeto. • C = Pendiente de establecer integridad • N = Normal • X = No operativo
BASE_TABSCHEMA	VARCHAR (128)	S	Si TYPE = 'A', contiene el nombre de esquema de la tabla, vista, alias o apodo al que hace referencia este alias; valor nulo en caso contrario.
BASE_TABNAME	VARCHAR (128)	S	Si TYPE = 'A', contiene el nombre no calificado de la tabla, vista, alias o apodo al que hace referencia este alias; valor nulo en caso contrario.
ROWTYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de fila correspondiente a esta tabla, si se aplica; valor nulo en caso contrario.
ROWTYPENAME	VARCHAR (128)	S	Nombre de no calificado del tipo de fila correspondiente a esta tabla, si se aplica; valor nulo en caso contrario.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el objeto.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez el objeto.
INVALIDATE_TIME	TIMESTAMP		Hora a la que se ha invalidado por última vez el objeto.
STATS_TIME	TIMESTAMP	S	Hora a la que se ha hecho por última vez un cambio en las estadísticas registradas correspondientes a este objeto. El valor es nulo si no se recopilan estadísticas.
COLCOUNT	SMALLINT		Número de columnas, incluidas las columnas heredadas (si las hay).
TABLEID	SMALLINT		Identificador interno del objeto lógico.

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TBSPACEID	SMALLINT		Identificador lógico interno del espacio de tablas primario correspondiente a este objeto.
CARD	BIGINT		Número total de filas de la tabla; -1 si no se recopilan estadísticas.
NPAGES	BIGINT		Número total de páginas en las que existen filas de la tabla; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
FPAGES	BIGINT		Número total de páginas; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
OVERFLOW	BIGINT		Número total de registros de desbordamiento de la tabla; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
TBSPACE	VARCHAR (128)	S	Nombre del espacio de tablas primario correspondiente a la tabla. Si no se especifica ningún otro espacio de tablas, todas las partes de la tabla se almacenan en este espacio de tablas. El valor es nulo para alias, vistas y tablas particionadas.
INDEX_TBSPACE	VARCHAR (128)	S	Nombre del espacio de tablas que contiene todos los índices creados en esta tabla. El valor es nulo para alias, vistas y tablas particionadas o si se ha omitido la cláusula INDEX IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE.
LONG_TBSPACE	VARCHAR (128)	S	Nombre del espacio de tablas que contiene todos los datos largos (tipos de columna LONG o LOB) para esta tabla. El valor es nulo para alias, vistas y tablas particionadas o si se ha omitido la cláusula LONG IN o se ha especificado con el mismo valor que la cláusula IN de la sentencia CREATE TABLE.
PARENTS	SMALLINT	S	Número de tablas padre de este objeto; es decir, el número de restricciones de referencia de las que depende este objeto.
CHILDREN	SMALLINT	S	Número de tablas dependientes de este objeto; es decir, el número de restricciones de referencia de las que depende este objeto.
SELFREFS	SMALLINT	S	Número de restricciones de referencia propia para este objeto; es decir, el número de restricciones de referencia en las que este objeto es tanto padre como dependiente.
KEYCOLUMNS	SMALLINT	S	Número de columnas de la clave primaria.
KEYINDEXID	SMALLINT	S	Identificador de índice correspondiente al índice de clave primaria; 0 o el valor nulo si no hay clave primaria.
KEYUNIQUE	SMALLINT		Número de restricciones de clave de unicidad (que no sean la restricción de clave primara) definidas en este objeto.
CHECKCOUNT	SMALLINT		Número de restricciones de comprobación definidas en este objeto.

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
DATACAPTURE	CHAR (1)		• L = La tabla interviene en la duplicación de datos, incluida la duplicación de las columnas LONG VARCHAR y LONG VARGRAPHIC • N = La tabla no interviene en la duplicación de datos • Y = La tabla interviene en la duplicación de datos, sin incluir la duplicación de las columnas LONG VARCHAR y LONG VARGRAPHIC
CONST_CHECKED	CHAR(32)		• El byte 1 representa restricción de clave foránea. • El byte 2 representa restricción de comprobación. • El byte 5 representa la tabla de consultas materializadas. • El byte 6 representa columna generada. • El byte 7 representa la tabla dispuesta con antelación. • El byte 8 representa restricción de particionamiento de datos. • Otros bytes están reservados para su utilización en el futuro. Los valores posibles son: • F = En el byte 5, la tabla de consultas materializadas no puede renovarse de forma incremental. En el byte 7, el contenido de la tabla dispuesta con antelación es incompleto y no puede utilizarse para renovaciones incrementales de la tabla de consultas materializadas asociada. • N = No comprobado • U = Comprobado por el usuario • W = Estaba en el estado 'U' cuando la tabla se colocó en el estado de pendiente de establecer integridad • Y = Comprobado por el sistema
PMAP_ID	SMALLINT	S	Identificador de la correlación de distribución que utiliza actualmente esta tabla (el valor es nulo para alias o vistas).
PARTITION_MODE	CHAR (1)		Indica la forma en que los datos se distribuyen entre particiones de base de datos en un sistema de bases de datos particionadas. • H = Generación aleatoria • R = Duplicado entre particiones de base de datos • Blanco = No hay particionamiento de base de datos
LOG_ATTRIBUTE	CHAR (1)		• Siempre 0. Esta columna ya no se utiliza.
PCTFREE	SMALLINT		Porcentaje de cada página que se ha de reservar para inserciones posteriores.

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
APPEND_MODE	CHAR (1)		Controla la forma en que se insertan filas en páginas. • N = Las filas nuevas se insertan en los espacios existentes, si los hay • Y = Las filas nuevas se añaden al final de los datos
REFRESH	CHAR (1)		Modalidad de renovación. • D = Diferida • I = Inmediata • O = Una vez • Blanco = No es una tabla de consultas materializadas
REFRESH_TIME	TIMESTAMP	S	Para REFRESH = 'D' y 'O', hora a la que los datos se han renovado por última vez (sentencia REFRESH TABLE); valor nulo en caso contrario.
LOCKSIZE	CHAR (1)		Indica la granularidad de bloqueo preferida para tablas a las que acceden las sentencias DML (lenguaje de manipulación de datos). Sólo se aplica a tablas. Los valores posibles son: • I = Inserción de bloque • R = Fila • T = Tabla • Blanco = No se aplica
VOLATILE	CHAR (1)		• C = La cardinalidad de la tabla es volátil • Blanco = No se aplica
ROW_FORMAT	CHAR (1)		No se utiliza.
PROPERTY	VARCHAR (32)		Propiedades para una tabla. Un solo blanco indica que la tabla no tiene propiedades. Lo que se menciona a continuación es la posición en la serie, valor y significado: • 1, Y = tabla de consultas materializadas mantenida por el usuario • 2, Y = Tabla por etapas • 3, Y = Propagación inmediata • 11, Y = Apodo que no se pondrá en la antememoria
STATISTICS_PROFILE	CLOB(10M)	S	Mandato RUNSTATS utilizado para registrar un perfil estadístico para el objeto.
COMPRESSION	CHAR (1)		• B = Tanto la compresión de valores como la de filas están activadas • N = No hay ninguna compresión activada; se utiliza un formato de fila que no da soporte a la compresión • R = La compresión de filas está activada si tiene licencia; puede que se utilice un formato de fila que da soporte a la compresión • V = La compresión de valores está activada; se utiliza un formato de fila que da soporte a la compresión • Blanco = No se aplica

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ACCESS_MODE	CHAR (1)		Estado de restricción de acceso del objeto. Estos estados sólo se aplican a objetos que están en estado pendiente de establecer integridad o a objetos procesados por una sentencia SET INTEGRITY. Los valores posibles son: • D = No hay movimiento de datos • F = Acceso completo • N = No hay acceso • R = Acceso de sólo lectura
CLUSTERED	CHAR (1)	S	• Y = La tabla está en un clúster de varias dimensiones (aunque sólo según una dimensión) • Valor nulo = La tabla no está en un clúster de varias dimensiones
ACTIVE_BLOCKS	BIGINT		Número total de bloques activos en la tabla, o -1. Sólo se aplica a las tablas que están en un clúster de varias dimensiones (MDC).
DROPRULE	CHAR (1)		• N = No hay norma • R = Se aplica la norma restrictiva a la eliminación
MAXFREESPACESEARCH	SMALLINT		Reservado para una utilización futura.
AVGCOMPRESSEDROWSIZE	SMALLINT		Longitud media (en bytes) de filas comprimidas en esta tabla; -1 si no se recopilan estadísticas.
AVGROWCOMPRESSIONRATIO	REAL		Para filas comprimidas en la tabla, es la proporción media de compresión por fila; es decir, la longitud media de filas no comprimidas dividido por la longitud media de filas comprimidas; -1 si no se recopilan estadísticas.
AVGROWSIZE	SMALLINT		Longitud media (en bytes) de filas comprimidas y no comprimidas de esta tabla; -1 si no se recopilan estadísticas.
PCTROWSCOMPRESSED	REAL		Filas comprimidas como un porcentaje del número total de filas de la tabla; -1 si no se recopilan estadísticas.
LOGINDEXBUILD	VARCHAR(3)	S	Nivel de anotación cronológica que se debe realizar durante las operaciones de crear, volver a crear o reorganizar el índice de la tabla. • OFF = las operaciones de creación del índice de la tabla se anotarán cronológicamente de forma mínima • ON = las operaciones de creación del índice de la tabla se anotarán cronológicamente por completo • Valor nulo = 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
CODEPAGE	SMALLINT		Página de códigos del objeto. Es la página de códigos por omisión utilizada para todas las columnas de caracteres, activadores, restricciones de comprobación y columnas generadas por expresión.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para la tabla.

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para la tabla.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en la tabla.
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en la tabla.
ENCODING_SCHEME	CHAR (1)		• A = Se ha especificado CCSID ASCII • U = Se ha especificado CCSID UNICODE • Blanco = No se ha especificado la cláusula CCSID
PCTPAGESSAVED	SMALLINT		Porcentaje aproximado de páginas guardadas en la tabla como un resultado de la compresión de filas. Este valor incluye bytes de actividad general para cada fila de datos de usuario de la tabla, pero no incluye el espacio que consume la actividad general del diccionario; es -1 si no se recopilan estadísticas.
LAST_REGEN_TIME	TIMESTAMP	S	Hora a la que se han vuelto a generar por última vez las vistas y restricciones de comprobación de la tabla.
SECPOLICYID	INTEGER		Identificador de la política de seguridad que protege la tabla; 0 para tablas no protegidas.
PROTECTIONGRANULARITY	CHAR (1)		• B = Granularidad tanto a nivel de columna como a nivel de fila • C = Granularidad a nivel de columna • R = Granularidad a nivel de fila • Blanco = Tabla no protegida
AUDITPOLICYID	INTEGER	S	Identificador de la política de comprobación.
AUDITPOLICYNAME	VARCHAR (128)	S	Nombre de la política de comprobación.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la tabla, vista, alias o apodo.
ONCOMMIT	CHAR (1)		Especifica la acción que se realiza sobre la tabla temporal creada cuando se ejecuta una operación COMMIT. • D = Suprimir filas • P = Conservar filas • Blanco = La tabla no es una tabla temporal creada
LOGGED	CHAR (1)		Especifica si la tabla temporal creada está anotada. • N = No anotada • Y = Anotada • Blanco = La tabla no es una tabla temporal creada
ONROLLBACK	CHAR (1)		Especifica la acción que se realiza sobre la tabla temporal creada cuando se ejecuta una operación ROLLBACK. • D = Suprimir filas • P = Conservar filas • Blanco = La tabla no es una tabla temporal creada

Tabla 176. Vista de catálogo SYSCAT.TABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
LASTUSED	DATE		Fecha de última utilización de la tabla por parte de una sentencia DML o mandato LOAD. Esta columna no se actualiza en el caso de un alias, una tabla creada temporalmente, un apodo o una vista. El valor por omisión es '0001-01-01'. Este valor se actualiza de forma asíncrona.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.TABLESPACES

Cada fila representa un espacio de tablas.

Tabla 177. Vista de catálogo SYSCAT.TABLESPACES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TBSPACE	VARCHAR (128)		Nombre del espacio de tablas.
OWNER	VARCHAR (128)		ID de autorización del propietario del espacio de tablas.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el espacio de tablas.
TBSPACEID	INTEGER		Identificador del espacio de tablas.
TBSPACETYPE	CHAR (1)		Tipo de espacio de tablas. • D = Espacio gestionado por la base de datos • S = Espacio gestionado por el sistema
DATATYPE	CHAR (1)		Tipo de datos que se pueden almacenar en este espacio de tablas. • A = todos los tipos de datos permanentes; espacio de tablas regular • L = Todos los tipos de datos permanentes; espacio de tablas grande • T = Sólo tablas temporales del sistema • U = Sólo tablas temporales creadas o tablas temporadas declaradas
EXTENTSIZE	INTEGER		Tamaño de cada extensión, en páginas de tamaño PAGESIZE. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente.
PREFETCHSIZE	INTEGER		Número de páginas de tamaño PAGESIZE que se van a leer cuando se realice la captación previa; -1 para el valor AUTOMATIC.
OVERHEAD	DOUBLE		Actividad general del controlador y tiempo de latencia y de búsqueda en disco, en milisegundos (promedio para los contenedores de este espacio de tablas).
TRANSFERRATE	DOUBLE		Tiempo para leer una página de tamaño PAGESIZE en el almacenamiento intermedio (promedio para los contenedores de este espacio de tablas).
WRITEOVERHEAD	DOUBLE	S	Reservado para una utilización futura.

Tabla 177. Vista de catálogo SYSCAT.TABLESPACES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRITETRANSFERRATE	DOUBLE	S	Tiempo para grabar una página de tamaño PAGESIZE desde el almacenamiento intermedio al espacio de tablas (promedio para los contenedores de este espacio de tablas). El valor nulo significa que se utilizará el mismo valor que TRANSFERRATE.
PAGESIZE	INTEGER		Tamaño (en bytes) de páginas en este espacio de tablas.
DBPGNAME	VARCHAR (128)		Nombre del grupo de particiones de base de datos asociado a este espacio de tablas.
BUFFERPOOLID	INTEGER		Identificador de la agrupación de almacenamientos intermedios utilizada por este espacio de tablas (1 indica la agrupación de almacenamientos intermedios por omisión).
DROP_RECOVERY	CHAR (1)		Indica si las tablas de este espacio de tablas se pueden o no recuperar tras una operación de eliminación de la tabla. • N = Las tablas no se pueden recuperar • Y = Las tablas se pueden recuperar
NGNAME[1]	VARCHAR (128)		Nombre del grupo de particiones de base de datos asociado a este espacio de tablas.
DEFINER[2]	VARCHAR (128)		ID de autorización del propietario del espacio de tablas.
DATAPRIORITY	CHAR (1)		Reservado para una utilización futura.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna NGNAME se incluye por razones de compatibilidad con versiones anteriores. Consulte DBPGNAME.
2. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.TABOPTIONS

Cada fila representa una opción que está asociada a una tabla remota.

Tabla 178. Vista de catálogo SYSCAT.TABOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABSCHEMA	VARCHAR (128)		Nombre de esquema de una tabla, vista, alias o apodo.
TABNAME	VARCHAR (128)		Nombre no calificado de una tabla, vista, alias o apodo.
OPTION	VARCHAR (128)		Nombre de la opción de tabla.
SETTING	CLOB (32K)		Valor de la opción de tabla.

SYSCAT.TBSPACEAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado el privilegio USE sobre un determinado espacio de tablas de la base de datos.

Tabla 179. Vista de catálogo SYSCAT.TBSPACEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
TBSPACE	VARCHAR (128)		Nombre del espacio de tablas.
USEAUTH	CHAR (1)		Privilegio para crear tablas dentro del espacio de tablas. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.THRESHOLDS

Cada fila representa un umbral.

Tabla 180. Vista de catálogo SYSCAT.THRESHOLDS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
THRESHOLDNAME	VARCHAR (128)		Nombre del umbral.
THRESHOLDID	INTEGER		Identificador del umbral.
ORIGIN	CHAR (1)		Origen del umbral. • U = El usuario creó el umbral. • W = Un conjunto de acciones de trabajo creó el umbral.
THRESHOLDCLASS	CHAR (1)		Clasificación del umbral. • A = Umbral agregado • C = Umbral de actividad
THRESHOLDPREDICATE	VARCHAR (15)		Tipo de umbral. Los valores posibles son: • AGGTEMPSPACE • CONCDBC • CONCWCN • CONCWOC • CONNIDLETIME • CPUTIME • CPUTIMEINSC • DBCONN • ESTSQLCOST • ROWSREAD • ROWSREADINSC • ROWSRET • SCCONN • TEMPSPACE • TOTALTIME • UOWTOTALTIME
THRESHOLDPREDICATEID	SMALLINT		Identificador del predicado de umbral.
DOMAIN	CHAR (2)		Dominio del umbral. • DB = Base de datos • SB = Subclase de servicio • SP = Superclase de servicio • WA = Conjunto de acciones de trabajo • WD = Definición de carga de trabajo
DOMAINID	INTEGER		Identificador del objeto con el que está asociado el umbral. Puede ser una clase de servicio, acción de trabajo o ID exclusivo de carga de trabajo. Si es un umbral de base de datos, este valor es 0.

Tabla 180. Vista de catálogo SYSCAT.THRESHOLDS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ENFORCEMENT	CHAR (1)		Ámbito de implantación para el umbral. • D = Base de datos • P = Partición de base de datos • W = Aparición de carga de trabajo
QUEUING	CHAR (1)		• N = El umbral no está en cola • Y = El umbral está en cola
MAXVALUE	BIGINT		Límite superior especificado por el umbral.
QUEUESIZE	INTEGER		Si QUEUEING es 'Y', el tamaño de la cola. -1 en caso contrario.
COLLECTACTDATA	CHAR (1)		Especifica los datos de actividad que deben recopilarse por medio del supervisor de sucesos aplicable. • D = Datos de actividad con detalles • N = Ninguno • S = Datos de actividad con detalles y entorno de sección • V = Datos de actividad con detalles y valores • W = Datos de actividad sin detalles • X = Datos de actividad con detalles, entorno de sección y valores
COLLECTACTPARTITION	CHAR (1)		Especifica el lugar en el que se recopilarán los datos de actividad. • C = Partición de base de datos del coordinador de la actividad • D = Todas las particiones de base de datos
EXECUTION	CHAR (1)		Indica la acción de ejecución que se lleva a cabo después de que se haya superado un umbral. • C = La ejecución continúa • F = Se fuerza a que la aplicación salga del sistema • R = La ejecución se vuelve a correlacionar con una subclase de servicio diferente • S = La ejecución se detiene
REMAPSCID	SMALLINT		ID de la subclase de servicio de destino de la acción REMAP ACTIVITY.
VIOLATIONRECORDLOGGED	CHAR (1)		Indica si se ha anotado un registro en el supervisor de sucesos tras producirse la violación de umbral. • N = No • Y = Sí

Tabla 180. Vista de catálogo SYSCAT.THRESHOLDS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
CHECKINTERVAL	INTEGER		Intervalo, en segundos, en el que se comprueba la condición de umbral si THRESHOLDPREDICATE es: • 'CPUTIME' • 'CPUTIMEINSC' • 'ROWSREAD' • 'ROWSREADINSC' De lo contrario, -1.
ENABLED	CHAR (1)		• N = Se ha inhabilitado este umbral. • Y = Se ha habilitado este umbral.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado el umbral.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez el umbral.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.TRANSFORMS

Cada fila representa las funciones que manejan las transformaciones entre un tipo definido por el usuario y un tipo SQL base, o a la inversa.

Tabla 181. Vista de catálogo SYSCAT.TRANSFORMS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPEID	SMALLINT		Identificador para el tipo de datos
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos si TYPEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece el tipo de datos.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos.
GROUPNAME	VARCHAR (128)		Nombre del grupo de transformación.
FUNCID	INTEGER		Identificador de la rutina.
FUNCSCHEMA	VARCHAR (128)		Nombre de esquema de la rutina si ROUTINEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
FUNCNAME	VARCHAR (128)		Nombre no calificado de la rutina.
SPECIFICNAME	VARCHAR (128)		Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
TRANSFORMTYPE	VARCHAR(8)		• 'FROM SQL' = La función de transformación transforma un tipo estructurado desde SQL • 'TO SQL' = La función de transformación transforma un tipo estructurado a SQL
FORMAT	CHAR (1)		Formato generado por la transformación de SQL FROM. • S = Tipo de datos estructurado • U = Definido por el usuario
MAXLENGTH	INTEGER	S	Longitud máxima (en bytes) de la salida de la transformación FROM SQL; valor nulo para transformaciones de SQL TO.
ORIGIN	CHAR (1)		Fuente de este grupo de transformaciones. • O = Grupo de transformaciones original (incorporado o definido por el usuario) • R = Grupo de transformaciones redefinido (sólo los grupos incorporados se pueden redefinir)
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.TRIGDEP

Cada fila representa una dependencia de un activador sobre algún otro objeto. El activador depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta al activador.

Tabla 182. Vista de catálogo SYSCAT.TRIGDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TRIGSCHEMA	VARCHAR (128)		Nombre de esquema del activador.
TRIGNAME	VARCHAR (128)		Nombre no calificado del activador.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • K = Paquete • L = Tabla desenlazada • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • Q = Secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • X = Extensión de índice • Z = Objeto XSR • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.

Tabla 182. Vista de catálogo SYSCAT.TRIGDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita un activador dependiente; valor nulo en caso contrario.

SYSCAT.TRIGGERS

Cada fila representa un activador. Para jerarquías de tablas, cada activador se registra sólo al nivel de la jerarquía donde se ha creado.

Tabla 183. Vista de catálogo SYSCAT.TRIGGERS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TRIGSCHEMA	VARCHAR (128)		Nombre de esquema del activador.
TRIGNAME	VARCHAR (128)		Nombre no calificado del activador.
OWNER	VARCHAR (128)		ID de autorización del propietario del activador.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
TABSCHEMA	VARCHAR (128)		Nombre de esquema de la tabla o vista a la que se aplica este activador.
TABNAME	VARCHAR (128)		Nombre no calificado de la tabla o vista a la que se aplica este activador.
TRIGTIME	CHAR (1)		Momento en que se aplican las acciones activadas a la base de datos, en relación al suceso que ha disparado el activador. • A = El activador se aplica después del suceso • B = El activador se aplica antes del suceso • I = El activador se aplica en lugar del suceso
TRIGEVENT	CHAR (1)		Suceso que dispara el activador. • D = Operación de supresión • I = Operación de inserción • U = Operación de actualización
GRANULARITY	CHAR (1)		El activador se ejecuta una vez por: • R = Fila • S = Sentencia
VALID	CHAR (1)		• N = El activador no es válido • X = El activador no está operativo y se debe volver a crear • Y = El activador es válido
CREATE_TIME	TIMESTAMP		Hora en la que se ha definido el activador. Utilizada para resolver las funciones y tipos.
QUALIFIER	VARCHAR (128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.
FUNC_PATH	CLOB (2K)		Vía de acceso de SQL en vigor en el momento en que se definió el activador.
TEXT	CLOB(2M)		Texto completo de la sentencia CREATE TRIGGER, tal como se ha escrito.
LAST_REGEN_TIME	TIMESTAMP		Hora a la que se ha vuelto a generar por última ves el descriptor de paquete correspondiente al activador.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para el activador.
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para el activador.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en el activador.

Tabla 183. Vista de catálogo SYSCAT.TRIGGERS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en el activador.
PRECOMPILE_OPTIONS	VARCHAR(1024)	S	Opciones de precompilación y vinculación que estaban en vigor cuando se creó el activador compilado. Valor nulo si el activador no está compilado.
COMPILE_OPTIONS	VARCHAR(1024)	S	Valor del registro especial SQL_CCFLAGS que estaba en vigor cuando el activador compilado se creó y había presentes directivas de consulta. Serie vacía si no había presentes directivas de consulta en el activador compilado. Valor nulo si el activador no está compilado.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario del activador.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.TYPEMAPPINGS

Cada fila representa una correlación de tipos de datos entre un tipo de datos definido localmente y un tipo de datos de la fuente de datos. Hay dos tipos de correlación (direcciones de correlación):

- La correlación de tipos hacia adelante correlaciona un tipo de datos de fuente de datos con un tipo de datos definido localmente.
- La correlación de tipos hacia atrás correlaciona un tipo de datos definido localmente con un tipo de datos de fuente de datos.

Tabla 184. Vista de catálogo SYSCAT.TYPEMAPPINGS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TYPE_MAPPING	VARCHAR (18)		Nombre de la correlación de tipos (puede ser una correlación generada por el sistema).
MAPPINGDIRECTION	CHAR (1)		Indica si esta correlación de tipos es una correlación hacia adelante o hacia atrás. • F = Correlación de tipos hacia adelante • R = Correlación de tipos hacia atrás
TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo local en una correlación de tipos de datos; valor nulo para tipos incorporados.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo local en una correlación de tipos de datos.
TYPEID	SMALLINT		Identificador para el tipo de datos
SOURCETYPEID	SMALLINT		Identificador del tipo de fuente.
OWNER	VARCHAR (128)		ID de autorización del propietario de la correlación de tipos. 'SYSIBM' indica una correlación de tipos incorporada.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
LENGTH	INTEGER	S	Longitud o precisión máxima del tipo de datos local en este tipo de correlación. Si el valor es nulo, el sistema determina la longitud o precisión máxima. Para tipos de carácter, representa el número máximo de bytes.
SCALE	SMALLINT	S	Número máximo de dígitos de la parte fraccional de un valor decimal local o el número máximo de dígitos de segundos fraccionarios de un valor TIMESTAMP local en esta correlación. Si el valor es nulo, el sistema determina el número máximo.
LOWER_LEN	INTEGER	S	Longitud o precisión mínima del tipo de datos local en esta correlación. Si el valor es nulo, el sistema determina la longitud o precisión mínima. Para tipos de carácter, representa el número mínimo de bytes.

Tabla 184. Vista de catálogo SYSCAT.TYPEMAPPINGS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
UPPER_LEN	INTEGER	S	Longitud o precisión máxima del tipo de datos local en este tipo de correlación. Si el valor es nulo, el sistema determina la longitud o precisión máxima. Para tipos de carácter, representa el número máximo de bytes.
LOWER_SCALE	SMALLINT	S	Número mínimo de dígitos de la parte fraccional de un valor decimal local o el número mínimo de dígitos de segundos fraccionarios de un valor TIMESTAMP local en esta correlación. Si el valor es nulo, el sistema determina el número mínimo.
UPPER_SCALE	SMALLINT	S	Número máximo de dígitos de la parte fraccional de un valor decimal local o el número máximo de dígitos de segundos fraccionarios de un valor TIMESTAMP local en esta correlación. Si el valor es nulo, el sistema determina el número máximo.
S_OPR_P	CHAR (2)	S	Relación entre la escala y la precisión de un valor decimal local en esta correlación. Se pueden utilizar los operadores básicos de comparación (=, <, >, <=, >=, <>). Un valor nulo indica que no es necesaria ninguna relación específica.
BIT_DATA	CHAR (1)	S	Indica si este tipo de caracteres es o no para datos de bits. Los valores posibles son: • N = Este tipo no es para datos de bits • Y = Este tipo es para datos de bits • Valor nulo = No es un tipo de datos de caracteres o el sistema determina el atributo de datos de bits
WRAPNAME	VARCHAR (128)	S	Protocolo de acceso a datos (derivador) al que se aplica esta correlación.
SERVERNAME	VARCHAR (128)	S	Nombre en mayúsculas del servidor.
SERVERTYPE	VARCHAR (30)	S	Tipo de servidor.
SERVERVERSION	VARCHAR (18)	S	Versión de servidor.
REMOTE_TYPESCHEMA	VARCHAR (128)	S	Nombre de esquema del tipo de datos de la fuente de datos.
REMOTE_TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos de fuente de datos.
REMOTE_META_TYPE	CHAR (1)	S	Indica si este tipo remoto es un tipo incorporado del sistema o un tipo diferenciado. • S = Tipo incorporado del sistema • T = Tipo diferenciado

Tabla 184. Vista de catálogo SYSCAT.TYPEMAPPINGS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
REMOTE_LOWER_LEN	INTEGER	S	Longitud o precisión mínima del tipo de datos remoto en esta correlación. Para tipos de carácter, representa el número mínimo de caracteres (no bytes). Para tipos binarios, representa el número mínimo de bytes. Un valor de -1 indica que se utiliza la longitud o la precisión por omisión o que el tipo remoto carece de longitud o precisión.
REMOTE_UPPER_LEN	INTEGER	S	Longitud o precisión máxima del tipo de datos remoto en esta correlación. Para tipos de carácter, representa el número máximo de caracteres (no bytes). Para tipos binarios, representa el número máximo de bytes. Un valor de -1 indica que se utiliza la longitud o la precisión por omisión o que el tipo remoto carece de longitud o precisión.
REMOTE_LOWER_SCALE	SMALLINT	S	Número mínimo de dígitos de la parte fraccional de un valor decimal remoto o el número mínimo de dígitos de segundos fraccionarios de un valor TIMESTAMP remoto en esta correlación, o el valor nulo.
REMOTE_UPPER_SCALE	SMALLINT	S	Número máximo de dígitos de la parte fraccional de un valor decimal remoto o el número máximo de dígitos de segundos fraccionarios de un valor TIMESTAMP remoto en esta correlación, o el valor nulo.
REMOTE_S_OPR_P	CHAR (2)	S	Relación entre la escala y la precisión de un valor decimal remoto en esta correlación. Se pueden utilizar los operadores básicos de comparación (=, <, >, <=, >=, <>). Un valor nulo indica que no es necesaria ninguna relación específica.
REMOTE_BIT_DATA	CHAR (1)	S	Indica si este tipo de caracteres remoto es o no para datos de bits. Los valores posibles son: • N = Este tipo no es para datos de bits • Y = Este tipo es para datos de bits • Valor nulo = No es un tipo de datos de caracteres o el sistema determina el atributo de datos de bits
USER_DEFINED	CHAR (1)		Indica si la correlación es o no una correlación definida por el usuario. El valor siempre es 'Y'; es decir, la correlación siempre es una correlación definida por el usuario.
CREATE_TIME	TIMESTAMP		Hora en que se ha creado esta correlación.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la correlación de tipos. 'SYSIBM' indica una correlación de tipos incorporada.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

Tabla 184. Vista de catálogo SYSCAT.TYPEMAPPINGS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.USEROPTIONS

Cada fila representa un valor de opción de usuario específico del servidor.

Tabla 185. Vista de catálogo SYSCAT.USEROPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
AUTHID	VARCHAR (128)		ID de autorización local, en mayúsculas.
AUTHIDTYPE	CHAR (1)		• U = Se otorga a un usuario individual
SERVERNAME	VARCHAR (128)		Nombre del servidor en el que se ha definido el usuario.
OPTION	VARCHAR (128)		Nombre de la opción de usuario.
SETTING	VARCHAR (2048)		Valor de la opción de usuario.

SYSCAT.VARIABLEAUTH

Cada fila representa un usuario, grupo o rol al que un otorgante específico ha otorgado uno o más privilegios sobre una variable global de la base de datos que no está definida en un módulo.

Tabla 186. Vista de catálogo SYSCAT.VARIABLEAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
VARSCHEMA	VARCHAR (128)		Nombre de esquema de la variable global si VARMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la variable global.
VARNAME	VARCHAR (128)		Nombre no calificado de la variable global.
VARID	INTEGER		Identificador de la variable global.
READAUTH	CHAR (1)		Privilegio para leer la variable global. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene
WRITEAUTH	CHAR (1)		Privilegio para escribir la variable global. • G = Se mantiene y se puede otorgar • N = No se mantiene • Y = Se mantiene

SYSCAT.VARIABLEDEP

Cada fila representa una dependencia de una variable global sobre algún otro objeto. La variable global depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta a la variable global.

Tabla 187. Vista de catálogo SYSCAT.VARIABLEDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
VARSCHEMA	VARCHAR (128)		Nombre de esquema de la variable global que tiene dependencias sobre otro objeto.
VARMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece la variable global. El valor es nulo si no es una variable de módulo.
VARNAME	VARCHAR (128)		Nombre no calificado de la variable global que tiene dependencias sobre otro objeto.
VARMODULEID	INTEGER	S	Identificador para el módulo del objeto que tiene dependencias sobre otro objeto.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.

Tabla 187. Vista de catálogo SYSCAT.VARIABLEDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita la variable global dependiente; en caso contrario, valor nulo.

SYSCAT.VARIABLES

Cada fila representa una variable global.

Tabla 188. Vista de catálogo SYSCAT.VARIABLES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
VARSCHEMA	VARCHAR (128)		Nombre de esquema de la variable global si VARMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la variable global.
VARMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece la variable global. El valor es nulo si no es una variable de módulo.
VARNAME	VARCHAR (128)		Nombre no calificado de la variable global.
VARMODULEID	INTEGER	S	Identificador para el módulo al que pertenece la variable global. El valor es nulo si no es una variable de módulo.
VARID	INTEGER		Identificador de la variable global.
OWNER	VARCHAR (128)		ID de autorización del propietario de la variable global.
OWNERTYPE	CHAR (1)		• U = El propietario es un usuario individual
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la variable global.
LAST_REGEN_TIME	TIMESTAMP		Hora a la que se ha vuelto a generar por última vez la expresión por omisión.
VALID	CHAR (1)		• N = La variable global no es válida • Y = La variable global es válida
PUBLISHED	CHAR (1)		Indica si se puede hacer referencia a la variable de módulo desde fuera de su módulo. • N = La variable de módulo no está publicada • Y = La variable de módulo está publicada • Blanco = No se aplica
TYPESCHEMA	VARCHAR (128)		Nombre de esquema del tipo de datos si TYPEMODULEID es nulo; en caso contrario, nombre de esquema del módulo al que pertenece el tipo de datos.
TYPEMODULENAME	VARCHAR (128)		Nombre no calificado del módulo al que pertenece el tipo de datos de la variable. El valor es nulo si el tipo de datos de la variable no pertenece a ningún módulo.
TYPENAME	VARCHAR (128)		Nombre no calificado del tipo de datos.
TYPEMODULEID	INTEGER	S	Identificador para el módulo al que pertenece el tipo de datos de la variable. El valor es nulo si el tipo de datos de la variable no pertenece a ningún módulo.
LENGTH	INTEGER		Longitud máxima de la variable global.
SCALE	SMALLINT		Escala si el tipo de datos de la variable global es DECIMAL o un tipo diferenciado basado en DECIMAL; el número de dígitos de segundos fraccionarios si el tipo de datos de la variable global es TIMESTAMP o un tipo diferenciado basado en TIMESTAMP; de lo contrario, 0.
CODEPAGE	SMALLINT		Página de códigos de la variable global.
COLLATIONSCHEMA	VARCHAR (128)		Nombre de esquema de la clasificación para la variable.

Tabla 188. Vista de catálogo SYSCAT.VARIABLES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLATIONNAME	VARCHAR (128)		Nombre no calificado de la clasificación para la variable.
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		Nombre de esquema de la clasificación para cláusulas ORDER BY en la variable.
COLLATIONNAME_ORDERBY	VARCHAR (128)		Nombre no calificado de la clasificación para cláusulas ORDER BY en la variable.
SCOPE	CHAR (1)		Ámbito de la variable global. • S = Sesión
DEFAULT	CLOB (64K)	S	Expresión utilizada para calcular el valor inicial de la variable global la primera vez que se hace referencia a la misma.
QUALIFIER	VARCHAR (128)	S	Valor del esquema por omisión en el momento de la definición de la variable.
FUNC_PATH	CLOB (2K)	S	Vía de acceso de SQL en vigor en el momento en que se definió la variable.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.
READONLY	CHAR (1)		• C = Sólo lectura porque la variable global se define con una cláusula CONSTANT • N = No de sólo lectura

SYSCAT.VIEWS

Cada fila representa una vista o tabla de consulta materializada.

Tabla 189. Vista de catálogo SYSCAT.VIEWS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
VIEWSCHEMA	VARCHAR (128)		Nombre de esquema de la vista o tabla de consultas materializadas.
VIEWNAME	VARCHAR (128)		Nombre no calificado de la vista o tabla de consultas materializadas.
OWNER	VARCHAR (128)		ID de autorización del propietario de la vista o tabla de consulta materializada.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
SEQNO	SMALLINT		Siempre 1.
VIEWCHECK	CHAR (1)		Tipo de comprobación de vista. • C = Opción de comprobación en cascada • L = Opción de comprobación local • N = Sin opción de comprobación o es una tabla de consulta materializada
READONLY	CHAR (1)		• N = Pueden actualizar la vista los usuarios con la autorización adecuada o es una tabla de consulta materializada • Y = La vista es de sólo lectura debido a su definición
VALID	CHAR (1)		• N = La definición de la vista o tabla de consulta materializada no es válida • X = La definición de la vista o tabla de consulta materializada no está operativa y se debe volver a crear • Y = La definición de la vista o tabla de consulta materializada es válida
QUALIFIER	VARCHAR (128)		Valor del esquema por omisión en el momento de la definición del objeto. Se utiliza para completar cualquier referencia no calificada.
FUNC_PATH	CLOB (2K)		Vía de acceso de SQL en vigor en el momento en que se definió la vista o tabla de consulta materializada.
TEXT	CLOB(2M)		Texto completo de la sentencia CREATE de la vista o tabla de consulta materializada, tal como se ha escrito.
DEFINER[1]	VARCHAR (128)		ID de autorización del propietario de la vista o tabla de consulta materializada.

Nota:

1. La columna DEFINER se incluye por razones de compatibilidad con versiones anteriores. Consulte OWNER.

SYSCAT.WORKACTIONS

Cada fila representa una acción de trabajo definida para un conjunto de acciones de trabajo.

Tabla 190. Vista de catálogo SYSCAT.WORKACTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ACTIONNAME	VARCHAR (128)		Nombre de la acción de trabajo.
ACTIONID	INTEGER		Identificador de la acción de trabajo.
ACTIONSETNAME	VARCHAR (128)	S	Nombre del conjunto de acciones de trabajo.
ACTIONSETID	INTEGER		Identificador del conjunto de acciones de trabajo al que pertenece esta acción de trabajo. Esta columna hace referencia a la columna ACTIONSETID en la vista SYSCAT.WORKACTIONSETS.
WORKCLASSNAME	VARCHAR (128)	S	Nombre de la clase de trabajo.
WORKCLASSID	INTEGER		Identificador de la clase de trabajo. Esta columna hace referencia a la columna WORKCLASSID en la vista SYSCAT.WORKCLASSES.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la acción de trabajo.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez la acción de trabajo.
ENABLED	CHAR (1)		• N = Se ha inhabilitado esta acción de trabajo. • Y = Se ha habilitado esta acción de trabajo.

Tabla 190. Vista de catálogo SYSCAT.WORKACTIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ACTIONTYPE	CHAR (1)		Tipo de acción efectuada en cada actividad de DB2 que coincide con los atributos de la clase de trabajo que pertenece al ámbito. • B = Recopilar datos de actividad agregados básicos; sólo puede especificarse para los conjuntos de acciones de trabajo que se aplican a las clases de servicio o a las cargas de trabajo. • C = Permitir toda actividad de DB2 bajo la clase de trabajo asociada para ejecutar e incrementar el contador de la clase de trabajo. • D = Recopilar datos de actividad con detalles en la partición de base de datos del coordinador de la actividad. • E = Recopilar datos de actividad agregados ampliados; sólo puede especificarse para los conjuntos de acciones de trabajo que se aplican a las clases de servicio o las cargas de trabajo. • F = Recopilar datos de actividad con detalles, secciones y valores en la partición de la base de datos del coordinador de la actividad. • G = Recopilar detalles de actividad y secciones en la partición de la base de datos del coordinador de la actividad y recopilar datos de actividad en todas las particiones de la base de datos. • H = Recopilar detalles de actividad, secciones y valores en la partición de la base de datos del coordinador de la actividad y recopilar datos de actividad en todas las particiones de la base de datos. • M = Correlacionar con una subclase de servicio; sólo puede especificarse para los conjuntos de acciones de trabajo que se aplican a las clases de servicio. • P = Impedir la ejecución de cualquier actividad de DB2 bajo la clase de trabajo con la que está asociada esta acción de trabajo. • S = Recopilar datos de actividad con detalles y secciones en la partición de la base de datos del coordinador de la actividad. • T = La acción representa un umbral; sólo puede especificarse para los conjuntos de acciones de trabajo que están asociados con una base de datos o una carga de trabajo. • U = Correlacionar todas las actividades con un nivel de anidamiento de 0, y todas las actividades anidadas bajo estas actividades con una subclase de servicio; sólo puede especificarse para los conjuntos de acciones de trabajo que se aplican a las clases de servicio. • V = Recopilar datos de actividad con detalles y valores en la partición del coordinador. • W = Recopilar datos de actividad sin detalles en la partición del coordinador.

Tabla 190. Vista de catálogo SYSCAT.WORKACTIONS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
			• X = Recopilar datos de actividad con detalles en la partición del coordinador y recopilar datos de actividad en todas las particiones de la base de datos. • Y = Recopilar datos de actividad con detalles y valores en la partición del coordinador y recopilar datos de actividad en todas las particiones de la base de datos. • Z = Recopilar datos de actividad sin detalles en todas las particiones de la base de datos.
REFOBJECTID	INTEGER	S	Si ACTIONTYPE es 'M' (correlación) o 'N' (correlación anidada), este valor se establece en el ID de la subclase de servicio con la que se correlaciona la actividad de DB2. Si ACTIONTYPE es 'T' (umbral), este valor se establece en el ID del umbral a utilizar. Para todas las demás acciones, este valor es NULL.
REFOBJECTTYPE	VARCHAR (30)		Si ACTIONTYPE es 'M' o 'N', este valor se establece en 'SERVICE CLASS'; si ACTIONTYPE es 'T', este valor es 'THRESHOLD'; en caso contrario es un valor nulo.

SYSCAT.WORKACTIONSETS

Cada fila representa un conjunto de acciones de trabajo.

Tabla 191. Vista de catálogo SYSCAT.WORKACTIONSETS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
ACTIONSETNAME	VARCHAR (128)		Nombre del conjunto de acciones de trabajo.
ACTIONSETID	INTEGER		Identificador del conjunto de acciones de trabajo.
WORKCLASSSETNAME	VARCHAR (128)	S	Nombre del conjunto de clases de trabajo.
WORKCLASSSETID	INTEGER		El identificador del conjunto de clases de trabajo que ha de correlacionarse con el objeto especificado por medio de OBJECTID. Esta columna hace referencia a WORKCLASSSETID en la vista SYSCAT.WORKCLASSSETS.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el conjunto de acciones de trabajo.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez el conjunto de acciones de trabajo.
ENABLED	CHAR (1)		• N = Se ha inhabilitado este conjunto de acciones de trabajo. • Y = Se ha habilitado este conjunto de acciones de trabajo.
OBJECTTYPE	CHAR (1)		• b = Superclase de servicio • w = Carga de trabajo • Blanco = Base de datos
OBJECTNAME	VARCHAR (128)	S	Nombre de la clase de servicio o la carga de trabajo.
OBJECTID	INTEGER		El identificador del objeto al que está correlacionado el conjunto de clases de trabajo (especificado mediante WORKCLASSSETID). Si OBJECTTYPE es 'b', OBJECTID es el ID de la superclase de servicio. Si OBJECTTYPE es 'w', OBJECTID es el ID de la carga de trabajo. Si OBJECTTYPE es un blanco, OBJECTID es -1.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.WORKCLASSES

Cada fila representa una clase de trabajo definida para un conjunto de clases de trabajo.

Tabla 192. Vista de catálogo SYSCAT.WORKCLASSES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WORKCLASSNAME	VARCHAR (128)		Nombre de la clase de trabajo.
WORKCLASSSETNAME	VARCHAR (128)	S	Nombre del conjunto de clases de trabajo.
WORKCLASSID	INTEGER		Identificador de la clase de trabajo.
WORKCLASSSETID	INTEGER		Identificador del conjunto de clases de trabajo al que pertenece esta clase de trabajo. Esta columna hace referencia a la columna WORKCLASSSETID en la vista SYSCAT.WORKCLASSSETS.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la clase de trabajo.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez la clase de trabajo.
WORKTYPE	SMALLINT		El tipo de actividad de DB2. • 1 = ALL • 2 = READ • 3 = WRITE • 4 = CALL • 5 = DML • 6 = DDL • 7 = LOAD
RANGEUNITS	CHAR (1)		Las unidades a utilizar para el rango superior y el inferior. • C = Cardinalidad • T = Activaciones de temporizador • Blanco = No se aplica
FROMVALUE	DOUBLE	S	El valor inferior del rango en las unidades que especifica RANGEUNITS. Valor nulo cuando RANGEUNITS es un blanco.
TOVALUE	DOUBLE	S	El valor superior del rango en las unidades que especifica RANGEUNITS. Valor nulo cuando RANGEUNITS es un blanco. El valor -1 se utiliza para indicar que no hay vinculación superior.
ROUTINESCHEMA	VARCHAR (128)	S	Nombre de esquema de los procedimientos que se llaman desde la sentencia CALL. Valor nulo cuando WORKTYPE no es 4 (CALL) o 1 (ALL).
INITIALSQLDATAPRIORITY	CHAR (1)		Reservado para una utilización futura.
EVALUATIONORDER	SMALLINT		Identifica exclusivamente el orden de evaluación para seleccionar una clase de trabajo en un conjunto de clases de trabajo.

SYSCAT.WORKCLASSSETS

Cada fila representa un conjunto de clases de trabajo.

Tabla 193. Vista de catálogo SYSCAT.WORKCLASSSETS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WORKCLASSSETNAME	VARCHAR (128)		Nombre del conjunto de clases de trabajo.
WORKCLASSSETID	INTEGER		Identificador del conjunto de clases de trabajo.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado el conjunto de clases de trabajo.
ALTER_TIME	TIMESTAMP		Hora a la que se modificó por última vez el conjunto de clases de trabajo.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.WORKLOADAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado el privilegio USAGE sobre una carga de trabajo.

Tabla 194. Vista de catálogo SYSCAT.WORKLOADAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WORKLOADID	INTEGER		Identificador para la carga de trabajo.
WORKLOADNAME	VARCHAR (128)		Nombre de la carga de trabajo.
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• U = Se otorga a un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
USAGEAUTH	CHAR (1)		Indica si el otorgado retiene el privilegio de USAGE en la carga de trabajo. • N = No se mantiene • Y = Se mantiene

SYSCAT.WORKLOADCONNATTR

Cada fila representa un atributo de conexión en la definición de una carga de trabajo.

Tabla 195. Vista de catálogo SYSCAT.WORKLOADCONNATTR

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WORKLOADID	INTEGER		Identificador para la carga de trabajo.
WORKLOADNAME	VARCHAR (128)		Nombre de la carga de trabajo.
CONNATTRTYPE	VARCHAR (30)		Tipo del atributo de conexión. • 1 = APPLNAME • 2 = SYSTEM_USER • 3 = SESSION_USER • 4 = SESSION_USER GROUP • 5 = SESSION_USER ROLE • 6 = CURRENT CLIENT_USERID • 7 = CURRENT CLIENT_APPLNAME • 8 = CURRENT CLIENT_WRKSTNNAME • 9 = CURRENT CLIENT_ACCTNG • 10 = ADDRESS
CONNATTRVALUE	VARCHAR (1000)		Valor del atributo de conexión.

SYSCAT.WORKLOADS

Cada fila representa una carga de trabajo.

Tabla 196. Vista de catálogo SYSCAT.WORKLOADS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WORKLOADID	INTEGER		Identificador para la carga de trabajo.
WORKLOADNAME	VARCHAR (128)		Nombre de la carga de trabajo.
EVALUATIONORDER	SMALLINT		Orden de evaluación utilizado para seleccionar una carga de trabajo.
CREATE_TIME	TIMESTAMP		Hora a la que se ha creado la carga de trabajo.
ALTER_TIME	TIMESTAMP		Hora a la que se ha modificado por última vez la carga de trabajo.
ENABLED	CHAR (1)		• N = Se ha inhabilitado esta carga de trabajo. • Y = Se ha habilitado esta carga de trabajo.
ALLOWACCESS	CHAR (1)		• N = Una UOW asociada a esta carga de trabajo se rechazará. • Y = Una unidad de trabajo (UOW) asociada a esta carga de trabajo podrá acceder a la base de datos.
SERVICECLASSNAME	VARCHAR (128)		Nombre de la subclase de servicio a la que se asigna una unidad de trabajo (asociada a esta carga de trabajo).
PARENTSERVICECLASSNAME	VARCHAR (128)	S	Nombre de la superclase de servicio a la que se asigna una unidad de trabajo (asociada a esta carga de trabajo).
COLLECTAGGACTDATA	CHAR (1)		Especifica los datos de actividad agregados que deben capturarse para la carga de trabajo por medio del supervisor de sucesos aplicable. • B = Recopilar datos de actividad agregados básicos • E = Recopilar datos de actividad agregados ampliados • N = Ninguno
COLLECTACTDATA	CHAR (1)		Especifica los datos de actividad que deben recopilarse por medio del supervisor de sucesos aplicable. • D = Datos de actividad con detalles • N = Ninguno • S = Datos de actividad con detalles y entorno de sección • V = Datos de actividad con detalles y valores. Se aplica cuando la columna COLLECT se establece en 'C' • W = Datos de actividad sin detalles • X = Datos de actividad con detalles, entorno de sección y valores

Tabla 196. Vista de catálogo SYSCAT.WORKLOADS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLECTACTPARTITION	CHAR (1)		Especifica el lugar en el que se recopilarán los datos de actividad. • C = Partición de base de datos del coordinador de la actividad • D = Todas las particiones de base de datos
COLLECTDEADLOCK	CHAR (1)		Especifica los sucesos de punto muerto que deben recopilarse por medio del supervisor de sucesos aplicable. • H = Recopilar datos de punto muerto con actividades anteriores únicamente • N = No recopilar datos de punto muerto • V = Recopilar datos de punto muerto con valores y actividades anteriores • W = Recopilar datos de punto muerto sin valores y actividades anteriores
COLLECTLOCKTIMEOUT	CHAR (1)		Especifica los sucesos de tiempo de espera de bloqueo que deben recopilarse por medio del supervisor de sucesos aplicable. • H = Recopilar datos de tiempo de espera de bloqueo con actividades anteriores únicamente • N = No recopilar datos de tiempo de espera de bloqueo • V = Recopilar datos de tiempo de espera de bloqueo con valores y actividades anteriores • W = Recopilar datos de tiempo de espera de bloqueo sin valores y actividades anteriores
COLLECTLOCKWAIT	CHAR (1)		Especifica los sucesos de espera de bloqueo que deben recopilarse por medio del supervisor de sucesos aplicable. • H = Recopilar datos de espera de bloqueo con actividades anteriores únicamente • N = No recopilar datos de espera de bloqueo • V = Recopilar datos de espera de bloqueo con valores y actividades anteriores • W = Recopilar datos de espera de bloqueo sin valores y actividades anteriores
LOCKWAITVALUE	INTEGER		Especifica el tiempo en milisegundos que debe esperar un bloqueo antes de que el supervisor de sucesos aplicable recopile un suceso de bloqueo; 0 cuando COLLECTLOCKWAIT = 'N'

Tabla 196. Vista de catálogo SYSCAT.WORKLOADS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
COLLECTACTMETRICS	CHAR (1)		Especifica el nivel de supervisión para las actividades enviadas por una aparición de la carga de trabajo. • B = Recopilar métricas básicas de actividad • E = Recopilar métricas ampliadas de actividad • N = Ninguno
COLLECTUOWDATA	CHAR (1)		Especifica los datos de unidad de trabajo que deben recopilarse por medio del supervisor de sucesos aplicable. • B = Recopilar datos base de unidad de trabajo • N = Ninguno • P = Recopilar datos base de unidad de trabajo y la lista de paquetes
EXTERNALNAME	VARCHAR (128)	S	Reservado para una utilización futura.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.WRAPOPTIONS

Cada fila representa una opción específica del derivador.

Tabla 197. Vista de catálogo SYSCAT.WRAPOPTIONS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRAPNAME	VARCHAR (128)		Nombre del derivador.
OPTION	VARCHAR (128)		Nombre de la opción del derivador.
SETTING	VARCHAR (2048)		Valor de la opción del derivador.

SYSCAT.WRAPPERS

Cada fila representa un derivador registrado.

Tabla 198. Vista de catálogo SYSCAT.WRAPPERS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
WRAPNAME	VARCHAR (128)		Nombre del derivador.
WRAPTYPE	CHAR (1)		Tipo de derivador. • N = No relacional • R = Relacional
WRAPVERSION	INTEGER		Versión del derivador.
LIBRARY	VARCHAR (255)		Nombre del archivo que contiene el código utilizado para comunicarse con las fuentes de datos asociadas a este derivador.
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSCAT.XDBMAPGRAPHS

Cada fila representa un gráfico de esquema para un mapa XDB (objeto XSR).

Tabla 199. Vista de catálogo SYSCAT.XDBMAPGRAPHS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto XSR.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto XSR.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto XSR.
SCHEMAGRAPHID	INTEGER		Identificador del gráfico de esquema, que es exclusivo dentro de un identificador de mapa XDB.
NAMESPACE	VARCHAR(1001)	S	Identificador de serie para el URI de espacio de nombres del elemento raíz.
ROOTELEMENT	VARCHAR(1001)	S	Identificador de serie para el nombre del elemento raíz.

SYSCAT.XDBMAPSHREDTREES

Cada fila representa un árbol de trocear para un determinado gráfico de esquema.

Tabla 200. Vista de catálogo SYSCAT.XDBMAPSHREDTREES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto XSR.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto XSR.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto XSR.
SCHEMAGRAPHID	INTEGER		Identificador del gráfico de esquema, que es exclusivo dentro de un identificador de mapa XDB.
SHREDTREEID	INTEGER		Identificador del árbol de trocear, que es exclusivo dentro de un identificador de mapa XDB.
MAPPINGDESCRIPTION	CLOB(1M)	S	Información de correlación de diagnóstico.

SYSCAT.XMLSTRINGS

Cada fila representa un única serie y su ID de serie exclusivo; se usa para condensar los datos XML estructurales. La serie se proporciona con codificación UTF-8 y codificación en la página de códigos de la base de datos.

Tabla 201. Vista de catálogo SYSCAT.XMLSTRINGS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
STRINGID	INTEGER		ID de serie exclusivo.
STRING	VARCHAR(1001)		Serie representada en la página de códigos de la base de datos.
STRING_UTF8	VARCHAR(1001)		Serie en codificación UTF-8 (como se almacena en la tabla de catálogo).

SYSCAT.XSROBJECTAUTH

Cada fila representa un usuario, grupo o rol al que se ha otorgado el privilegio USAGE sobre un determinado objeto XSR.

Tabla 202. Vista de catálogo SYSCAT.XSROBJECTAUTH

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
GRANTOR	VARCHAR (128)		El que ha otorgado el privilegio.
GRANTORTYPE	CHAR (1)		• S = El sistema es el otorgante • U = El otorgante es un usuario individual
GRANTEE	VARCHAR (128)		El que mantiene el privilegio.
GRANTEETYPE	CHAR (1)		• G = Se otorga a un grupo • R = Se otorga a un rol • U = Se otorga a un usuario individual
OBJECTID	BIGINT		Identificador del objeto XSR.
USAGEAUTH	CHAR (1)		Privilegio para utilizar el objeto XSR y sus componentes. • N = No se mantiene • Y = Se mantiene

SYSCAT.XSROBJECTCOMPONENTS

Cada fila representa un componente de objeto XSR.

Tabla 203. Vista de catálogo SYSCAT.XSROBJECTCOMPONENTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto XSR.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto XSR.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto XSR.
COMPONENTID	BIGINT		Identificador exclusivo generado para un componente del objeto XSR.
TARGETNAMESPACE	VARCHAR(1001)	S	Identificador de serie del espacio de nombres de destino.
SCHEMALOCATION	VARCHAR(1001)	S	Identificador de serie de la ubicación del esquema.
COMPONENT	BLOB(30M)		Representación externa del componente.
CREATE_TIME	TIMESTAMP		Hora a la que se ha registrado el componente de objeto XSR.
STATUS	CHAR (1)		Estado de registro • C = Completo • I = Incompleto

SYSCAT.XSROBJECTDEP

Cada fila representa una dependencia de un objeto XSR sobre algún otro objeto. El objeto XSR depende del objeto de tipo BTYPE de nombre BNAME, de modo que un cambio en el objeto afecta al objeto XSR.

Tabla 204. Vista de catálogo SYSCAT.XSROBJECTDEP

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto XSR.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto XSR.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto XSR.
BTYPE	CHAR (1)		Tipo de objeto sobre el que hay una dependencia. Los valores posibles son: • A = Alias de tabla • B = Activador • F = Rutina • G = Tabla temporal global • H = Tabla de jerarquía • K = Paquete • L = Tabla desenlazada • N = Apodo • O = Dependencia de privilegios en todas las subtablas o subvistas de una jerarquía de tablas o de vistas • Q = Secuencia • R = Tipo de datos definido por el usuario • S = Tabla de consultas materializadas • T = Tabla (sin tipo) • U = Tabla con tipo • V = Vista (sin tipo) • W = Vista con tipo • X = Extensión de índice • Z = Objeto XSR • q = Alias de secuencia • u = Alias de módulo • v = Variable global • * = Anclada a la fila de una tabla base
BSCHEMA	VARCHAR (128)		Nombre de esquema del objeto sobre el que hay una dependencia.
BMODULENAME	VARCHAR(128)	S	Nombre no calificado del módulo al que pertenece el objeto sobre el que hay una dependencia. El valor es nulo si no es un objeto de módulo.
BNAME	VARCHAR (128)		Nombre no calificado del objeto sobre el que hay una dependencia. Para rutinas (BTYPE = 'F'), es el nombre específico.
BMODULEID	INTEGER	S	Identificador para el módulo del objeto sobre el que hay una dependencia.

Tabla 204. Vista de catálogo SYSCAT.XSROBJECTDEP (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
TABAUTH	SMALLINT	S	Si BTYPE = 'O', 'S', 'T', 'U', 'V', 'W' o 'v', codifica los privilegios sobre la tabla o vista que necesita un activador dependiente; valor nulo en caso contrario.

SYSCAT.XSROBJECTDETAILS

Cada fila representa un objeto de depósito del esquema XML.

Tabla 205. Vista de catálogo SYSCAT.XSROBJECTDETAILS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto de esquema XML.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto de esquema XML.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto de esquema XML.
GRAMMAR	BLOB (127M)	S	Representación binaria de la gramática del objeto de esquema XML.
PROPERTIES	BLOB (4190000)	S	Documento de propiedades del objeto de esquema XML.

SYSCAT.XSROBJECTHIERARCHIES

Cada fila representa la relación jerárquica entre un objeto XSR y sus componentes.

Tabla 206. Vista de catálogo SYSCAT.XSROBJECTHIERARCHIES

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador de un objeto XSR.
COMPONENTID	BIGINT		Identificador de un componente de XSR.
HTYPE	CHAR (1)		Tipo de jerarquía. • D = Documento • N = Espacio de nombres de nivel superior • P = Documento primario
TARGETNAMESPACE	VARCHAR(1001)	S	Identificador de serie del espacio de nombres de destino del componente.
SCHEMALOCATION	VARCHAR(1001)	S	Identificador de serie de la ubicación del esquema del componente.

SYSCAT.XSROBJECTS

Cada fila representa un objeto de depósito del esquema XML.

Tabla 207. Vista de catálogo SYSCAT.XSROBJECTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
OBJECTID	BIGINT		Identificador exclusivo generado para un objeto XSR.
OBJECTSCHEMA	VARCHAR (128)		Nombre de esquema del objeto XSR.
OBJECTNAME	VARCHAR (128)		Nombre no calificado del objeto XSR.
TARGETNAMESPACE	VARCHAR(1001)	S	Identificador de serie del espacio de nombres de destino o identificador público.
SCHEMALOCATION	VARCHAR(1001)	S	Identificador de serie de la ubicación del esquema o identificador del sistema.
OBJECTINFO	XML	S	Documento de metadatos.
OBJECTTYPE	CHAR (1)		Tipo de objeto XSR. • D = DTD • E = Entidad externa • S = Esquema XML
OWNER	VARCHAR (128)		ID de autorización del propietario del objeto XSR.
OWNERTYPE	CHAR (1)		• S = El sistema es el propietario • U = El propietario es un usuario individual
CREATE_TIME	TIMESTAMP		Hora a la que se ha registrado el objeto.
ALTER_TIME	TIMESTAMP		Hora a la que se ha actualizado (sustituido) por última vez el objeto.
STATUS	CHAR (1)		Estado de registro • C = Completo • I = Incompleto • R = Sustituir • T = Temporal
DECOMPOSITION	CHAR (1)		Indica si la descomposición (troceamiento) está o no habilitada sobre este objeto XSR. • N = No habilitada • X = No operativo • Y = Habilitada
REMARKS	VARCHAR (254)	S	Comentarios proporcionados por el usuario o valor nulo.

SYSIBM.SYSDUMMY1

Contiene una fila. Esta vista está disponible para las aplicaciones que necesitan compatibilidad con DB2 para z/OS.

Tabla 208. Vista de catálogo SYSIBM.SYSDUMMY1

Nombre de columna	Tipo de datos	Posibil. de nulos	Descripción
IBMREQD	CHAR(1)		'Y'

SYSSTAT.COLDIST

Cada fila representa el valor número *n* más frecuente de alguna columna, o el valor cuantil *n* (distribución acumulada) de la columna. Sólo se aplica a columnas de tablas reales (no a vistas). No se registra ninguna estadística para columnas heredadas de tablas con tipo.

Tabla 209. Vista de catálogo SYSSTAT.COLDIST

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
TABSCHEMA	VARCHAR (128)			Nombre de esquema de la tabla a la que se aplican las estadísticas.
TABNAME	VARCHAR (128)			Nombre no calificado de la tabla a la que se aplican las estadísticas.
COLNAME	VARCHAR (128)			Nombre de la columna a la que se aplican las estadísticas.
TYPE	CHAR (1)			• F = Valor de frecuencia • Q = Valor cuantil
SEQNO	SMALLINT			Si TYPE = 'F', *n* en esta columna identifica el valor número *n* más frecuente. Si TYPE = 'Q', *n* en esta columna identifica el valor cuantil número *n*.
COLVALUE[1]	VARCHAR (254)	S	S	El valor de los datos como un literal de caracteres o un valor nulo.
VALCOUNT	BIGINT		S	Si TYPE = 'F', VALCOUNT es el número de apariciones de COLVALUE en la columna. Si TYPE = 'Q', VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE.
DISTCOUNT[2]	BIGINT	S	S	Si TYPE = 'Q', esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE (valor nulo si no está disponible).

Nota:

1. En la vista de catálogo, el valor de COLVALUE siempre se muestra en la página de códigos de la base de datos y puede contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.
2. DISTCOUNT sólo se recopila para las columnas que son la primera columna de clave de un índice.

SYSSTAT.COLGROUPDIST

Cada fila representa el valor de la columna del grupo de columnas que forma el valor más frecuente número *n* del grupo de columnas o el valor cuantil *n* del grupo de columnas.

Tabla 210. Vista de catálogo SYSSTAT.COLGROUPDIST

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
COLGROUPID	INTEGER			Identificador del grupo de columnas.
TYPE	CHAR (1)			• F = Valor de frecuencia • Q = Valor cuantil
ORDINAL	SMALLINT			Número ordinal de la columna del grupo de columnas.
SEQNO	SMALLINT			Si TYPE = 'F', *n* en esta columna identifica el valor número *n* más frecuente. Si TYPE = 'Q', *n* en esta columna identifica el valor cuantil número *n*.
COLVALUE	VARCHAR (254)		S	El valor de los datos como un literal de caracteres o un valor nulo.

SYSSTAT.COLGROUPDISTCOUNTS

Cada fila representa las estadísticas de distribución que se aplican al valor más frecuente número *n* de un grupo de columnas o el valor cuantil *n* de un grupo de columnas.

Tabla 211. Vista de catálogo SYSSTAT.COLGROUPDISTCOUNTS

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
COLGROUPID	INTEGER			Identificador del grupo de columnas.
TYPE	CHAR (1)			• F = Valor de frecuencia • Q = Valor cuantil
SEQNO	SMALLINT			El número de secuencia *n* que representa el valor *n* de TYPE.
VALCOUNT	BIGINT		S	Si TYPE = 'F', VALCOUNT es el número de apariciones de COLVALUE para el grupo de columnas con este SEQNO. Si TYPE = 'Q', VALCOUNT es el número de filas cuyo valor es menor o igual que COLVALUE para el grupo de columnas con este SEQNO.
DISTCOUNT	BIGINT		S	Si TYPE = 'Q', esta columna registra el número de valores diferenciados que son menores o iguales que COLVALUE para el grupo de columnas con este SEQNO (valor nulo si no está disponible).

SYSSTAT.COLGROUPS

Cada fila representa un grupo de columnas y estadísticas que se aplican a todo el grupo de columnas.

Tabla 212. Vista de catálogo SYSSTAT.COLGROUPS

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
COLGROUPSCHEMA	VARCHAR (128)			Nombre de esquema del grupo de columnas.
COLGROUPNAME	VARCHAR (128)			Nombre no calificado del grupo de columnas.
COLGROUPID	INTEGER			Identificador del grupo de columnas.
COLGROUPCARD	BIGINT		S	Cardinalidad del grupo de columnas.
NUMFREQ_VALUES	SMALLINT			Número de valores frecuentes recopilados para el grupo de columnas.
NUMQUANTILES	SMALLINT			Número de valores cuantiles recopilados para el grupo de columnas.

SYSSTAT.COLUMNS

Cada fila representa una columna definida para una tabla, vista o apodo.

Tabla 213. Vista de catálogo SYSSTAT.COLUMNS

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
TABSCHEMA	VARCHAR (128)			Nombre de esquema de la tabla, vista o apodo que contiene la columna.
TABNAME	VARCHAR (128)			Nombre no calificado de la tabla, vista o apodo que contiene la columna.
COLNAME	VARCHAR (128)			Nombre de la columna.
COLCARD	BIGINT		S	Número de valores diferenciados de la columna; -1 se no se recopilan estadísticas; -2 para columnas heredadas y columnas de tablas de jerarquía.
HIGH2KEY[1]	VARCHAR (254)	S	S	El segundo valor de datos más alto. Representación de datos numéricos modificados por literales de caracteres. Vacío si no se recopilan estadísticas. Vacío para columnas heredadas y columnas de tablas jerárquicas.
LOW2KEY[1]	VARCHAR (254)	S	S	El segundo valor de datos más bajo. Representación de datos numéricos modificados por literales de caracteres. Vacío si no se recopilan estadísticas. Vacío para columnas heredadas y columnas de tablas jerárquicas.
AVGCOLLEN	INTEGER		S	Espacio promedio en bytes si la columna se guarda en la memoria de la base de datos o en una tabla temporal. En los tipos de datos LOB que no están en línea, los tipos de datos LONG y los documentos XML, el valor utilizado para calcular el promedio de longitud de columna es la longitud del descriptor de datos. Se requiere un byte adicional si la columna es anulable; -1 si no se han recopilado estadísticas; -2 para las columnas heredadas y las columnas de las tablas de jerarquía. Nota: el espacio promedio necesario para almacenar la columna en el disco puede ser diferente del valor representado por esta estadística.
NUMNULLS	BIGINT		S	Número de valores nulos de la columna; -1 si no se recopilan estadísticas.
PCTINLINED	SMALLINT			Porcentaje de datos LOB o documentos XML en línea. -1 si no se han recopilado estadísticas.
SUB_COUNT	SMALLINT		S	Número medio de subelementos de la columna. Sólo se aplica a columnas de tipo serie de caracteres.

Tabla 213. Vista de catálogo SYSSTAT.COLUMNS (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
SUB_DELIM_LENGTH	SMALLINT		S	Longitud media de los delimitadores que separan cada subelemento de la columna. Sólo se aplica a columnas de tipo serie de caracteres.
AVGCOLLENCHAR	INTEGER		S	Promedio del número de caracteres (basándose en la clasificación en vigor para la columna) necesarios para la columna; -1 si el tipo de datos de la columna es long, LOB o XML o si no se han recopilado estadísticas; -2 par columnas heredadas y columnas de tablas jerárquicas.

Nota:

1. En la vista de catálogo, los valores de HIGH2KEY y de LOW2KEY siempre se muestran en la página de códigos de la base de datos y pueden contener caracteres de sustitución. Sin embargo, las estadísticas se reúnen internamente en la página de códigos de la tabla de la columna y, por tanto, utilizarán los valores reales de la columna cuando se apliquen durante la optimización de la consulta.

SYSSTAT.INDEXES

Cada fila representa un índice. Los índices en tablas con tipo se representan mediante dos filas: una para el ″índice lógico″ de la tabla con tipo y otro para el ″índice-H″ de la tabla de jerarquía.

Tabla 214. Vista de catálogo SYSSTAT.INDEXES

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
INDSCHEMA	VARCHAR (128)			Nombre de esquema del índice.
INDNAME	VARCHAR (128)			Nombre no calificado del índice.
TABSCHEMA	VARCHAR (128)			Nombre de esquema de la tabla o apodo en el que se define el índice.
TABNAME	VARCHAR (128)			Nombre no calificado de la tabla o apodo en el que se define el índice.
COLNAMES	VARCHAR(640)			Esta columna ya no se utiliza y se eliminará en el próximo release.
NLEAF	BIGINT		S	Número de páginas de índice; -1 si no se recopilan estadísticas.
NLEVELS	SMALLINT		S	Número de niveles de índice; -1 si no se recopilan estadísticas.
FIRSTKEYCARD	BIGINT		S	Número de valores de primera clave diferenciada; -1 si no se recopilan estadísticas.
FIRST2KEYCARD	BIGINT		S	Número de claves diferenciadas que utilizan las dos primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST3KEYCARD	BIGINT		S	Número de claves diferenciadas que utilizan las tres primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FIRST4KEYCARD	BIGINT		S	Número de claves diferenciadas que utilizan las cuatro primeras columnas del índice; -1 si no se recopilan estadísticas o si no se aplica.
FULLKEYCARD	BIGINT		S	Número de valores de clave completa diferenciada; -1 si no se recopilan estadísticas.
CLUSTERRATIO[4]	SMALLINT		S	Grado de clúster de datos con el índice; -1 si no se recopilan estadísticas o si se recopilan estadísticas detalladas de índice (en este caso se utilizará CLUSTERFACTOR en su lugar).
CLUSTERFACTOR[4]	DOUBLE		S	Mejor medición del grado de clúster; -1 si no se recopilan estadísticas o si el índice se define sobre un apodo.
SEQUENTIAL_PAGES	BIGINT		S	Número de páginas ubicadas en disco por orden de clave de índice, con pocos o ningún hueco entre ellas; -1 si no se recopilan estadísticas.

Tabla 214. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
DENSITY	INTEGER		S	Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100; -1 si no se recopilan estadísticas.
PAGE_FETCH_PAIRS[4]	VARCHAR(520)		S	Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. Serie de longitud cero si no hay datos disponibles.
NUMRIDS[4]	BIGINT		S	Número total de identificadores de fila (RID) o identificadores de bloque (BID) del índice; -1 si no se conoce.
NUMRIDS_DELETED[4]	BIGINT		S	Número total de identificadores de filas (o identificadores de bloque) del índice que están marcados como suprimidos, excluidos los identificadores de páginas de hojas en las que todos los identificadores de filas están marcados como suprimidos.
NUM_EMPTY_LEAFS	BIGINT		S	Número total de páginas de hoja del índice que tienen todos los identificadores de filas (o identificadores de bloqueo) marcados como suprimidos.
AVERAGE_RANDOM_FETCH_PAGES[1,2,4]	DOUBLE		S	Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice; -1 si no se conoce.
AVERAGE_RANDOM_PAGES[2]	DOUBLE		S	Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales; -1 si no se conoce.
AVERAGE_SEQUENCE_GAP[2]	DOUBLE		S	Espacio entre las secuencias de páginas del índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas del índice que deben captarse de forma aleatoria entre las secuencias de las páginas del índice; -1 si no se conoce.
AVERAGE_SEQUENCE_FETCH_GAP[1,2,4]	DOUBLE		S	Espacio entre las secuencias de páginas de la tabla al captar utilizando el índice. Detectado mediante una exploración de las páginas del índice, cada espacio representa el promedio de páginas de la tabla que deben captarse de forma aleatoria entre las secuencias de las páginas de la tabla; -1 si no se conoce.

Tabla 214. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
AVERAGE_SEQUENCE_PAGES[2]	DOUBLE		S	Promedio de páginas del índice accesibles en secuencia (es decir, el número de páginas del índice que la captación previa detectaría que forman una secuencia); -1 si no se conoce.
AVERAGE_SEQUENCE_FETCH_PAGES[1,2,4]	DOUBLE		S	Promedio de páginas de la tabla accesibles en secuencia (es decir, el número de páginas de la tabla que la captación previa detectaría que forman una secuencia) al captar utilizando el índice; -1 si no se conoce.
AVGPARTITION_CLUSTERRATIO[3,4]	SMALLINT		S	Nivel de clúster de los datos dentro de una sola partición de datos. -1 si la tabla no está particionada, si no se recopilan estadísticas o si se recopilan estadísticas detalladas (en cuyo caso se utiliza AVGPARTITION_ CLUSTERFACTOR en su lugar).
AVGPARTITION_CLUSTERFACTOR[3,4]	DOUBLE		S	Mejor medición del nivel de clúster dentro de una sola partición de datos. -1 si la tabla no está particionada, si no se recopilan estadísticas o si el índice está definido sobre un apodo.
AVGPARTITION_PAGE_FETCH_PAIRS[3,4]	VARCHAR(520)		S	Una lista de pares de enteros en formato de caracteres. Cada par representa un tamaño potencial de la agrupación de almacenamientos intermedios en las captaciones de páginas correspondientes necesarias para acceder a una sola partición de datos desde la tabla. Serie de longitud cero si no hay datos disponibles o si la tabla no está particionada.
DATAPARTITION_CLUSTERFACTOR	DOUBLE		S	Una medición estadística del ″clúster″ de claves de índices con respecto a particiones de datos. Es un número comprendido entre 0 y 1; 1 representa clúster perfecto y 0 representa que no hay clúster.
INDCARD	BIGINT		S	Cardinalidad del índice. Puede ser distinta de la cardinalidad de la tabla para índices que no tienen una relación de uno a uno entre las filas de la tabla y las entradas de índice.
PCTPAGESSAVED	SMALLINT			Porcentaje aproximado de páginas guardadas en el índice como un resultado de la compresión de índice. -1 si no se recopilan estadísticas.
AVGLEAFKEYSIZE	INTEGER		S	Promedio del tamaño de clave de índice para las claves en páginas hojas del índice.

Tabla 214. Vista de catálogo SYSSTAT.INDEXES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
AVGNLEAFKEYSIZE	INTEGER		S	Promedio del tamaño de clave de índice para las claves en páginas no hojas del índice.

Nota:

1. Cuando se utilizan espacios de tablas DMS, no puede calcularse esta estadística.
2. No se recopilan estadísticas de captación previa durante una operación LOAD...STATISTICS YES o CREATE INDEX...COLLECT STATISTICS ni cuando el parámetro de configuración *seqdetect* está desactivado.
3. AVGPARTITION_CLUSTERRATIO, AVGPARTITION_CLUSTERFACTOR y AVGPARTITION_PAGE_FETCH_PAIRS miden el grado de clúster dentro de una sola partición de datos (clúster local). CLUSTERRATIO, CLUSTERFACTOR y PAGE_FETCH_PAIRS miden el grado de clúster en la tabla entera (clúster global). Los valores de clúster global y de clúster local pueden diferir significativamente si la clave de particionamiento de la tabla no es un prefijo de la clave de índice o cuando la clave de particionamiento de la tabla y la clave de índice son independientes, en términos lógicos, una de otra.
4. Esta estadística no se puede actualizar si el tipo de índice es 'XPTH' (un índice de vía de acceso XML).
5. Puesto que los índices lógicos de una columna XML no tienen estadísticas, la vista de catálogo SYSSTAT.INDEXES no incluye las filas cuyo tipo de índice es 'XVIL'.

SYSSTAT.ROUTINES

Cada fila representa una rutina definida por el usuario (función escalar, función de tabla, función fuente, método o procedimiento). No incluye las funciones incorporadas.

Tabla 215. Vista de catálogo SYSSTAT.ROUTINES

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
ROUTINESCHEMA	VARCHAR (128)			Nombre de esquema de la rutina si ROUTINEMODULENAME es nulo; en caso contrario, nombre de esquema del módulo al que pertenece la rutina.
ROUTINEMODULENAME	VARCHAR (128)			Nombre no calificado del módulo al que pertenece la rutina. El valor es nulo si no es una rutina de módulo.
ROUTINENAME	VARCHAR (128)			Nombre no calificado de la rutina.
ROUTINETYPE	CHAR (1)			Tipo de rutina. • F = Función • M = Método • P = Procedimiento
SPECIFICNAME	VARCHAR (128)			Nombre de la instancia de la rutina (puede ser un nombre generado por el sistema).
IOS_PER_INVOC	DOUBLE		S	Número estimado de entradas/salidas (I/O) por invocación; 0 es el valor por omisión; -1 si no se conoce.
INSTS_PER_INVOC	DOUBLE		S	Número estimado de instrucciones por invocación; 450 es el valor por omisión; -1 si no se conoce.
IOS_PER_ARGBYTE	DOUBLE		S	Número estimado de I/O por byte de argumento de entrada; 0 es el valor por omisión; -1 si no se conoce.
INSTS_PER_ARGBYTE	DOUBLE		S	Número estimado de instrucciones por byte de argumento de entrada; 0 es el valor por omisión; -1 si no se conoce.
PERCENT_ARGBYTES	SMALLINT		S	Porcentaje medio estimado de bytes de argumento de entrada que leerá la rutina realmente; 100 es el valor por omisión; -1 si no se conoce.
INITIAL_IOS	DOUBLE		S	Número estimado de I/O realizadas la primera vez que se ha invocado la rutina; 0 es el valor por omisión; -1 si no se conoce.
INITIAL_INSTS	DOUBLE		S	Número estimado de instrucciones ejecutadas la primera vez que se ha invocado la rutina; 0 es el valor por omisión; -1 si no se conoce.
CARDINALITY	BIGINT		S	Cardinalidad prevista de una función de tabla; -1 si no se conoce o si la rutina no es una función de tabla.

Tabla 215. Vista de catálogo SYSSTAT.ROUTINES (continuación)

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
SELECTIVITY	DOUBLE		S	Para predicados definidos por el usuario; -1 si no hay ningún predicado definido por el usuario.

SYSSTAT.TABLES

Cada fila representa una tabla, vista, alias o apodo. Cada jerarquía de tabla o de vista tiene una fila adicional que representa la tabla de jerarquía o la vista de jerarquía que implanta la jerarquía. Se incluyen las tablas de catálogo y las vistas.

Tabla 216. Vista de catálogo SYSSTAT.TABLES

Nombre de columna	Tipo de datos	Posibil. de nulos	Actualizable	Descripción
TABSCHEMA	VARCHAR (128)			Nombre de esquema del objeto.
TABNAME	VARCHAR (128)			Nombre no calificado del objeto.
CARD	BIGINT		S	Número total de filas de la tabla; -1 si no se recopilan estadísticas.
NPAGES	BIGINT		S	Número total de páginas en las que existen filas de la tabla; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
FPAGES	BIGINT		S	Número total de páginas; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
OVERFLOW	BIGINT		S	Número total de registros de desbordamiento de la tabla; -1 para una vista o alias, o si no se recopilan estadísticas; -2 para una subtabla o tabla de jerarquía.
CLUSTERED	CHAR (1)	S		• Y = La tabla está en un clúster de varias dimensiones (aunque sólo según una dimensión) • Valor nulo = La tabla no está en un clúster de varias dimensiones
ACTIVE_BLOCKS	BIGINT		S	Número total de bloques activos en la tabla, o -1. Sólo se aplica a las tablas que están en un clúster de varias dimensiones (MDC).
AVGCOMPRESSEDROWSIZE	SMALLINT		S	Longitud media (en bytes) de filas comprimidas en esta tabla; -1 si no se recopilan estadísticas.
AVGROWCOMPRESSIONRATIO	REAL		S	Para filas comprimidas en la tabla, es la proporción media de compresión por fila; es decir, la longitud media de filas no comprimidas dividido por la longitud media de filas comprimidas; -1 si no se recopilan estadísticas.
AVGROWSIZE	SMALLINT			Longitud media (en bytes) de filas comprimidas y no comprimidas de esta tabla; -1 si no se recopilan estadísticas.
PCTROWSCOMPRESSED	REAL		S	Filas comprimidas como un porcentaje del número total de filas de la tabla; -1 si no se recopilan estadísticas.
PCTPAGESSAVED	SMALLINT		S	Porcentaje aproximado de páginas guardadas en la tabla como un resultado de la compresión de filas. Este valor incluye bytes de actividad general para cada fila de datos de usuario de la tabla, pero no incluye el espacio que consume la actividad general del diccionario; es -1 si no se recopilan estadísticas.

Apéndice E. Sistemas federados

Tipos de servidor válidos en sentencias de SQL

El tipo de servidor indica la clase de fuentes de datos que representa la definición de servidor.

Los tipos de servidor varían según el proveedor, la finalidad y el sistema operativo. Los valores soportados dependen de la fuente de datos.

Para la mayoría de las fuentes de datos, debe especificar un tipo de servidor válido en la sentencia CREATE SERVER.

Tabla 217. Fuentes de datos y tipos de servidor

Fuente de datos	Tipo de servidor
BioRS	No es necesario un tipo de servidor en la sentencia CREATE SERVER.
Excel	No es necesario un tipo de servidor en la sentencia CREATE SERVER.
IBM DB2 Universal Database para Linux, UNIX y Windows	DB2/UDB
IBM DB2 Universal Database para System i y AS/400	DB2/ISERIES
IBM DB2 Universal Database para z/OS	DB2/ZOS
IBM DB2 para VM	DB2/VM
Informix	INFORMIX
JDBC	JDBC (necesario para fuentes de datos JDBC soportadas por controladores JDBC 3.0 y posterior).
Microsoft SQL Server	MSSQLSERVER (necesario para las fuentes de datos soportadas por el controlador DataDirect Connect ODBC 4.2 (o posterior) o el controlador Microsoft SQL Server ODBC 3.0 (o posterior)).
ODBC	ODBC (necesario para fuentes de datos ODBC soportadas por el controlador ODBC 3.x).
OLE DB	No es necesario un tipo de servidor en la sentencia CREATE SERVER.
Oracle	ORACLE (necesario para fuentes de datos Oracle soportadas por el software de cliente Oracle NET8).
Sybase (CTLIB)	SYBASE
Archivos con estructura de tabla	No es necesario un tipo de servidor en la sentencia CREATE SERVER.
Teradata	TERADATA
Servicios Web	No es necesario un tipo de servidor en la sentencia CREATE SERVER.
XML	No es necesario un tipo de servidor en la sentencia CREATE SERVER.

Opciones de correlación de funciones para sistemas federados

El servidor federado proporciona correlaciones por omisión entre funciones de DB2 y funciones de fuente de datos. Para la mayoría de fuentes de datos, las correlaciones de funciones por omisión están en los derivadores. Para utilizar una función de fuente de datos que el servidor federado no reconoce o para cambiar la correlación por omisión, debe crear una correlación de funciones.

Cuando se crea una correlación de funciones, se especifica el nombre de la función de fuente de datos y se debe habilitar la función correlacionada. Después, cuando se utiliza la función correlacionada, el optimizador de consultas compara el coste de ejecutar la función en la fuente de datos con el coste de ejecutar la función en el servidor federado.

Tabla 218. Opciones para las correlaciones de funciones

Nombre	Descripción
DISABLE	Habilita o inhabilita una correlación de funciones por omisión. Los valores válidos son Y y N. El valor por omisión es N.
REMOTE_NAME	El nombre de la función de fuente de datos. El valor por omisión es el nombre local.

Correlaciones de tipos de datos directas por omisión

Existen dos tipos de correlaciones entre los tipos de datos de la fuente de datos y los tipos de datos de la base de datos federada: las correlaciones de tipos directas y las correlaciones de tipos inversas. En una correlación de tipos directa, la correlación se realiza desde un tipo remoto a un tipo local comparable.

Puede alterar temporalmente una correlación de tipos por omisión o crear una nueva con la sentencia CREATE TYPE MAPPING.

Estas correlaciones son válidas con todas las versiones soportadas, a menos que se indique lo contrario.

Para todas las correlaciones de tipos de datos directas por omisión desde una fuente de datos a la base de datos federada, el esquema federado es SYSIBM.

En las tablas siguientes se muestran las correlaciones directas por omisión entre tipos de datos de base de datos federada y tipos de datos de fuente de datos.

Correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 Database para Linux, UNIX y Windows

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 Database para Linux, UNIX y Windows.

Tabla 219. Correlaciones de tipos de datos directas por omisión de DB2 Database para Linux, UNIX y Windows (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DATE	-	-	-	-	-	-	TIMESTAMP [1]	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DECFLOAT [2]	-	-	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(*p*)	-	*p*	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

Nota:

1. El tipo federado es TIMESTAMP(0) si el parámetro de configuración date_compat está establecido en ON.
2. Por omisión, la opción de servidor SAME_DECFLT_ROUNDING está establecida en N y ninguna operación se impulsará a la fuente de datos remota a menos que SAME_DECFLT_ROUNDING esté establecida en Y. Para obtener información sobre la opción de servidor SAME_DECFLT_ROUNDING, consulte la referencia de opciones de bases de datos DB2.

Correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para System i

La tabla siguiente lista las correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para System i.

Tabla 220. Correlaciones de tipos de datos directas por omisión para DB2 para System i (No se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para VM y VSE

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para VM y VSE.

Tabla 221. Correlaciones de tipos de datos directas por omisión de DB2 Server para VM y VSE (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para z/OS

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos DB2 para z/OS.

Tabla 222. Correlaciones de tipos de datos directas por omisión de DB2 para z/OS (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	-	6	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Correlaciones de tipos de datos directas por omisión para fuentes de datos Informix

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos Informix.

Tabla 223. Correlaciones de tipos de datos directas por omisión de Informix (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATE	-	-	-	-	-	-	TIMESTAMP [1]	-	0	-
DATETIME[2]	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP(6)	10	6	-
DATETIME	6	10	11	15	-	-	TIMESTAMP(6)	10	6	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
DECIMAL	1	32	255	255	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

Notas:

1. El tipo federado es TIMESTAMP(0) si el parámetro de configuración date_compat está establecido en ON.
2. Para el tipo de datos DATETIME de Informix, el servidor federado de DB2, UNIX y Windows utiliza el calificador de alto nivel de Informix como REMOTE_LENGTH y el calificador de bajo nivel de Informix como REMOTE_SCALE.

Los calificadores de Informix son las constantes "TU_" definidas en el archivo `datatime.h` del SDK del cliente Informix. Las constantes son:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)

Correlaciones de tipos de datos directas por omisión para fuentes de datos Informix

Tabla 223. Correlaciones de tipos de datos directas por omisión de Informix (no se muestran todas las columnas) (continuación)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
2 = MONTH		10 = SECOND					14 = FRACTION(4)			
4 = DAY		11 = FRACTION(1)					15 = FRACTION(5)			
6 = HOUR		12 = FRACTION(2)								

Correlaciones de tipos de datos directas por omisión para fuentes de datos Microsoft SQL Server

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos Microsoft SQL Server.

Tabla 224. Correlaciones de tipos de datos directas por omisión de Microsoft SQL Server

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
bigint [1]	-	-	-	-	-	-	BIGINT	-	-	-
binary	1	254	-	-	-	-	CHARACTER	-	-	Y
binary	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	8	-	-	-	-	DOUBLE	8	-	-
float	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	8	-	-
ntext	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID	-	-	-	-	-	-	VARCHAR	-	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N

Correlaciones de tipos de datos directas por omisión para fuentes de datos Microsoft SQL Server

Tabla 224. Correlaciones de tipos de datos directas por omisión de Microsoft SQL Server (continuación)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	6	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	254	-	-	-	-	CHARACTER	-	-	N
SQL_WCHAR	255	8800	-	-	-	-	VARCHAR	-	-	N
SQL_WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
text	-	-	-	-	-	-	CLOB	-	-	N
timestamp	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

Nota:

1. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 2000.

Correlaciones de tipos de datos directas por omisión para fuentes de datos ODBC

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos ODBC.

Tabla 225. Correlaciones de tipos de datos directas por omisión de ODBC (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	-	-	-
SQL_DATE	-	-	-	-	-	-	TIMESTAMP [1]	-	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	8	-	-	-	-	FLOAT	8	-	-
SQL_FLOAT	-	4	-	-	-	-	FLOAT	4	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	2147483647	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
SQL_TIMESTAMP(*p*)	-	-	-	-	-	-	TIMESTAMP(6)	10	6	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_ TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

Nota:

1. El tipo federado es TIMESTAMP(0) si el parámetro de configuración date_compat está establecido en ON.

Correlaciones de tipos de datos directas por omisión para fuentes de datos Oracle NET8

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos Oracle NET8.

Tabla 226. Correlaciones de tipos de datos directas por omisión para Oracle NET8

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	0	0	0	0	-	\0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	\0	CHAR	0	0	N
CHAR	255	2000	0	0	-	\0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	\0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	\0	TIMESTAMP(6)	0	0	N
FLOAT	1	126	0	0	-	\0	DOUBLE	0	0	N
LONG	0	0	0	0	-	\0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	\0	BLOB	2147483647	0	Y
NUMBER	10	18	0	0	-	\0	BIGINT	0	0	N
NUMBER	1	38	-84	127	-	\0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	\0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	\0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	\0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	\0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP(*p*) [1]	-	-	-	-	-	\0	TIMESTAMP(6)	10	6	N
VARCHAR2	1	4000	0	0	-	\0	VARCHAR	0	0	N

Nota:

1.
- TIMESTAMP*(p)* representa una indicación de fecha y hora con una escala variable de 0-9. La escala de la indicación de fecha y hora de Oracle se correlaciona con TIMESTAMP(6) por omisión. Puede modificar esta correlación de tipos por omisión y correlacionar la TIMESTAMP de Oracle con la TIMESTAMP federada de la misma escala mediante una correlación de tipos definida por el usuario.
- Esta correlación de tipos solo es válida para configuraciones de cliente y servidor de Oracle 9i (o versión posterior).

Correlaciones de tipos de datos directas por omisión para fuentes de datos Sybase

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos Sybase.

Tabla 227. Correlaciones de tipos de datos directas por omisión para Sybase CTLIB

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	32672	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	32672	-	-	-	-	VARCHAR	-	-	N
char null (véase varchar)										
date	-	-	-	-	-	-	DATE	-	-	-
date	-	-	-	-	-	-	TIMESTAMP [1]	-	-	-
datetime	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
decimal	1	31	0	31	-	-	DECIMAL	-	-	-
decimal	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
float	-	4	-	-	-	-	REAL	-	-	-
float	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	32672	-	-	-	-	VARCHAR	-	-	N
nchar null (véase nvarchar)										
numeric	1	31	0	31	-	-	DECIMAL	-	-	-
numeric	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	32672	-	-	-	-	VARCHAR	-	-	N
real	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP(6)	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	-	-	-	-	-	-	VARCHAR	30	-	N

Correlaciones de tipos de datos directas por omisión para fuentes de datos Sybase

Tabla 227. Correlaciones de tipos de datos directas por omisión para Sybase CTLIB (continuación)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
text	-	-	-	-	-	-	CLOB	-	-	-
time	-	-	-	-	-	-	TIME	-	-	-
timestamp	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[2]	1	254	-	-	-	-	CHAR	-	-	N
unichar[2]	255	32672	-	-	-	-	VARCHAR	-	-	N
unichar null (véase univarchar)										
univarchar[2]	1	32672	-	-	-	-	VARCHAR	-	-	N
varbinary	1	32672	-	-	-	-	VARCHAR	-	-	Y
varchar	1	32672	-	-	-	-	VARCHAR	-	-	N

Nota:

1. El tipo federado es TIMESTAMP(0) si el parámetro de configuración date_compat está establecido en ON.
2. Válido para bases de datos federadas que no utilizan Unicode.

Correlaciones de tipos de datos directas por omisión para fuentes de datos Teradata

La siguiente tabla lista las correlaciones de tipos de datos directas por omisión para fuentes de datos Teradata.

Tabla 228. Correlaciones de tipos de datos directas por omisión de Teradata (no se muestran todas las columnas)

Nombre tipo remoto	Long. inf. remota	Long. sup. remota	Escala inf. remota	Escala sup. remota	Datos bit remotos	Operad. datos remotos	Nombre tipo federado	Long. federada	Escala federada	Datos bit federados
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
CLOB	1	2097088000 (Latín)	-	-	-	-	CLOB	-	-	-
CLOB	1	1048544000 (Unicode)	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DATE	-	-	-	-	-	-	TIMESTAMP [1]	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	0	21	0	21	-	-	TIME	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(6)	10	6	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

Nota:

1. El tipo federado es TIMESTAMP(0) si el parámetro de configuración date_compat está establecido en ON.

Correlaciones de tipos de datos inversas por omisión

Para la mayor parte de fuentes de datos, las correlaciones de tipos por omisión se encuentran en los derivadores.

Existen dos tipos de correlaciones entre los tipos de datos de la fuente de datos y los tipos de datos de la base de datos federada: las correlaciones de tipos directas y las correlaciones de tipos inversas. En una correlación de tipos directa, la correlación se realiza desde un tipo remoto a un tipo local comparable. El otro tipo de correlación es una correlación de tipos inversa, que se utiliza con DDL transparente para crear o modificar tablas remotas.

Las correlaciones de tipos por omisión para fuentes de datos de la familia DB2 se encuentran en el derivador de DRDA. Las correlaciones de tipos por omisión para Informix se encuentran en el derivador de INFORMIX, y así sucesivamente.

Cuando se define una tabla o una vista remota en la base de datos federada, la definición incluye una correlación de tipos inversa. La correlación se realiza desde un tipo de datos de base de datos federada local para cada columna, y el correspondiente tipo de datos remoto. Por ejemplo, existe una correlación de tipos inversa por omisión en la que el tipo local REAL hace referencia al tipo SMALLFLOAT de Informix.

Las bases de datos federadas no dan soporte a las correlaciones para los tipos LONG VARCHAR, LONG VARGRAPHIC y para los tipos definidos por el usuario.

Cuando se utiliza la sentencia CREATE TABLE para crear una tabla remota, deben especificarse los tipos de datos locales que se desea incluir en la tabla remota. Estas correlaciones de tipos inversas por omisión asignarán los tipos remotos correspondientes a estas columnas. Por ejemplo, suponga que se utiliza la sentencia CREATE TABLE para definir una tabla Informix con una columna C2. Se especifica BIGINT como tipo de datos para C2 en la sentencia. La correlación de tipos inversa por omisión de BIGINT depende de la versión de Informix en la que esté creando la tabla. La correlación para C2 en la tabla Informix se establecerá con DECIMAL en Informix Versión 8 y con INT8 en Informix Versión 9.

Puede alterar temporalmente una correlación de tipos inversa por omisión o bien crear una nueva correlación de tipos inversa con la sentencia CREATE TYPE MAPPING.

En las tablas siguientes se muestran las correlaciones inversas por omisión entre tipos de datos locales de base de datos federada y tipos de datos de fuente de datos remota.

Estas correlaciones son válidas con todas las versiones soportadas, a menos que se indique lo contrario.

Correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 Database para Linux, UNIX y Windows

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 Database para Linux, UNIX y Windows.

Tabla 229. Correlaciones de tipos de datos inversas por omisión de DB2 Database para Linux, UNIX y Windows (no se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit federados
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE[1]	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DECFLOAT [2]	-	8	-	-	-	-	DECFLOAT	-	0	-
DECFLOAT [2]	-	16	-	-	-	-	DECFLOAT	-	0	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP(*p*)	-	*p*[3]	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Nota:

1. Cuando el parámetro date_compat está establecido en OFF, la DATE federada se correlaciona con TIMESTAMP(0).
2. Por omisión, la opción de servidor SAME_DECFLT_ROUNDING está establecida en N y ninguna operación se impulsará a la fuente de datos remota a menos que SAME_DECFLT_ROUNDING esté establecida en Y. Para obtener información sobre la opción de servidor SAME_DECFLT_ROUNDING, consulte la referencia de opciones de bases de datos DB2.
3. Para la Versión 9.5 o anteriores, la escala remota para TIMESTAMP es 6.

Correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para System i

La tabla siguiente lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para System i.

Tabla 230. Correlaciones de tipos de datos inversas por omisión para DB2 para System i (No se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operaciones datos feder.	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

Correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para VM y VSE

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para VM y VSE.

Tabla 231. Correlaciones de tipos de datos inversas por omisión de DB2 para VM y VSE (no se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP*(p)*	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

Correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para z/OS

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos DB2 para z/OS.

Tabla 232. Correlaciones de tipos de datos inversas por omisión de DB2 para z/OS (no se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP(*p*)	-	-	*p*	*p*	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Informix

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos Informix.

Tabla 233. Correlaciones de tipos de datos inversas por omisión para Informix

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR[1]	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

Nota:

1. Esta correlación de tipos sólo es válida con el servidor Informix Versión 8 (o inferior).
2. Esta correlación de tipos sólo es válida con el servidor Informix Versión 9 (o superior).

Para el tipo de datos DATETIME de Informix, el servidor federado utiliza el calificador de alto nivel de Informix como REMOTE_LENGTH y el calificador de bajo nivel de Informix como REMOTE_SCALE.

Los calificadores de Informix son las constantes ″TU_″ definidas en el archivo `datatime.h` del SDK del cliente Informix. Las constantes son:

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Microsoft SQL Server

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos Microsoft SQL Server.

Tabla 234. Correlaciones de tipos de datos inversas por omisión de Microsoft SQL Server (no se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

Nota:

1. Esta correlación de tipos sólo es válida con Microsoft SQL Server Versión 2000.

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Oracle NET8

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos Oracle NET8.

Tabla 235. Correlaciones de tipos de datos inversas por omisión para Oracle NET8

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BIGINT	0	8	0	0	N	\0	NUMBER	19	0	N
BLOB	0	2147483647	0	0	Y	\0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	\0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	\0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	\0	CLOB	0	0	N
DATE[1]	0	4	0	0	N	\0	DATE	0	0	N
DECIMAL	0	0	0	0	N	\0	NUMBER	0	0	N
DECFLOAT	0	8	0	0	N	\0	NUMBER	0	0	N
DECFLOAT	0	16	0	0	N	\0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	\0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	\0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	\0	NUMBER	10	0	N
REAL	0	4	0	0	N	\0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	\0	NUMBER	5	0	N
TIME	0	3	0	0	N	\0	DATE	0	0	N
TIMESTAMP [2]	0	10	0	0	N	\0	DATE	0	0	N
TIMESTAMP*(p)* [3]	-	-	-	-	N	\0	TIMESTAMP*(p)*	-	-	N
VARCHAR	1	4000	0	0	N	\0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	\0	RAW	0	0	Y

Nota:

1. Cuando el parámetro date_compat está establecido en OFF, la DATE federada se correlaciona con el dato Oracle. Cuando el parámetro date_compat está establecido en ON, la DATE federada (equivalente a TIMESTAMP(0)), se correlaciona con la TIMESTAMP(0) de Oracle.
2. Esta correlación de tipos sólo es válida con Oracle Versión 8.
3.
   - TIMESTAMP*(p)* representa una indicación de fecha y hora con una escala variable de 0-9 para Oracle y 0-12 para la federación. Cuando la escala es 0-9, la TIMESTAMP remota de Oracle tiene la misma escala que la TIMESTAMP federada. Si la escala de la TIMESTAMP federada es superior a 9, la escala correspondiente de la TIMESTAMP de Oracle es 9, que es la mayor escala de Oracle.
   - Esta correlación de tipos sólo es válida con Oracle Versión 9, 10 y 11.

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Sybase

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos Sybase.

Tabla 236. Correlaciones de tipos de datos inversas por omisión para Sybase CTLIB

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

Nota:

1. Esta correlación de tipos solo es válida para CTLIB con el servidor Sybase versión 12.0 (o anterior).
2. Esta correlación de tipos solo es válida para CTLIB con el servidor Sybase versión 12.5 (o posterior).

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Teradata

La siguiente tabla lista las correlaciones de tipos de datos inversas por omisión para fuentes de datos Teradata.

Tabla 237. Correlaciones de tipos de datos inversas por omisión de Teradata (no se muestran todas las columnas)

Nombre tipo federado	Long. inf. federada	Long. sup. federada	Escala inf. federada	Escala sup. federada	Datos bit federados	Operad. datos federados	Nombre tipo remoto	Long. remota	Escala remota	Datos bit remotos
BLOB	1	2097088000	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2097088000	-	-		-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [1]	1	64000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	8	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	8	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	15	-	-
TIMESTAMP	10	10	6	6	-	-	TIMESTAMP	26	6	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

Nota:

1. El tipo de datos VARGRAPHIC de Teradata sólo puede contener la longitud especificada (1 a 32000) de un tipo de datos DBCLOB.

Correlaciones de tipos de datos inversas por omisión para fuentes de datos Teradata

Apéndice F. La base de datos SAMPLE

Existen varios motivos por los que utilizar la base de datos de ejemplo, como por ejemplo, para probar las aplicaciones, intentar poner en práctica distintas características de DB2, etc. La mayoría de los programas de aplicación de ejemplo bajo DB2PATH/sqllib/samples utilizan la base de datos de ejemplo para demostrar algunas de las características de DB2 de forma que sea más fácil entenderlas.

Una vez que se haya creado la base de datos de DB2, observará que:

- Se crea un esquema organizativo para datos que no son de XML y
- y un esquema de órdenes de compra para datos XML.

Los objetos de base de datos y datos bajo estos esquemas se crean utilizando un entorno de tiempo real a pequeña escala.

A continuación se proporciona una descripción de cada una de las tablas de la base de datos SAMPLE. Se proporcionan los valores de datos iniciales para cada tabla; un guión (-) indica un valor NULL (nulo).

Tabla ACT

Nombre:	ACTNO	ACTKWD	ACTDESC
Tipo:	SMALLINT	CHAR(6)	VARCHAR(20)
Valores:	10	MANAGE	MANAGE/ADVISE
	20	ECOST	ESTIMATE COST
	30	DEFINE	DEFINE SPECS
	40	LEADPR	LEAD PROGRAM/DESIGN
	50	SPECS	WRITE SPECS
	60	LOGIC	DESCRIBE LOGIC
	70	CODE	CODE PROGRAMS
	80	TEST	TEST PROGRAMS
	90	ADMQS	ADM QUERY SYSTEM
	100	TEACH	TEACH CLASSES
	110	COURSE	DEVELOP COURSES
	120	STAFF	PERS AND STAFFING
	130	OPERAT	OPER COMPUTER SYS
	140	MAINT	MAINT SOFTWARE SYS
	150	ADMSYS	ADM OPERATING SYS
	160	ADMDB	ADM DATA BASES
	170	ADMDC	ADM DATA COMM
	180	DOC	DOCUMENT

Tabla ADEFUSER

Nombre:	WORKDEPT	NO_OF_EMPLOYEES
Tipo:	CHAR(3)	INTEGER
Valores:	A00	5
	B01	1
	C01	4
	D11	11
	D21	7
	E01	1
	E11	7
	E21	6

Tabla CL_SCHED

Nombre:	CLASS_CODE	DAY	STARTING	ENDING
Tipo:	CHAR(7)	SMALLINT	TIME	TIME
Desc:	Código de clase (aula:profesor)	Día núm de programación de 4 días	Hora de inicio de la clase	Hora de finalización de la clase
Valores:	042:BF	4	12:10:00	14:00:00
	553:MJA	1	10:30:00	11:00:00
	543:CWM	3	09:10:00	10:30:00
	778:RES	2	12:10:00	14:00:00
	044:HD	3	17:12:30	18:00:00

Tabla DEPT

Nombre:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Tipo:	CHAR(3)	VARCHAR(36)	CHAR(6)	CHAR(3)	CHAR(16)
Valores:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	C01	INFORMATION CENTER	000030	A00	
	D01	DEVELOPMENT CENTER		A00	
	D11	MANUFACTURING SYSTEMS	000060	D01	
	D21	ADMINISTRATION SYSTEMS	000070	D01	
	E01	SUPPORT SERVICES	000050	A00	
	E11	OPERATIONS	000090	E01	
	E21	SOFTWARE SUPPORT	000100	E01	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

Tabla DEPARTMENT

Nombre:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Tipo:	CHAR(3) No nulo	VARCHAR(29) No nulo	CHAR(6)	CHAR(3) No nulo	CHAR(16)
Desc:	Número de departamento	Nombre que describe las actividades generales del departamento	Número de empleado (EMPNO) del jefe de departamento	Departamento (DEPTNO) del que depende este departamento	Nombre de la ubicación remota.
Valores:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	C01	INFORMATION CENTER	000030	A00	
	D01	DEVELOPMENT CENTER		A00	
	D11	MANUFACTURING SYSTEMS	000060	D01	
	D21	ADMINISTRATION SYSTEMS	000070	D01	
	E01	SUPPORT SERVICES	000050	A00	
	E11	OPERATIONS	000090	E01	
	E21	SOFTWARE SUPPORT	000100	E01	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

Tablas EMPLOYEE y EMP

Estas dos tablas tienen un contenido idéntico.

Nombres:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
Tipo:	CHAR(6) No nulo	VARCHAR(12) No nulo	CHAR(1) No nulo	VARCHAR(15) No nulo	CHAR(3)	CHAR(4)	DATE
Desc:	Número de empleado	Nombre propio	Inicial del segundo nombre	Apellido	Departamento (DEPTNO) en el que trabaja el empleado	Número de teléfono	Fecha de contratación

+

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(8)	SMALLINT No nulo	CHAR(1)	DATE	DECIMAL(9,2)	DECIMAL(9,2)	DECIMAL(9,2)
Trabajo	Número de años de educación formal	Sexo (V varón, M mujer)	Fecha de nacimiento	Salario anual	Plus anual	Comisión anual

La tabla siguiente contiene los valores de la tabla EMPLOYEE.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(6) No nulo	VARCHAR(12) No nulo	CHAR(1) No nulo	VARCHAR(15) No nulo	CHAR(3)	CHAR(4)	DATE	CHAR(8)	SMALLINT No nulo	CHAR(1)	DATE	DECIMAL (9,2)	DECIMAL (9,2)	DECIMAL (9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

Tabla EMP_ACT

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
Tipo:	CHAR(6) No nulo	CHAR(6) No nulo	SMALLINT No nulo	DEC(5,2)	DATE	DATE
Desc:	Número de empleado	Número de proyecto	Número de actividad	Proporción del tiempo de empleado que se ha invertido en el proyecto	Fecha de inicio de la actividad	Fecha de finalización de la actividad
Valores:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

Tabla EMP_PHOTO

Nombre:	EMPNO	PHOTO_FORMAT	PICTURE
Tipo:	CHAR(6) No nulo	VARCHAR(10) No nulo	BLOB(100K)
Desc:	Número de empleado	Formato de foto	Foto del empleado
Valores:	000130	mapa de bits	db200130.bmp
	000130	gif	db200130.gif
	000140	mapa de bits	db200140.bmp
	000140	gif	db200140.gif
	000150	mapa de bits	db200150.bmp
	000150	gif	db200150.gif
	000190	mapa de bits	db200190.bmp
	000190	gif	db200190.gif

Tabla EMPPROJACT

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
Tipo:	CHAR(6)	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
Valores:	000070	AD3110	10	1.00	01/01/1982	02/01/1983
	000230	AD3111	60	1.00	01/01/1982	03/15/1982
	000230	AD3111	60	0.50	03/15/1982	04/15/1982
	000230	AD3111	70	0.50	03/15/1982	10/15/1982
	000230	AD3111	80	0.50	04/15/1982	10/15/1982
	000230	AD3111	180	0.50	10/15/1982	01/01/1983
	000240	AD3111	70	1.00	02/15/1982	09/15/1982
	000240	AD3111	80	1.00	09/15/1982	01/01/1983
	000250	AD3112	60	1.00	01/01/1982	02/01/1982
	000250	AD3112	60	0.50	02/01/1982	03/15/1982
	000250	AD3112	60	1.00	01/01/1983	02/01/1983
	000250	AD3112	70	0.50	02/01/1982	03/15/1982
	000250	AD3112	70	1.00	03/15/1982	08/15/1982
	000250	AD3112	70	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.50	10/15/1982	12/01/1982
	000250	AD3112	180	0.50	08/15/1982	01/01/1983
	000260	AD3113	70	0.50	06/15/1982	07/01/1982
	000260	AD3113	70	1.00	07/01/1982	02/01/1983
	000260	AD3113	80	1.00	01/01/1982	03/01/1982
	000260	AD3113	80	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	1.00	04/15/1982	06/01/1982
	000260	AD3113	180	1.00	06/01/1982	07/01/1982
	000270	AD3113	60	0.50	03/01/1982	04/01/1982
	000270	AD3113	60	1.00	04/01/1982	09/01/1982
	000270	AD3113	60	0.25	09/01/1982	10/15/1982
	000270	AD3113	70	0.75	09/01/1982	10/15/1982
	000270	AD3113	70	1.00	10/15/1982	02/01/1983
	000270	AD3113	80	1.00	01/01/1982	03/01/1982
	000270	AD3113	80	0.50	03/01/1982	04/01/1982
	000030	IF1000	10	0.50	06/01/1982	01/01/1983
	000130	IF1000	90	1.00	10/01/1982	01/01/1983
	000130	IF1000	100	0.50	10/01/1982	01/01/1983
	000140	IF1000	90	0.50	10/01/1982	01/01/1983
	000030	IF2000	10	0.50	01/01/1982	01/01/1983
	000140	IF2000	100	1.00	01/01/1982	03/01/1982
	000140	IF2000	100	0.50	03/01/1982	07/01/1982

Nombre:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000140	IF2000	110	0.50	03/01/1982	07/01/1982
	000140	IF2000	110	0.50	10/01/1982	01/01/1983
	000010	MA2100	10	0.50	01/01/1982	11/01/1982
	000110	MA2100	20	1.00	01/01/1982	03/01/1983
	000010	MA2110	10	1.00	01/01/1982	02/01/1983
	000200	MA2111	50	1.00	01/01/1982	06/15/1982
	000200	MA2111	60	1.00	06/15/1982	02/01/1983
	000220	MA2111	40	1.00	01/01/1982	02/01/1983
	000150	MA2112	60	1.00	01/01/1982	07/15/1982
	000150	MA2112	180	1.00	07/15/1982	02/01/1983
	000170	MA2112	60	1.00	01/01/1982	06/01/1983
	000170	MA2112	70	1.00	06/01/1982	02/01/1983
	000190	MA2112	70	1.00	01/01/1982	10/01/1982
	000190	MA2112	80	1.00	10/01/1982	10/01/1982
	000160	MA2113	60	1.00	07/15/1982	02/01/1983
	000170	MA2113	80	1.00	01/01/1982	02/01/1983
	000180	MA2113	70	1.00	04/01/1982	06/15/1982
	000210	MA2113	80	0.50	10/01/1982	02/01/1983
	000210	MA2113	180	0.50	10/01/1982	02/01/1983
	000050	OP1000	10	0.25	01/01/1982	02/01/1983
	000090	OP1010	10	1.00	01/01/1982	02/01/1983
	000280	OP1010	130	1.00	01/01/1982	02/01/1983
	000290	OP1010	130	1.00	01/01/1982	02/01/1983
	000300	OP1010	130	1.00	01/01/1982	02/01/1983
	000310	OP1010	130	1.00	01/01/1982	02/01/1983
	000050	OP1010	10	0.75	01/01/1982	02/01/1983
	000100	OP1010	10	1.00	01/01/1982	02/01/1983
	000320	OP2011	140	0.75	01/01/1982	02/01/1983
	000320	OP2011	150	0.25	01/01/1982	02/01/1983
	000330	OP2012	140	0.25	01/01/1982	02/01/1983
	000330	OP2012	160	0.75	01/01/1982	02/01/1983
	000340	OP2013	140	0.50	01/01/1982	02/01/1983
	000340	OP2013	170	0.50	01/01/1982	02/01/1983
	000020	PL2100	30	1.00	01/01/1982	09/15/1982

Tabla EMP_RESUME

Nombre:	EMPNO	RESUME_FORMAT	RESUME
Tipo:	CHAR(6) No nulo	VARCHAR(10) No nulo	CLOB(5K)
Desc:	Número de empleado	Formato del currículum	Currículum del empleado
Valores:	000130	ascii	db200130.asc

Nombre:	EMPNO	RESUME_FORMAT	RESUME
	000130	html	db200130.htm
	000140	ascii	db200140.asc
	000140	html	db200140.htm
	000150	ascii	db200150.asc
	000150	html	db200150.htm
	000190	ascii	db200190.asc
	000190	html	db200190.htm

Tabla IN_TRAY

Nombre:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
Tipo:	TIMESTAMP	CHAR(8)	CHAR(64)	VARCHAR(3000)
Desc:	Fecha y hora recibida	ID de usuario de la persona que envía la nota	Descripción breve	La nota
	1988-12-25-17.12.30.000000	BADAMSON	FWD: ¡Un año fantástico! Plus del 4º trimestre.	A: JWALKER Cc: QUINTANA, NICHOLLS Jim, Parece que el trabajo duro ha dado resultado. Tengo unas buenas cervezas en la nevera si desean celebrarlo. Delores y Heather, ¿Estáis también interesadas? Bruce <Reenvío desde ISTERN> Tema: FWD: ¡Año fantástico! Plus del 4º trimestre. Ao: Dept_D11 Felicidades por un trabajo bien hecho. Disfruten del plus de este año. Irv <Reenvío desde CHAAS> Tema: ¡Un año fantástico! Plus del 4º trimestre. A: Todos_los_jefes_de_departamento Han llegado los resultados del 4º trimestre. ¡Hemos cooperado como equipo y hemos superado lo planificado! Me complace anunciar un plus este año del 18%. Disfruten las vacaciones. Christine Haas

Nombre:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
	1988-12-23-08.53.58.000000	ISTERN	FWD: ¡Un año fantástico! Plus del 4º trimestre.	Ao: Dept_D11 Felicidades por un trabajo bien hecho. Disfruten del plus de este año. Irv <Reenvío desde CHAAS> Tema: ¡Un año fantástico! Plus del 4º trimestre. A: Todos_los_jefes_de_departamento Han llegado los resultados del 4º trimestre. ¡Hemos cooperado como equipo y hemos superado lo planificado! Me complace anunciar un plus este año del 18%. Disfruten las vacaciones. Christine Haas
	1988-12-22-14.07.21.136421	CHAAS	¡Un año fantástico! Plus del 4º trimestre.	A: Todos_los_jefes_de_departamento Han llegado los resultados del 4º trimestre. ¡Hemos cooperado como equipo y hemos superado lo planificado! Me complace anunciar un plus este año del 18%. Disfruten las vacaciones. Christine Haas

Tabla ORG

Nombre:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
Tipo:	SMALLINT No nulo	VARCHAR(14)	SMALLINT	VARCHAR(10)	VARCHAR(13)
Desc:	Número de departamento	Nombre del departamento	Número del jefe de departamento	División corporativa	Ciudad
Valores:	10	Oficina principal	160	Corporativo	Nueva York
	15	Nueva Inglaterra	50	Costa Este	Boston
	20	Atlántico Medio	10	Costa Este	Washington
	38	Atlántico Sur	30	Costa Este	Atlanta
	42	Grandes lagos	100	Medio Oeste	Chicago
	51	Grandes llanos	140	Medio Oeste	Dallas
	66	Pacífico	270	Oeste	San Francisco
	84	Montañas	290	Oeste	Denver

Tabla PROJ

Nombre:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
Tipo:	CHAR(6)	VARCHAR(36)	CHAR(3)	CHAR(6)	DEC(5,2)	DATE	DATE	CHAR(6)
Valores:	AD3100	ADMIN SERVICES	D01	000010	6.50	01/01/1982	02/01/1983	
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6.00	01/01/1982	02/01/1983	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2.00	01/01/1982	02/01/1983	AD3100
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1.00	01/01/1982	02/01/1983	AD3100
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2.00	01/01/1982	02/01/1983	AD3100
	IF1000	QUERY SERVICES	C01	000030	2.00	01/01/1982	02/01/1983	
	IF2000	USER EDUCATION	C01	000030	1.00	01/01/1982	02/01/1983	
	MA2100	WELD LINE AUTOMATION	D01	000010	12.00	01/01/1982	02/01/1983	
	MA2110	W L PROGRAMMING	D11	000060	9.00	01/01/1982	02/01/1983	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2.00	01/01/1982	12/01/1982	MA2100
	MA2112	W L ROBOT DESIGN	D11	000150	3.00	01/01/1982	12/01/1982	MA2100
	MA2113	W L PROD CONT PROGS	D11	000160	3.00	02/15/1982	12/01/1982	MA2100
	OP1000	OPERATION SUPPORT	E01	000050	6.00	01/01/1982	02/01/1983	
	OP1010	OPERATION	E11	000090	5.00	01/01/1982	02/01/1983	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5.00	01/01/1982	02/01/1983	
	OP2010	SYSTEMS SUPPORT	E21	000100	4.00	01/01/1982	02/01/1983	OP2010
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1.00	01/01/1982	02/01/1983	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1.00	01/01/1982	02/01/1983	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1.00	01/01/1982	02/01/1983	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1.00	01/01/1982	09/15/1982	MA2100

Tabla PROJACT

Nombre:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
Tipo:	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
Valores:	AD3100	10		01/01/1982	
	AD3110	10		01/01/1982	
	AD3111	60		01/01/1982	
	AD3111	60		03/15/1982	
	AD3111	70		03/15/1982	
	AD3111	80		04/15/1982	
	AD3111	180		10/15/1982	
	AD3111	70		02/15/1982	

Nombre:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	AD3111	80		09/15/1982	
	AD3112	60		01/01/1982	
	AD3112	60		02/01/1982	
	AD3112	60		01/01/1983	
	AD3112	70		02/01/1982	
	AD3112	70		03/15/1982	
	AD3112	70		08/15/1982	
	AD3112	80		08/15/1982	
	AD3112	80		10/15/1982	
	AD3112	180		08/15/1982	
	AD3113	70		06/15/1982	
	AD3113	70		07/01/1982	
	AD3113	80		01/01/1982	
	AD3113	80		03/01/1982	
	AD3113	180		03/01/1982	
	AD3113	180		04/15/1982	
	AD3113	180		06/01/1982	
	AD3113	60		03/01/1982	
	AD3113	60		04/01/1982	
	AD3113	60		09/01/1982	
	AD3113	70		09/01/1982	
	AD3113	70		10/15/1982	
	IF1000	10		06/01/1982	
	IF1000	90		10/01/1982	
	IF1000	100		10/01/1982	
	IF2000	10		01/01/1982	
	IF2000	100		01/01/1982	
	IF2000	100		03/01/1982	
	IF2000	110		03/01/1982	
	IF2000	110		10/01/1982	
	MA2100	10		01/01/1982	
	MA2100	20		01/01/1982	
	MA2110	10		01/01/1982	
	MA2111	50		01/01/1982	
	MA2111	60		06/15/1982	
	MA2111	40		01/01/1982	
	MA2112	60		01/01/1982	
	MA2112	180		07/15/1982	
	MA2112	70		06/01/1982	
	MA2112	70		01/01/1982	
	MA2112	80		10/01/1982	

Nombre:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	MA2113	60		07/15/1982	
	MA2113	80		01/01/1982	
	MA2113	70		04/01/1982	
	MA2113	80		10/01/1982	
	MA2113	180		10/01/1982	
	OP1000	10		01/01/1982	
	OP1010	10		01/01/1982	
	OP1010	130		01/01/1982	
	OP2010	10		01/01/1982	
	OP2011	140		01/01/1982	
	OP2011	150		01/01/1982	
	OP2012	140		01/01/1982	
	OP2012	160		01/01/1982	
	OP2013	140		01/01/1982	
	OP2013	170		01/01/1982	
	PL2100	30		01/01/1982	

Tabla PROJECT

Nombre:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
Tipo:	CHAR(6) No nulo	VARCHAR(24) No nulo	CHAR(3) No nulo	CHAR(6) No nulo	DEC(5,2)	DATE	DATE	CHAR(6)
Desc:	Número de proyecto	Nombre del proyecto	Departamento responsable	Empleado responsable	Dotación media de personal estimada	Fecha de inicio estimada	Fecha de finalización estimada	Proyecto importante, para un subproyecto
Valores:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Tabla SALES

Nombre:	SALES_DATE	SALES_PERSON	REGION	SALES
Tipo:	DATE	VARCHAR (15)	VARCHAR (15)	INTEGER
Desc:	Fecha de venta	Apellido del empleado	Región de venta	Número de venta
Valores:	12/31/2005	LUCCHESSI	Ontario-Sur	1
	12/31/2005	LEE	Ontario-Sur	3
	12/31/2005	LEE	Quebec	1
	12/31/2005	LEE	Manitoba	2
	12/31/2005	GOUNOT	Quebec	1
	03/29/2006	LUCCHESSI	Ontario-Sur	3
	03/29/2006	LUCCHESSI	Quebec	1
	03/29/2006	LEE	Ontario-Sur	2
	03/29/1996	LEE	Ontario-Norte	2
	03/29/2006	LEE	Quebec	3
	03/29/2006	LEE	Manitoba	5
	03/29/2006	GOUNOT	Ontario-Sur	3
	03/29/2006	GOUNOT	Quebec	1
	03/29/2006	GOUNOT	Manitoba	7
	03/30/2006	LUCCHESSI	Ontario-Sur	1
	03/30/2006	LUCCHESSI	Quebec	2
	03/30/2006	LUCCHESSI	Manitoba	1
	03/30/2006	LEE	Ontario-Sur	7
	03/30/2006	LEE	Ontario-Norte	3
	03/30/2006	LEE	Quebec	7
	03/30/2006	LEE	Manitoba	4
	03/30/2006	GOUNOT	Ontario-Sur	2
	03/30/2006	GOUNOT	Quebec	18
	03/30/2006	GOUNOT	Manitoba	1
	03/31/2006	LUCCHESSI	Manitoba	1
	03/31/2006	LEE	Ontario-Sur	14
	03/31/2006	LEE	Ontario-Norte	3
	03/31/2006	LEE	Quebec	7
	03/31/2006	LEE	Manitoba	3
	03/31/2006	GOUNOT	Ontario-Sur	2
	03/31/2006	GOUNOT	Quebec	1
	04/01/2006	LUCCHESSI	Ontario-Sur	3
	04/01/2006	LUCCHESSI	Manitoba	1
	04/01/2006	LEE	Ontario-Sur	8
	04/01/2006	LEE	Ontario-Norte	-
	04/01/2006	LEE	Quebec	8
	04/01/2006	LEE	Manitoba	9

Nombre:	SALES_DATE	SALES_PERSON	REGION	SALES
	04/01/2006	GOUNOT	Ontario-Sur	3
	04/01/2006	GOUNOT	Ontario-Norte	1
	04/01/2006	GOUNOT	Quebec	3
	04/01/2006	GOUNOT	Manitoba	7

Tabla STAFF

Nombre:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Tipo:	SMALLINT No nulo	VARCHAR(9)	SMALLINT	CHAR(5)	SMALLINT	DECIMAL(7,2)	DECIMAL(7,2)
Desc:	Número de empleado	Nombre del empleado	Número de departamento	Tipo de trabajo	Años de servicio	Salario actual	Comisión
Valores:	10	Sanders	20	Dir	7	18357,50	-
	20	Pernal	20	Ventas	8	18171,25	612,45
	30	Marenghi	38	Dir	5	17506,75	-
	40	O'Brien	38	Ventas	6	18006,00	846,55
	50	Hanes	15	Dir	10	20659,80	-
	60	Quigley	38	Ventas	-	16808,30	650,25
	70	Rothman	15	Ventas	7	16502,83	1152,00
	80	James	20	Oficinista	-	13504,60	128,20
	90	Koonitz	42	Ventas	6	18001,75	1386,70
	100	Plotz	42	Dir	7	18352,80	-
	110	Ngan	15	Oficinista	5	12508,20	206,60
	120	Naughton	38	Oficinista	-	12954,75	180,00
	130	Yamaguchi	42	Oficinista	6	10505,90	75,60
	140	Fraye	51	Dir	6	21150,00	-
	150	Williams	51	Ventas	6	19456,50	637,65
	160	Molinare	10	Dir	7	22959,20	-
	170	Kermisch	15	Oficinista	4	12258,50	110,10
	180	Abrahams	38	Oficinista	3	12009,75	236,50
	190	Sneider	20	Oficinista	8	14252,75	126,50
	200	Scoutten	42	Oficinista	-	11508,60	84,20
	210	Lu	10	Dir	10	20010,00	-
	220	Smith	51	Ventas	7	17654,50	992,80
	230	Lundquist	51	Oficinista	3	13369,80	189,65
	240	Daniels	10	Dir	5	19260,25	-
	250	Wheeler	51	Oficinista	6	14460,00	513,30
	260	Jones	10	Dir	12	21234,00	-
	270	Lea	66	Dir	9	18555,50	-
	280	Wilson	66	Ventas	9	18674,50	811,50
	290	Quill	84	Dir	10	19818,00	-
	300	Davis	84	Ventas	5	15454,50	806,10
	310	Graham	66	Ventas	13	21000,00	200,30
	320	Gonzales	66	Ventas	4	16858,20	844,00
	330	Burke	66	Oficinista	1	10988,00	55,50
	340	Edwards	84	Ventas	7	17844,00	1285,00
	350	Gafney	84	Oficinista	5	13030,50	188,00

Tabla PRODUCT

Nombre:	PID	NAME	PRICE	PROMOPRICE	PROMOSTART	PROMOEND	DESCRIPTION
Tipo:	VARCHAR(10) No nulo	VARCHAR(128)	DECIMAL(30,2)	DECIMAL(30,2)	DATE	DATE	XML
Valores:	100-100-01	Pala para la nieve, modelo base de 22 pulgadas	9,99	7,25	11/19/2004	12/19/2004	`p1.xml`
	100-101-01	Pala para la nieve, modelo de lujo de 24 pulgadas	19,99	15,99	12/18/2005	02/28/2006	`p2.xml`
	100-103-01	Pala para la nieve, modelo de superlujo de 26 pulgadas	49,99	39,99	12/22/2005	02/22/2006	`p3.xml`
	100-201-01	Espátula para el hielo, Parabrisas de 4 pulgadas	3,99	--	--	--	`p4.xml`

A continuación se facilita el archivo de definición de esquemas de XML para la columna XML de la tabla anterior: `product.xsd`

Tabla PURCHASEORDER

Nombre:	POID	STATUS	CUSTID	ORDERDATE	PORDER	COMMENTS
Tipo:	BIGINT No nulo	VARCHAR(10) No nulo	BIGINT	DATE	XML	VARCHAR(1000)
Valores:	5000	No enviado	1002	02/18/2006	po1.xml	THIS IS A NEW PURCHASE ORDER
	5001	Enviado	1003	02/03/2005	po2.xml	THIS IS A NEW PURCHASE ORDER
	5002	Enviado	1001	02/29/2004	po3.xml	THIS IS A NEW PURCHASE ORDER
	5003	Enviado	1002	02/28/2005	po4.xml	THIS IS A NEW PURCHASE ORDER
	5004	Enviado	1005	11/18/2005	po5.xml	THIS IS A NEW PURCHASE ORDER
	5006	Enviado	1002	03/01/2006	po6.xml	THIS IS A NEW PURCHASE ORDER

A continuación se facilita el archivo de definición de esquemas de XML para la columna XML de la tabla anterior: `porder.xsd`

Tabla CUSTOMER

Nombre:	CID	INFO
Tipo:	BIGINT No nulo	XML
Valores:	1000	c1.xml
	1001	c2.xml
	1002	c3.xml
	1003	c4.xml
	1004	c5.xml
	1005	c6.xml

A continuación se facilita el archivo de definición de esquemas de XML para la columna XML de la tabla anterior: `customer.xsd`

Tabla CATALOG

Nombre:	NAME	CATALOG
Tipo:	VARCHAR(128) No nulo	XML
Valores:	Catálogo de primavera	cat1.xml
	Catálogo de verano	cat2.xml
	Catálogo de otoño	cat3.xml
	Catálogo de invierno	cat4.xml

A continuación se facilita el archivo de definición de esquemas de XML para la columna XML de la tabla anterior: `catalog.xsd`

Tabla INVENTORY

Nombre:	PID	QUANTITY	LOCATION
Tipo:	VARCHAR(10) No nulo	INTEGER	VARCHAR(128)
Valores:	100-100-01	5	--
	100-101-01	25	Tienda
	100-103-01	55	Tienda

Nombre:	PID	QUANTITY	LOCATION
	100-201-01	99	Almacén

Tabla PRODUCTSUPPLIER

Nombre:	PID	SID
Tipo:	VARCHAR(10) No nulo	VARCHAR(10) No nulo
Valores:	100-100-01	123-456-78
	100-101-01	123-456-78
	100-103-01	555-789-00
	100-201-01	989-897-23

Tabla SUPPLIERS

Nombre:	SID	ADDR
Tipo:	VARCHAR(10) No nulo	XML
Valores:	123-456-78	`s1.xml`
	555-789-00	`s2.xml`
	989-897-23	`s3.xml`
	111-898-45	`s4.xml`

A continuación se facilita el archivo de definición de esquemas de XML para la columna XML de la tabla anterior: `supplier.xsd`

Archivos de ejemplo con el tipo de datos BLOB y CLOB

Esta sección muestra los datos que se han encontrado en los archivos EMP_PHOTO (fotos de empleados) y en los archivos EMP_RESUME (currículums de empleados).

Foto de Quintana

Figura 14. Dolores M. Quintana

Currículum de Quintana

El texto siguiente se ha encontrado en el archivo db200130.asc.

Currículum: Dolores M. Quintana

Información personal

Dirección:
1150 Eglinton Ave Mellonville, Idaho 83725

Teléfono:
(208) 555-9933

Fecha de nacimiento:
15 de Setiembre de 1925

Sexo: Mujer

Estado civil:
Casada

Estatura:
1,58cm

Peso: 55 kg.

Información del departamento

Número de empleada:
000130

Número del departamento:
C01

Directora:
Sally Kwan

Empleo:
Analista

Teléfono:
(208) 555-4578

Fecha de contratación:
1971-07-28

Formación:

1965 Matemáticas e Inglés, Licenciatura Adelphi University

1960 Técnico Dental Florida Institute of Technology

Historial laboral

10/91 - presente
Analista de sistemas consultivos Producción de herramientas de documentación para el departamento de ingeniería.

12/85 - 9/91
Redactora técnica, Redactora, Programadora de textos y planificadora.

1/79 - 11/85
Programadora de nóminas en COBOL Escribir programas de nóminas para una empresa de venta de gasóleo.

Intereses

- Cocinar
- Leer
- Coser

- Remodelar

A continuación se muestra el contenido del archivo db200130.htm.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Currículum:  Delores M. Quintana
<!-- Begin Header Records  ========================================== -->
<!-- DB200130 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:23                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:33:57">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:33:57">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:33:57"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Currículum:  Delores M. Quintana<HR>
<H2><A NAME="ToC">Tabla de contenidos<A NAME="ToC_1" HREF="#Header_1">
Currículum:  Delores M. Quintana<BR>
<A NAME="ToC_2" HREF="#Header_2">Información personal<BR>
<A NAME="ToC_3" HREF="#Header_3">Información del departamento<BR>
<A NAME="ToC_4" HREF="#Header_4">Formación<BR>
<A NAME="ToC_5" HREF="#Header_5">Historial de trabajo<BR>
<A NAME="ToC_6" HREF="#Header_6">Intereses<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Currículum:  Delores M. Quintana<P>
<H3><A NAME="Header_2">Información personal<DL COMPACT>
<DT>Dirección:
<DD>1150 Eglinton Ave
<BR>
Mellonville, Idaho 83725
<DT>Teléfono:
<DD>(208) 875-9933
<DT>Fecha de nacimiento:
<DD>15 de Setiembre de 1925
<DT>Sexo:
<DD>Mujer
<DT>Estado civil:
<DD>Casada
<DT>Estatura:
<DD>1,58 cm
<DT>Peso:
<DD>55 kg.<P>
<H3><A NAME="Header_3">Información del departamento<DL COMPACT>
<DT>Número de empleada:
<DD>000130
<DT>Número del departamento:
<DD>C01
<DT>Directora:
<DD>Sally Kwan
<DT>Empleo:
<DD>Analista
<DT>Teléfono:
<DD>(208) 385-4578
<DT>Fecha de contratación:
<DD>1971-07-28<P>
<H3><A NAME="Header_4">Formación<DL>
<P><DT>1965
<DD>Matemáticas e Inglés, Licenciatura
<BR>
Adelphi University
<P><DT>1960
<DD>Técnico Dental
<BR>
Florida Institute of Technology<P>
<H3><A NAME="Header_5">Historial laboral<DL>
<P><DT>10/91 - presente
<DD>Analista de sistemas consultivos
```

```
<BR>
Producción de herramientas de documentación para el departamento de ingeniería.
<P><DT>12/85 - 9/91
<DD>Redactora técnica
<BR>
Redactora, Programadora de textos y planificadora.
<P><DT>1/79 - 11/85
<DD>Programadora de nóminas en COBOL
<BR>
Escribir programas de nóminas para una empresa de venta de gasóleo.<P>
<H3><A NAME="Header_6">Intereses<UL COMPACT>
<LI>Cocinar
<LI>Leer
<LI>Coser
<LI>Remodelar<A NAME="Bot_Of_Page">
```

Foto de Nicholls

Figura 15. Heather A. Nicholls

Currículum de Nicholls

El texto siguiente se ha encontrado en el archivo db200140.asc.

Currículum: Heather A. Nicholls

Información personal

Dirección:
844 Don Mills Ave Mellonville, Idaho 83734

Teléfono:
(208) 555-2310

Fecha de nacimiento:
19 de Enero de 1946

Sexo: Mujer

Estado civil:
Soltera

Estatura:
1,73 cm.

Peso: 59 Kg.

Información del departamento

Número de empleada:
000140

Número del departamento:
C01

Directora:
Sally Kwan

Empleo:
Analista

Teléfono:
(208) 555-1793

Fecha de contratación:
1976-12-15

Formación:

1972 Ingeniera informática, Doctorado University of Washington

1969 Música y Física, Master Vassar College

Historial laboral

2/83 - presente
Diseño, Desarrollo de OCR, Diseño de la arquitectura de productos de OCR.

12/76 - 1/83
Programador de textos, Programación del reconocimiento de caracteres ópticos (OCR) en PL/I.

9/72 - 11/76
Analista de calidad de tarjetas perforadas Comprobar si las tarjetas perforadas satisfacen las especificaciones de calidad.

Intereses

- Ferrocarriles en miniatura
- Decorar interiores
- Bordar
- Hacer punto

A continuación se muestra el contenido del archivo db200140.htm.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Currículum:  Heather A. Nicholls
<!-- Begin Header Records  ========================================== -->
<!-- DB200140 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:17">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:17">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:17"><BODY>
<!-- End Header Records  ============================================ -->
<A NAME="Top_Of_Page"><H1>Currículum:  Heather A. Nicholls<HR>
<H2><A NAME="ToC">Tabla de contenidos<A NAME="ToC_1" HREF="#Header_1">
Currículum:  Heather A. Nicholls<BR>
<A NAME="ToC_2" HREF="#Header_2">Información personal<BR>
<A NAME="ToC_3" HREF="#Header_3">Información del departamento<BR>
<A NAME="ToC_4" HREF="#Header_4">Formación<BR>
<A NAME="ToC_5" HREF="#Header_5">Historial de trabajo<BR>
<A NAME="ToC_6" HREF="#Header_6">Intereses<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Currículum:  Heather A. Nicholls<P>
```

```
<H3><A NAME="Header_2">Información personal<DL COMPACT>
<DT>Dirección:
<DD>844 Don Mills Ave
<BR>
Mellonville, Idaho 83734
<DT>Teléfono:
<DD>(208) 610-2310
<DT>Fecha de nacimiento:
<DD>19 de Enero de 1946
<DT>Sexo:
<DD>Mujer
<DT>Estado civil:
<DD>Soltera
<DT>Estatura:
<DD>1,73 cm
<DT>Peso:
<DD>59 kg.<P>
<H3><A NAME="Header_3">Información del departamento<DL COMPACT>
<DT>Número de empleada:
<DD>000140
<DT>Número del departamento:
<DD>C01
<DT>Directora:
<DD>Sally Kwan
<DT>Empleo:
<DD>Analista
<DT>Teléfono:
<DD>(208) 385-1793
<DT>Fecha de contratación:
<DD>1976-12-15<P>
<H3><A NAME="Header_4">Formación<DL>
<P><DT>1972
<DD>Ingeniera informática, Doctorado
<BR>
University of Washington
<P><DT>1969
<DD>Música y Física, Licenciatura
<BR>
Vassar College<P>
<H3><A NAME="Header_5">Historial laboral<DL>
<P><DT>2/83 - presente
<DD>Diseño, Desarrollo de OCR
<BR>
Diseño de la arquitectura de productos de OCR.
<P><DT>12/76 - 1/83
<DD>Programadora de textos
<BR>
Programación del reconocimiento de caracteres ópticos (OCR) en PL/I.
<P><DT>9/72 - 11/76
<DD>Analista de calidad de tarjetas perforadas
<BR>
Comprobar si las tarjetas perforadas satisfacen las especificaciones de calidad.<P>
<H3><A NAME="Header_6">Intereses<UL COMPACT>
<LI>Ferrocarriles en miniatura
<LI>Decorar interiores
<LI>Bordar
<LI>Hacer punto<A NAME="Bot_Of_Page">
```

Foto de Adamson

Figura 16. Bruce Adamson

Currículum de Adamson

El texto siguiente se ha encontrado en el archivo db200150.asc.

Currículum: Bruce Adamson

Información personal

Dirección:
3600 Steeles Ave Mellonville, Idaho 83757

Teléfono:
(208) 555-4489

Fecha de nacimiento:
17 de Mayo de 1947

Sexo: Varón

Estado civil:
Casada

Estatura:
1,83 cm.

Peso: 83 kg.

Información del departamento

Número de empleada:
000150

Número del departamento:
D11

Directora:
Irving Stern

Empleo:
Diseñador

Teléfono:
(208) 555-4510

Fecha de contratación:
1972-02-12

Formación:

1971 Ingeniería Medioambiental, Master Johns Hopkins University

1968 Historia de América, Licenciatura, Northwestern University

Historial laboral

8/79 - presente
Diseño de redes neuronales, Desarrollo de redes neuronales para productos de inteligencia artificial.

2/72 - 7/79
Desarrollo de visión robótica; Desarrollo de sistemas basados en reglas para emular la vista.

9/71 - 1/72
Especialista en integración numérica, Ayudar a los sistemas bancarios a comunicarse entre sí.

Intereses

- Motocicletas de carreras
- Fabricar altavoces
- Ensamblar ordenadores personales
- Hacer bosquejos

A continuación se muestra el contenido del archivo db200150.htm.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Currículum:  Bruce Adamson
<!-- Begin Header Records  ========================================== -->
<!-- DB200150 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:39">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:39">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:39"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Currículum:  Bruce Adamson<HR>
<H2><A NAME="ToC">Tabla de contenidos<A NAME="ToC_1" HREF="#Header_1">
Currículum:  Bruce Adamson<BR>
<A NAME="ToC_2" HREF="#Header_2">Información personal<BR>
<A NAME="ToC_3" HREF="#Header_3">Información del departamento<BR>
<A NAME="ToC_4" HREF="#Header_4">Formación<BR>
<A NAME="ToC_5" HREF="#Header_5">Historial de trabajo<BR>
<A NAME="ToC_6" HREF="#Header_6">Intereses<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Currículum:  Bruce Adamson<P>
<H3><A NAME="Header_2">Información personal<DL COMPACT>
<DT>Dirección:
<DD>3600 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Teléfono:
<DD>(208) 725-4489
<DT>Fecha de nacimiento:
<DD>17 de Mayo de 1947
<DT>Sexo:
<DD>Varón
<DT>Estado civil:
<DD>Casada
<DT>Estatura:
<DD>1,83 cm
```

```
<DT>Peso:
<DD>83 kg.<P>
<H3><A NAME="Header_3">Información del departamento<DL COMPACT>
<DT>Número de empleada:
<DD>000150
<DT>Número del departamento:
<DD>D11
<DT>Directora:
<DD>Irving Stern
<DT>Empleo:
<DD>Diseñador
<DT>Teléfono:
<DD>(208) 385-4510
<DT>Fecha de contratación:
<DD>1972-02-12<P>
<H3><A NAME="Header_4">Formación<DL>
<P><DT>1971
<DD>Ingeniería Medioambiental, Master
<BR>
Johns Hopkins University
<P><DT>1968
<DD>Historia de América, Licenciatura
<BR>
Northwestern University<P>
<H3><A NAME="Header_5">Historial laboral<DL>
<P><DT>8/79 - presente
<DD>Diseño de redes neuronales
<BR>
Desarrollo de redes neuronales para productos de inteligencia artificial.
<P><DT>2/72 - 7/79
<DD>Desarrollo de visión robótica
<BR>
Desarrollo de sistemas basados en reglas para emular la vista.
<P><DT>9/71 - 1/72
<DD>Especialista en integración numérica
<BR>
Ayudar a los sistemas bancarios a comunicarse entre sí.<P>
<H3><A NAME="Header_6">Intereses<UL COMPACT>
<LI>Motocicletas de carreras
<LI>Fabricar altavoces
<LI>Ensamblar ordenadores personales
<LI>Hacer bosquejos<A NAME="Bot_Of_Page">
```

Foto de Walker

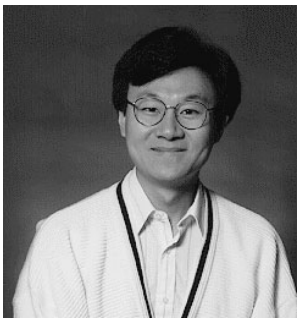


Figura 17. James H. Walker

Currículum de Walker

El texto siguiente se ha encontrado en el archivo db200190.asc.

Currículum: James H. Walker

Información personal

Dirección:
3500 Steeles Ave Mellonville, Idaho 83757

Teléfono:
(208) 555-7325

Fecha de nacimiento:
25 de Junio de 1952

Sexo: Varón

Estado civil:
Soltera

Estatura:
1,80 cm.

Peso: 75 kg.

Información del departamento

Número de empleada:
000190

Número del departamento:
D11

Directora:
Irving Stern

Empleo:
Diseñador

Teléfono:
(208) 555-2986

Fecha de contratación:
1974-07-26

Formación:

1974 Ingeniería Informática, Licenciatura University of Massachusetts

1972 Antropología lingüística, Licenciatura University of Toronto

Historial laboral

6/87 - presente
Diseño de microcódigos Optimización de algoritmos para funciones matemáticas.

4/77 - 5/87
Soporte técnico a impresoras, Instalación y soporte de impresoras láser.

9/74 - 3/77
Programación de mantenimiento, Parchear compilador de lenguaje assembly para sistemas principales.

Intereses

- Catar vinos
- Esquiar
- Nadar

- Bailar

A continuación se muestra el contenido del archivo db200190.htm.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Currículum:  James H. Walker
<!-- Begin Header Records  ========================================== -->
<!-- DB200190 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:20                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:59">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:59">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:59"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Currículum:  James H. Walker<HR>
<H2><A NAME="ToC">Tabla de contenidos<A NAME="ToC_1" HREF="#Header_1">
Currículum:  James H. Walker<BR>
<A NAME="ToC_2" HREF="#Header_2">Información personal<BR>
<A NAME="ToC_3" HREF="#Header_3">Información del departamento<BR>
<A NAME="ToC_4" HREF="#Header_4">Formación<BR>
<A NAME="ToC_5" HREF="#Header_5">Historial de trabajo<BR>
<A NAME="ToC_6" HREF="#Header_6">Intereses<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Currículum:  James H. Walker<P>
<H3><A NAME="Header_2">Información personal<DL COMPACT>
<DT>Dirección:
<DD>3500 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Teléfono:
<DD>(208) 725-7325
<DT>Fecha de nacimiento:
<DD>25 de Junio de 1952
<DT>Sexo:
<DD>Varón
<DT>Estado civil:
<DD>Soltera
<DT>Estatura:
<DD>1,80 cm
<DT>Peso:
<DD>75 kg.<P>
<H3><A NAME="Header_3">Información del departamento<DL COMPACT>
<DT>Número de empleada:
<DD>000190
<DT>Número del departamento:
<DD>D11
<DT>Directora:
<DD>Irving Stern
<DT>Empleo:
<DD>Diseñador
<DT>Teléfono:
<DD>(208) 385-2986
<DT>Fecha de contratación:
<DD>1974-07-26<P>
<H3><A NAME="Header_4">Formación<DL>
<P><DT>1974
<DD>Ingeniería Informática, Licenciatura
<BR>
University of Massachusetts
<P><DT>1972
<DD>Antropología lingüística, Licenciatura
<BR>
University of Toronto<P>
<H3><A NAME="Header_5">Historial laboral<DL>
<P><DT>6/87 - presente
<DD>Diseño de microcódigos
```

```
<BR>
Optimización de algoritmos para funciones matemáticas.
<P><DT>4/77 - 5/87
<DD>Soporte técnico a impresoras
<BR>
Instalación y soporte de impresoras láser.
<P><DT>9/74 - 3/77
<DD>Programación de mantenimiento
<BR>
Parchear compilador de lenguaje ensamblador para sistemas principales.<P>
<H3><A NAME="Header_6">Intereses<UL COMPACT>
<LI>Catar vinos
<LI>Esquiar
<LI>Nadar
<LI>Bailar<A NAME="Bot_Of_Page">
```

Apéndice G. Nombres de esquema reservados y palabras reservadas

Existen restricciones sobre el uso de ciertos nombres que el gestor de bases de datos necesita. En algunos casos, los nombres están reservados y los programas de aplicación no pueden utilizarlos. En otros casos, no es aconsejable la utilización de algunos nombres en programas de aplicación, aunque el gestor de bases de datos no impida su utilización.

Los nombres de esquema reservados son los siguientes:

- SYSCAT
- SYSFUN
- SYSIBM
- SYSIBMADM
- SYSPROC
- SYSPUBLIC
- SYSSTAT

Se recomienda encarecidamente que los nombres de esquema no empiecen nunca por el prefijo 'SYS', ya que, por convenio, 'SYS' se utiliza para indicar un área reservada por el sistema. Los alias, las variables globales, los activadores, las funciones definidas por el usuario o los tipos definidos por el usuario no se pueden colocar en un esquema cuyo nombre empiece por 'SYS' (SQLSTATE 42939).

El esquema DB2QP y el esquema SYSTOOLS se ponen a parte para que los utilicen las herramientas de DB2. No es aconsejable que los usuarios definan explícitamente objetos en estos esquemas, aunque el gestor de bases de datos no impida su utilización.

También se recomienda que no se utilice SESSION como nombre de esquema. Como las tablas temporales declaradas deben estar calificadas por SESSION, es posible que una aplicación declare una tabla temporal que tenga un nombre idéntico al de una tabla permanente, lo cual puede producir confusión en la lógica del programa. Para evitar esta situación, no utilice el esquema SESSION excepto cuando utilice tablas temporales declaradas.

En la versión 9 de DB2 no hay palabras específicamente reservadas. Las palabras clave se pueden utilizar como identificadores normales, excepto en un contexto en que también se interpretarían como palabras clave SQL. En dichos casos, la palabra debe especificarse como un identificador delimitado. Por ejemplo, COUNT no se puede utilizar como un nombre de columna en una sentencia SELECT, a menos que esté delimitado.

ISO/ANSI SQL2003 y otros productos de base de datos DB2 incluyen palabras reservadas que DB2 Database para Linux, UNIX y Windows no las impone; de todas formas, se recomienda que estas palabras no se utilicen como identificadores normales, porque reduce la portabilidad.

Por motivos de portabilidad entre productos de base de datos DB2, las siguientes palabras se deben considerar palabras reservadas:

Nombres de esquema reservados y palabras reservadas

```
ACTIVATE
ADD
AFTER
ALIAS
ALL
ALLOCATE
ALLOW
ALTER
AND
ANY
AS
ASENSITIVE
ASSOCIATE
ASUTIME
AT
ATTRIBUTES
AUDIT
AUTHORIZATION
AUX
AUXILIARY
BEFORE
BEGIN
BETWEEN
BINARY
BUFFERPOOL
BY
CACHE
CALL
CALLED
CAPTURE
CARDINALITY
CASCADED
CASE
CAST
CCSID
CHAR
CHARACTER
CHECK
CLONE
CLOSE
CLUSTER
COLLECTION
COLLID
COLUMN
COMMENT
COMMIT
CONCAT
CONDITION
CONNECT
CONNECTION
CONSTRAINT
CONTAINS
CONTINUE
COUNT
COUNT_BIG
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_LC_CTYPE
CURRENT_PATH
CURRENT_SCHEMA
CURRENT_SERVER
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_TIMEZONE
CURRENT_USER
DOCUMENT
DOUBLE
DROP
DSSIZE
DYNAMIC
EACH
EDITPROC
ELSE
ELSEIF
ENABLE
ENCODING
ENCRYPTION
END
END-EXEC
ENDING
ERASE
ESCAPE
EVERY
EXCEPT
EXCEPTION
EXCLUDING
EXCLUSIVE
EXECUTE
EXISTS
EXIT
EXPLAIN
EXTERNAL
EXTRACT
FENCED
FETCH
FIELDPROC
FILE
FINAL
FOR
FOREIGN
FREE
FROM
FULL
FUNCTION
GENERAL
GENERATED
GET
GLOBAL
GO
GOTO
GRANT
GRAPHIC
GROUP
HANDLER
HASH
HASHED_VALUE
HAVING
HINT
HOLD
HOUR
HOURS
IDENTITY
IF
IMMEDIATE
IN
INCLUDING
INCLUSIVE
INCREMENT
INDEX
INDICATOR
INF
INFINITY
LOCK
LOCKMAX
LOCKSIZE
LONG
LOOP
MAINTAINED
MATERIALIZED
MAXVALUE
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MINVALUE
MODE
MODIFIES
MONTH
MONTHS
NAN
NEW
NEW_TABLE
NEXTVAL
NO
NOCACHE
NOCYCLE
NODENAME
NODENUMBER
NOMAXVALUE
NOMINVALUE
NONE
NOORDER
NORMALIZED
NOT
NULL
NULLS
NUMPARTS
OBID
OF
OLD
OLD_TABLE
ON
OPEN
OPTIMIZATION
OPTIMIZE
OPTION
OR
ORDER
OUT
OUTER
OVER
OVERRIDING
PACKAGE
PADDED
PAGESIZE
PARAMETER
PART
PARTITION
PARTITIONED
PARTITIONING
PARTITIONS
PASSWORD
PATH
PIECESIZE
PLAN
POSITION
PRECISION
PREPARE
PREVVAL
ROUND_CEILING
ROUND_DOWN
ROUND_FLOOR
ROUND_HALF_DOWN
ROUND_HALF_EVEN
ROUND_HALF_UP
ROUND_UP
ROUTINE
ROW
ROW_NUMBER
ROWNUMBER
ROWS
ROWSET
RRN
RUN
SAVEPOINT
SCHEMA
SCRATCHPAD
SCROLL
SEARCH
SECOND
SECONDS
SECQTY
SECURITY
SELECT
SENSITIVE
SEQUENCE
SESSION
SESSION_USER
SET
SIGNAL
SIMPLE
SNAN
SOME
SOURCE
SPECIFIC
SQL
SQLID
STACKED
STANDARD
START
STARTING
STATEMENT
STATIC
STATMENT
STAY
STOGROUP
STORES
STYLE
SUBSTRING
SUMMARY
SYNONYM
SYSFUN
SYSIBM
SYSPROC
SYSTEM
SYSTEM_USER
TABLE
TABLESPACE
THEN
TIME
TIMESTAMP
TO
TRANSACTION
TRIGGER
TRIM
TRUNCATE
```

```
CURSOR              INHERIT          PRIMARY             TYPE
CYCLE               INNER            PRIQTY              UNDO
DATA                INOUT            PRIVILEGES          UNION
DATABASE            INSENSITIVE      PROCEDURE           UNIQUE
DATAPARTITIONNAME   INSERT           PROGRAM             UNTIL
DATAPARTITIONNUM    INTEGRITY        PSID                UPDATE
DATE                INTERSECT        PUBLIC              USAGE
DAY                 INTO             QUERY               USER
DAYS                IS               QUERYNO             USING
DB2GENERAL          ISOBID           RANGE               VALIDPROC
DB2GENRL            ISOLATION        RANK                VALUE
DB2SQL              ITERATE          READ                VALUES
DBINFO              JAR              READS               VARIABLE
DBPARTITIONNAME     JAVA             RECOVERY            VARIANT
DBPARTITIONNUM      JOIN             REFERENCES          VCAT
DEALLOCATE          KEEP             REFERENCING         VERSION
DECLARE             KEY              REFRESH             VIEW
DEFAULT             LABEL            RELEASE             VOLATILE
DEFAULTS            LANGUAGE         RENAME              VOLUMES
DEFINITION          LATERAL          REPEAT              WHEN
DELETE              LC_CTYPE         RESET               WHENEVER
DENSE_RANK          LEAVE            RESIGNAL            WHERE
DENSERANK           LEFT             RESTART             WHILE
DESCRIBE            LIKE             RESTRICT            WITH
DESCRIPTOR          LINKTYPE         RESULT              WITHOUT
DETERMINISTIC       LOCAL            RESULT_SET_LOCATOR  WLM
DIAGNOSTICS         LOCALDATE        RETURN              WRITE
DISABLE             LOCALE           RETURNS             XMLELEMENT
DISALLOW            LOCALTIME        REVOKE              XMLEXISTS
DISCONNECT          LOCALTIMESTAMP   RIGHT               XMLNAMESPACES
DISTINCT            LOCATOR          ROLE                YEAR
DO                  LOCATORS         ROLLBACK            YEARS
```

La lista siguiente contiene las palabras reservadas de ISO/ANSI SQL2003 que no se encuentran en la lista anterior:

```
ABS                                GROUPING         REGR_INTERCEPT
ARE                                INT              REGR_R2
ARRAY                              INTEGER          REGR_SLOPE
ASYMMETRIC                         INTERSECTION     REGR_SXX
ATOMIC                             INTERVAL         REGR_SXY
AVG                                LARGE            REGR_SYY
BIGINT                             LEADING          ROLLUP
BLOB                               LN               SCOPE
BOOLEAN                            LOWER            SIMILAR
BOTH                               MATCH            SMALLINT
CEIL                               MAX              SPECIFICTYPE
CEILING                            MEMBER           SQLEXCEPTION
CHAR_LENGTH                        MERGE            SQLSTATE
CHARACTER_LENGTH                   METHOD           SQLWARNING
CLOB                               MIN              SQRT
COALESCE                           MOD              STDDEV_POP
COLLATE                            MODULE           STDDEV_SAMP
COLLECT                            MULTISET         SUBMULTISET
CONVERT                            NATIONAL         SUM
CORR                               NATURAL          SYMMETRIC
CORRESPONDING                      NCHAR            TABLESAMPLE
COVAR_POP                          NCLOB            TIMEZONE_HOUR
COVAR_SAMP                         NORMALIZE        TIMEZONE_MINUTE
CUBE                               NULLIF           TRAILING
CUME_DIST                          NUMERIC          TRANSLATE
CURRENT_DEFAULT_TRANSFORM_GROUP    OCTET_LENGTH     TRANSLATION
CURRENT_ROLE                       ONLY             TREAT
CURRENT_TRANSFORM_GROUP_FOR_TYPE   OVERLAPS         TRUE
DEC                                OVERLAY          UESCAPE
DECIMAL                            PERCENT_RANK     UNKNOWN
DEREF                              PERCENTILE_CONT  UNNEST
```

```
ELEMENT
EXEC
EXP
FALSE
FILTER
FLOAT
FLOOR
FUSION
PERCENTILE_DISC
POWER
REAL
RECURSIVE
REF
REGR_AVGX
REGR_AVGY
REGR_COUNT
UPPER
VAR_POP
VAR_SAMP
VARCHAR
VARYING
WIDTH_BUCKET
WINDOW
WITHIN
```

Apéndice H. Ejemplos de interacción entre activados y restricciones de referencia

Las operaciones de actualización pueden causar la interacción de activadores con restricciones de referencia y restricciones de comprobación.

La Figura 18 y la descripción asociada son representativas del proceso que se lleva a cabo para una sentencia que actualiza los datos de la base de datos.

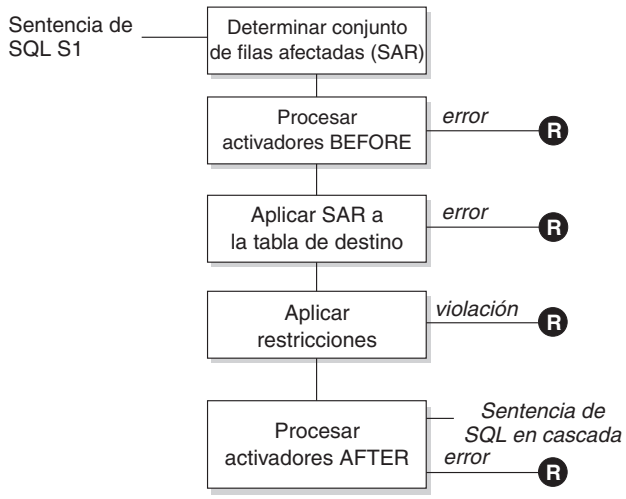


Figura 18. Proceso de una sentencia con activadores y restricciones asociados

La Figura 18 muestra el orden general de proceso para una sentencia que actualiza una tabla. Supone una situación donde la tabla incluye activadores BEFORE, restricciones de referencia, restricciones de comprobación y activadores AFTER en cascada. A continuación encontrará una descripción de los recuadros y de los demás elementos que se encuentran en la Figura 18.

- sentencia S_1

  Es la sentencia DELETE, INSERT o UPDATE que empieza el proceso. La sentencia SQL S_1 identifica una tabla (o una vista actualizable de alguna tabla) a la que se hace referencia como la *tabla objetivo* en toda la descripción.

- Determinación del conjunto de filas afectadas

  Este paso es el punto de arranque de un proceso que se repite para las reglas de supresión de restricción de referencia de CASCADE y SET NULL y para las sentencias SQL en cascada de los activadores AFTER.

  La finalidad de este paso es determinar el *conjunto de filas afectadas* para la sentencia. El conjunto de filas incluidas se basa en la sentencia:

  – para DELETE, todas las filas que satisfagan la condición de búsqueda de la sentencia (o la fila actual para una DELETE con posición)

  – para INSERT, las filas identificadas por la cláusula VALUES o la selección completa

– para UPDATE, todas las filas que satisfagan la condición de búsqueda (o la fila actual para una UPDATE con posición).

Si el conjunto de filas afectadas está vacío, no habrá activadores BEFORE, cambios para aplicar a la tabla objetivo ni restricciones para procesar la sentencia.

- Proceso de activadores BEFORE

  Todos los activadores BEFORE se procesan por orden ascendente de creación. Cada activador BEFORE procesará la acción activada una vez para cada fila del conjunto de filas afectadas.

  Puede producirse un error durante el proceso de una acción activada en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia original S_1 (hasta entonces).

  Si no hay ningún activador BEFORE o el conjunto de filas afectadas está vacío, este paso se salta.

- Aplicación del conjunto de filas afectadas a la tabla objetivo

  La supresión, inserción o actualización real se aplica utilizando el conjunto de filas afectadas en la tabla objetivo de la base de datos.

  Puede producirse un error al aplicar el conjunto de filas afectadas (como el intento de insertar una fila con una clave duplicada donde existe un índice de unicidad) en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia original S_1 (hasta entonces).

- Aplicación de restricciones

  Las restricciones asociadas con la tabla objetivo se aplican si el conjunto de filas afectadas no está vacío. Esto incluye restricciones de unicidad, índices de unicidad, restricciones de referencia, restricciones de comprobación y comprobaciones relacionadas con WITH CHECK OPTION en vistas. Las restricciones de referencia con reglas de supresión en cascada o de establecer nulo pueden provocar que se activen activadores adicionales.

  Una violación de cualquier restricción o WITH CHECK OPTION da como resultado un error y se retrotraen todos los cambios realizados como resultado de S_1 (hasta entonces).

- Proceso de activadores AFTER

  Todos los activadores AFTER activados por S_1 se procesan por orden ascendente de creación.

  Los activadores FOR EACH STATEMENT procesarán la acción activada exactamente una vez, incluso si el conjunto de filas afectadas está vacío. Los activadores FOR EACH ROW procesarán la acción activada una vez para cada fila del conjunto de filas afectadas.

  Puede producirse un error durante el proceso de una acción activada en cuyo caso se retrotraen todos los cambios realizados como resultado de la sentencia original S_1 (hasta entonces).

  La acción activada de un activador puede incluir sentencias activadas que sean sentencias DELETE, INSERT o UPDATE. Para esta descripción, cada una de dichas sentencias se considera una *sentencia en cascada*.

  Una sentencia en cascada es una sentencia DELETE, INSERT o UPDATE que se procesa como parte de la acción activada de un activador AFTER. Esta sentencia empieza un nivel en cascada del proceso de activador. Puede considerarse como la asignación de la sentencia activada como una S_1 nueva y la realización repetida de todos los pasos descritos aquí.

Una vez que todas las sentencias activadas de todos los activadores AFTER activados por cada S_1 hayan terminado su proceso, el proceso de la S_1 original se ha completado.

- R = retrotraer cambios antes de S_1

  Cualquier error que se produzca (incluyendo las violaciones de restricciones) durante el proceso da como resultado una retrotracción de todos los cambios realizados directa o indirectamente como resultado de la sentencia original S_1. Por lo tanto, la base de datos vuelve a estar en el mismo estado que estaba inmediatamente antes de la ejecución de la sentencia original S_1

Apéndice I. Tablas de Explain

Las tablas de Explain capturan planes de acceso cuando se activa el recurso Explain. Las tablas de Explain se deben crear antes de invocar Explain. Puede crearlas mediante uno de los métodos siguientes:

- Llame al procedimiento SYSPROC.SYSINSTALLOBJECTS:

  ```
  db2 CONNECT TO nombre-basedatos
  db2 CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C',
          CAST (NULL AS VARCHAR(128)), CAST (NULL AS VARCHAR(128))
  ```

  Esta llamada crea las tablas de Explain en el esquema SYSTOOLS. Para crearlas en un esquema distinto, especifique un nombre de esquema como el últimoparámetro de la llamada.

- Ejecute el archivo de mandatos de DB2 EXPLAIN.DDL:

  ```
  db2 CONNECT TO nombre-basedatos
  db2 -tf EXPLAIN.DDL
  ```

  Este archivo de mandatos crea tablas de Explain en el esquema actual. Está ubicado en el directorio DB2PATH\misc en los sistemas operativos Windows y en el directorio INSTHOME/sqllib/misc en los sistemas operativosLinux y UNIX. DB2PATH es la ubicación donde está instaladala copia de DB2 e INSTHOME es el directorio inicial de la instancia.

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.

Cuando el recurso Explain llene las tablas Explain, no activarán activadores ni restricciones de comprobación. Por ejemplo, si se ha definido un activador de inserción en la tabla EXPLAIN_INSTANCE y se ha explicado una sentencia elegible, el activador no se activará.

Para mejorar el rendimiento del recurso Explain en un sistema de bases de datos particionadas, se recomienda crear las tablas de Explain en un solo grupo de particiones de la base de datos particionada, preferiblemente en la misma partición de base de datos a la que se conectará al compilar la consulta.

Tabla ADVISE_INDEX

La tabla ADVISE_INDEX representa los índices recomendados.

Tabla 238. Tabla ADVISE_INDEX. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sección en el paquete con el que está relacionada esta información de Explain.
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.
NAME	VARCHAR(128)	No	No	Nombre del índice.
CREATOR	VARCHAR(128)	No	No	Calificador del nombre de índice.
TBNAME	VARCHAR(128)	No	No	Nombre de la tabla o apodo en el que se define el índice.
TBCREATOR	VARCHAR(128)	No	No	Calificador del nombre de tabla.
COLNAMES	CLOB(2M)	No	No	Lista de nombres de columnas.
UNIQUERULE	CHAR(1)	No	No	Norma de unicidad: • D = Los duplicados están permitidos • P = Índice primario • U = Sólo se permiten entradas exclusivas

Tabla 238. Tabla ADVISE_INDEX (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
COLCOUNT	SMALLINT	No	No	El número de columnas de la clave más el número de columnas INCLUDE si hay alguna.
IID	SMALLINT	No	No	ID interno de índice.
NLEAF	BIGINT	No	No	El número de páginas; -1 si no se reúnen estadísticas.
NLEVELS	SMALLINT	No	No	Número de niveles de índices; -1 si no se reúnen estadísticas.
FIRSTKEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada; -1 si no se reúnen estadísticas.
FULLKEYCARD	BIGINT	No	No	El número de valores de clave completa diferenciada; -1 si no se reúnen estadísticas.
CLUSTERRATIO	SMALLINT	No	No	Nivel de clúster de los datos con el índice; -1 si no se reúnen estadísticas o si se reúnen estadísticas de índice detalladas (en este caso, se utilizará CLUSTERFACTOR en su lugar).
AVGPARTITION_ CLUSTERRATIO	SMALLINT	No	No	Nivel de clúster de los datos dentro de una sola partición de datos. -1 si la tabla no está particionada en tablas, si no se recopilan estadísticas o si se recopilan estadísticas detalladas (en cuyo caso se utiliza AVGPARTITION_ CLUSTERFACTOR en su lugar).
AVGPARTITION_ CLUSTERFACTOR	DOUBLE	No	No	Mejor medición del nivel de clúster dentro de una sola partición de datos. -1 si la tabla no está particionada en tablas, si no se recopilan estadísticas o si el índice está definido sobre un apodo.
AVGPARTITION_PAGE_ FETCH_PAIRS	VARCHAR(520)	No	No	Una lista de pares de enteros en formato de caracteres. Cada par representa un tamaño potencial de la agrupación de almacenamientos intermedios en las captaciones de páginas correspondientes necesarias para acceder a una sola partición de datos desde la tabla. Serie de longitud cero si no hay datos disponibles o si la tabla no está particionada en tablas.
DATAPARTITION_ CLUSTERFACTOR	DOUBLE	No	No	Una medición estadística del "clúster" de claves de índices con respecto a particiones de datos. Este campo tiene un número entre cero y uno, done uno representa la agrupación en clúster perfecta y cero representa que no existe.
CLUSTERFACTOR	DOUBLE	No	No	Mejor medición del nivel de clúster o -1 si no se han reunido estadísticas de índice detalladas o si el índice está definido en un apodo.
USERDEFINED	SMALLINT	No	No	Definido por el usuario.

Tabla 238. Tabla ADVISE_INDEX (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
SYSTEM_REQUIRED	SMALLINT	No	No	• 1 si se cumple una de las condiciones siguientes: – Este índice es necesario para una restricción de clave primaria o exclusiva o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla con clústeres de varias dimensiones (MDC). – Se trata de un índice en la columna OID de una tabla con tipo. • 2 si se cumplen estas dos condiciones: – Este índice es necesario para una restricción de clave primaria o exclusiva o bien este índice es un índice de bloques de dimensión o un índice de bloques compuestos para una tabla MDC. – Se trata de un índice en la columna OID de una tabla con tipo. • De lo contrario, 0.
CREATE_TIME	TIMESTAMP	No	No	Hora en que se ha creado el índice.
STATS_TIME	TIMESTAMP	Sí	No	Última vez que se ha realizado un cambio en las estadísticas registradas para este índice. Nulo si no hay estadísticas disponibles.
PAGE_FETCH_PAIRS	VARCHAR(520)	No	No	Una lista de pares de enteros, representada en la forma de caracteres. Cada par representa el número de páginas de un almacenamiento intermedio hipotético y el número de lecturas de páginas necesario para explorar la tabla con este índice utilizando dicho almacenamiento intermedio hipotético. (Serie de longitud cero si no hay datos disponibles.)
REMARKS	VARCHAR(254)	Sí	No	Comentario suministrado por el usuario o nulo.
DEFINER	VARCHAR(128)	No	No	Usuario que ha creado el índice.
CONVERTED	CHAR(1)	No	No	Reservado para una utilización futura.
SEQUENTIAL_PAGES	BIGINT	No	No	El número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. (-1 si no hay estadísticas disponibles.)
DENSITY	INTEGER	No	No	Proporción de SEQUENTIAL_PAGES para numerar las páginas del rango de páginas ocupadas por el índice, expresada como un porcentaje (entero entre 0 y 100, -1 si no hay estadísticas disponibles.)
FIRST2KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las dos primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)
FIRST3KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las tres primeras columnas del índice (-1 si no hay estadísticas o si no son aplicables)

Tabla 238. Tabla ADVISE_INDEX (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
FIRST4KEYCARD	BIGINT	No	No	El número de claves diferenciadas que utilizan las cuatro primeras columnas del índice (-1 si no hay estadísticas o no son aplicables)
PCTFREE	SMALLINT	No	No	Porcentaje de cada página de hoja de índice que se va a reservar durante la creación inicial del índice. Este espacio está disponible para inserciones futuras después de la creación del índice.
UNIQUE_COLCOUNT	SMALLINT	No	No	El número de columnas necesarias para una clave de unicidad. Siempre <=COLCOUNT. < COLCOUNT solamente si hay columnas INCLUDE. -1 si el índice no tiene clave de unicidad (permite duplicados)
MINPCTUSED	SMALLINT	No	No	Si no es cero, se habilita la desfragmentación del índice en línea y el valor es el umbral del espacio mínimo utilizado antes de fusionar las páginas.
REVERSE_SCANS	CHAR(1)	No	No	• Y = El índice da soporte a las exploraciones invertidas • N = El índice no da soporte a las exploraciones invertidas
USE_INDEX	CHAR(1)	Sí	No	• Y = índice recomendado o evaluado • N = índice no recomendado • R = se ha recomendado la separación del clúster de un índice de clúster RID (por parte del Asesor de diseño); esto ocurre cuando se recomienda un índice de clúster RID nuevo para la tabla.
CREATION_TEXT	CLOB(2M)	No	No	La sentencia de SQL se utiliza para crear el índice.
PACKED_DESC	BLOB(1M)	Sí	No	Descripción interna de la tabla.
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
INDEXTYPE	VARCHAR(4)	No	No	Tipo de índice. • CLUS = Agrupación en clúster • REG = Regular • DIM = Índice de bloques de dimensión • BLOK = Índice de bloques
EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si el índice ya existe en el catálogo de la base de datos.
RIDTOBLOCK	CHAR(1)	No	No	Debe definirse en 'Y' si se ha utilizado el índice RID para crear un índice de bloques en el Asesor de diseño.

Tabla ADVISE_INSTANCE

La tabla ADVISE_INSTANCE contiene información acerca de la ejecución de db2advis, incluida la hora de inicio. Contiene una fila para cada ejecución de db2advis. Otras tablas ADVISE presentan una clave foránea (RUN_ID) que enlaza las filas creadas durante la misma ejecución del Asesor de diseño con la columna START_TIME de la tabla ADVISE_INSTANCE.

Tabla 239. Tabla ADVISE_INSTANCE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
START_TIME	TIMESTAMP	No	PK	Hora en la que empieza la ejecución de db2advis.
END_TIME	TIMESTAMP	No	No	Hora en la que finaliza la ejecución de db2advis.
MODE	VARCHAR(4)	No	No	El valor que se ha especificado con la opción -m en el Asesor de diseño; por ejemplo, 'MC' para especificar MQT y MDC.
WKLD_COMPRESSION	CHAR(4)	No	No	La compresión de carga de trabajo bajo la que se ha ejecutado el Asesor de diseño.
STATUS	CHAR(9)	No	No	Estado de una ejecución del Asesor de diseño. El estado puede ser 'STARTED', 'COMPLETED' (si es satisfactorio) o un número de error que lleve el prefijo 'EI' para los errores internos o 'EX' para los errores externos, en cuyo caso el número de error representa el SQLCODE.

Tabla ADVISE_MQT

La tabla ADVISE_MQT contiene información acerca de las tablas de consultas materializadas (MQT) recomendadas por el Asesor de diseño.

Tabla 240. Tabla ADVISE_MQT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
NAME	VARCHAR(128)	No	No	Nombre de la MQT.
CREATOR	VARCHAR(128)	No	No	Nombre del creador de la MQT.
IID	SMALLINT	No	No	Identificador interno.
CREATE_TIME	TIMESTAMP	No	No	Hora en que se ha creado la MQT.
STATS_TIME	TIMESTAMP	Sí	No	Hora en que se han recogido las estadísticas.
NUMROWS	DOUBLE	No	No	Número estimado de filas de la MQT.
NUMCOLS	SMALLINT	No	No	Número de columnas definido en la MQT.
ROWSIZE	DOUBLE	No	No	Longitud media (en bytes) de una fila de la MQT.
BENEFIT	FLOAT	No	No	Reservado para una utilización futura.
USE_MQT	CHAR(1)	Sí	No	Debe definirse en 'Y' si se recomienda la MQT.
MQT_SOURCE	CHAR(1)	Sí	No	Indica si se ha generado el candidato de la MQT. Debe definirse en 'I' si el candidato de la MQT es una MQT de renovación inmediata o en 'D' si sólo puede crearse como una MQT de renovación diferida completa.
QUERY_TEXT	CLOB(2M)	No	No	Contiene la consulta que define la MQT.
CREATION_TEXT	CLOB(2M)	No	No	Contiene el DDL de CREATE TABLE para la MQT.

Tabla ADVISE_MQT

Tabla 240. Tabla ADVISE_MQT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
SAMPLE_TEXT	CLOB(2M)	No	No	Contiene la consulta de muestreo que se utiliza para obtener estadísticas detalladas para la MQT. Sólo se utiliza cuando se necesitan estadísticas detalladas para el Asesor de diseño. Las estadísticas de muestreo resultantes se mostrarán en esta tabla. Si es nulo, no se había creado ninguna consulta de muestreo para esta MQT.
COLSTATS	CLOB(2M)	No	No	Contiene las estadísticas de columna para la MQT (si no es nulo). Estas estadísticas están en formato XML e incluyen el nombre de la columna, la cardinalidad de la columna y, opcionalmente, los valores HIGH2KEY y LOW2KEY.
EXTRA_INFO	BLOB(2M)	No	No	Reservado para salidas varias.
TBSPACE	VARCHAR(128)	No	No	El espacio de tablas que se recomienda para la MQT.
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
REFRESH_TYPE	CHAR(1)	No	No	Debe definirse en 'I' para inmediato o en 'D' para diferido.
EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si la MQT ya existe en el catálogo de la base de datos.

Tabla ADVISE_PARTITION

La tabla ADVISE_PARTITION contiene información acerca de las particiones de base de datos recomendadas por el Asesor de diseño y sólo puede rellenarse en un entorno de bases de datos particionadas.

Tabla 241. Tabla ADVISE_PARTITION. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	No	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	No	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	No	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	No	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	No	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	No	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.
TBNAME	VARCHAR(128)	Sí	No	Especifica el nombre de la tabla.
TBCREATOR	VARCHAR(128)	Sí	No	Especifica el nombre del creador de la tabla.
PMID	SMALLINT	Sí	No	Especifica el ID de correlación de la distribución.
TBSPACE	VARCHAR(128)	Sí	No	Especifica el espacio de tablas en el que reside la tabla.
COLNAMES	CLOB(2M)	Sí	No	Especifica los nombres de las columnas de las particiones de base de datos, separados por comas.

Tabla ADVISE_PARTITION

Tabla 241. Tabla ADVISE_PARTITION (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
COLCOUNT	SMALLINT	Sí	No	Especifica el número de columnas de particionamiento de base de datos.
REPLICATE	CHAR(1)	Sí	No	Especifica si la partición de base de datos se duplica o no se duplica.
COST	DOUBLE	Sí	No	Especifica el coste del uso de la partición de base de datos.
USEIT	CHAR(1)	Sí	No	Especifica si la partición de base de datos se utiliza o no en modalidad EVALUATE PARTITION. Se utiliza una partición de base de datos si USEIT está establecido en 'Y' o 'y'.
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.

Tabla ADVISE_TABLE

La tabla ADVISE_TABLE almacena el lenguaje de definición de datos (DDL) para la creación de tablas utilizando las recomendaciones finales del Asesor de diseño para las tablas de consultas materializadas (MQT), las tablas de clústeres de varias dimensiones (MDC) y la partición de bases de datos.

Tabla 242. Tabla ADVISE_TABLE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
RUN_ID	TIMESTAMP	Sí	FK	Valor correspondiente a START_TIME de una fila de la tabla ADVISE_INSTANCE que la enlaza con la misma ejecución del Asesor de diseño.
TABLE_NAME	VARCHAR(128)	No	No	Nombre de la tabla.
TABLE_SCHEMA	VARCHAR(128)	No	No	Nombre del creador de la tabla.
TABLESPACE	VARCHAR(128)	No	No	El espacio de tablas en el que va a crearse la tabla.
SELECTION_FLAG	VARCHAR(4)	No	No	Indica el tipo de recomendación. Los valores válidos son 'M' para MQT, 'P' para partición de bases de datos y 'C' para MDC. Este campo puede incluir cualquier subconjunto de estos valores. Por ejemplo, 'MC' indica que la tabla se recomienda como una tabla MQT y MDC.
TABLE_EXISTS	CHAR(1)	No	No	Debe definirse en 'Y' si la tabla ya existe en el catálogo de la base de datos.
USE_TABLE	CHAR(1)	No	No	Debe definirse en 'Y' si la tabla tiene recomendaciones del Asesor de diseño.
GEN_COLUMNS	CLOB(2M)	No	No	Contiene una serie de columnas generadas si esta fila incluye una recomendación de MDC que necesita columnas generadas en el DDL de creación de tablas.
ORGANIZE_BY	CLOB(2M)	No	No	Para las recomendaciones de MDC, contiene la cláusula ORGANIZE BY del DDL de creación de tablas.
CREATION_TEXT	CLOB(2M)	No	No	Contiene el DDL de creación de tablas.
ALTER_COMMAND	CLOB(2M)	No	No	Contiene una sentencia ALTER TABLE para la tabla.

Tabla ADVISE_WORKLOAD

La tabla ADVISE_WORKLOAD representa la sentencia que forma la carga de trabajo.

Tabla 243. Tabla ADVISE_WORKLOAD. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
WORKLOAD_NAME	CHAR(128)	No	No	Nombre del conjunto de sentencias de SQL (carga de trabajo) a la que esta sentencia pertenece.
STATEMENT_NO	INTEGER	No	No	El número de sentencias de la carga de trabajo con el que está relacionada esta información de explicación.
STATEMENT_TEXT	CLOB(1M)	No	No	Contenido de la sentencia de SQL.
STATEMENT_TAG	VARCHAR(256)	No	No	Distintivo identificador para cada sentencia de SQL explicada.
FREQUENCY	INTEGER	No	No	Las veces que esta sentencia aparece en la carga de trabajo.
IMPORTANCE	DOUBLE	No	No	Importancia de la sentencia.
WEIGHT	DOUBLE	No	No	Prioridad de la sentencia.
COST_BEFORE	DOUBLE	Sí	No	Coste de la consulta (in timerons) si no se han creado las recomendaciones.
COST_AFTER	DOUBLE	Sí	No	Coste de la consulta (in timerons) si se han creado las recomendaciones. COST_AFTER refleja todas las recomendaciones excepto las que pertenezcan a índices agrupados y al clustering de varias dimensiones (MDC).
COMPILABLE	CHAR(17)	Sí	No	Indica los errores de compilación de la consulta que se han producido al intentar preparar la sentencia. Si esta columna es NULL o no empieza por SQLCA, la consulta de SQL podría compilarla db2advis. Si db2advis o el Asesor de diseño detectan un error de compilación, el valor de columna COMPILABLE constará de un campo SQLCA.sqlcaid de 8 bytes de longitud seguido de dos puntos (:) y un campo SQLCA.sqlstate de 8 bytes de longitud, que es el código de retorno de la sentencia SQL.

Tabla EXPLAIN_ACTUALS

La tabla EXPLAIN_ACTUALS contiene información relacionada con los datos reales de la sección de Explain.

Tabla 244. Tabla EXPLAIN_ACTUALS. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	FK	ID exclusivo para este operador en esta consulta.
DBPARTITIONNUM	INTEGER	No	No	El número de partición de la partición de base de datos en la que se ha ejecutado el operador.
PREDICATE_ID	INTEGER	Sí	No	ID del predicado aplicado a este operador. NULL si los datos reales son datos reales de operador.
HOW_APPLIED	CHAR(10)	Sí	No	La forma en que este operador utiliza el predicado. NULL si PREDICATE_ID es NULL.
ACTUAL_TYPE	VARCHAR(12)	No	No	El tipo de los datos reales.
ACTUAL_VALUE	DOUBLE	Sí	No	El valor de los datos reales. NULL si los datos reales no están disponibles para este operador.

Tabla EXPLAIN_ARGUMENT

La tabla EXPLAIN_ARGUMENT representa las características exclusivas para cada operador individual, si hay alguno.

Tabla 245. Tabla EXPLAIN_ARGUMENT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	No	ID exclusivo para este operador en esta consulta.
ARGUMENT_TYPE	CHAR(8)	No	No	El tipo de argumento para este operador.
ARGUMENT_VALUE	VARCHAR(1024)	Sí	No	El valor del argumento para este operador. NULL si el valor está en LONG_ARGUMENT_VALUE.
LONG_ARGUMENT_ VALUE	CLOB(2M)	Sí	No	El valor del argumento para este operador, cuando el texto no encaja ARGUMENT_VALUE. NULL si el valor está en ARGUMENT_VALUE.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	Indicadores de agregación parcial.
BITFLTR	INTEGER FALSE	El tamaño del filtro de bits utilizado por la unión de generación aleatoria.
BLD_LEVEL	Identificador de nivel de creación de DB2	Serie de identificación interna de la versión del código fuente.
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	Intento de bloqueo del nivel de bloqueo.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción

CONCACCR	Cada fila de este tipo contendrá:
	• Nivel del valor de esta sentencia:
	BIND
	`Application BIND with CONCURRENT ACCESS RESOLUTION opción`
	PREP
	`Statement prepared with CONCURRENT ACCESS RESOLUTION atributos`
	• La resolución de acceso simultáneo en vigor:
	USE CURRENTLY COMMITTED
	`Concurrent access resolution of application bind or statement prepare is USE CURRENTLY COMMITTED`
	WAIT FOR OUTCOME
	`Concurrent access resolution of application bind or statement prepare is WAIT FOR OUTCOME`
		Indica la resolución de acceso simultáneo que se utiliza para generar el plan de acceso para esta sentencia.

CSERQY	TRUE FALSE	La consulta remota es una subexpresión común.
CSETEMP	TRUE FALSE	Tabla temporal sobre el Distintivo de subexpresión común.
CUR_COMM	TRUE	Accede a las filas confirmadas actualmente cuando el valor del parámetro de configuración de la base de datos **cur_commit** no es DISABLE. Este plan de acceso se habilita para las sentencias aplicables utilizando: • CONCURRENT ACCESS RESOLUTION con la opción USE CURRENTLY COMMITTED en la vinculación o preparación • El parámetro de configuración de la base de datos **cur_commit** con el valor ON
DIRECT	TRUE	Indicador de lectura directa.
DPESTFLG	TRUE FALSE	Indica si el valor DPNUMPRT se utiliza en una estimación o no. Los valores posibles son 'TRUE' (DPNUMPRT representa el número estimado de particiones de datos a las que se ha accedido) o 'FALSE' (DPNUMPRT representa el número real de particiones de datos a las que se ha accedido).

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
DPLSTPRT	NONE CHARACTER	Representa las particiones de datos a las que se ha accedido. Es una lista delimitada por comas (por ejemplo: 1,3,5) o por guiones (por ejemplo: 1-5) de las particiones de datos a las que se ha accedido. Un valor 'NONE' significa que no quedan particiones de datos una vez se han aplicado los predicados especificados.
DPNUMPRT	INTEGER	Representa el número real o estimado de particiones de datos a las que se ha accedido.
DSTSEVER	Nombre del servidor	Servidor de destino (enviar desde).
DUPLWARN	TRUE FALSE	Duplica el distintivo de Aviso.
EARLYOUT	LEFT RIGHT GROUPBY NONE	Indicador de pronto fuera. LEFT indica que cada fila de la tabla externa únicamente puede unirse a una fila de la tabla interna como máximo. RIGHT indica que cada fila de la tabla interna únicamente puede unirse a una fila de la tabla externa como máximo. NONE indica que no hay proceso externo anticipado. GROUPBY indica que se permite el proceso externo anticipado debido a una operación group by.
ENVVAR	Cada fila de este tipo contendrá: • Nombre de la variable de entorno • Valor de la variable de entorno	Variable de entorno que afecta al optimizador
ERRTOL	Cada fila de este tipo contendrá un par SQLSTATE y SQLCODE.	Una lista de errores que debe tolerarse.
EVALUNCO	TRUE	Evalúa los datos no confirmados utilizando un aplazamiento de bloqueo. Esta operación se habilita con la variable de registro **DB2_EVALUNCOMMITTED**.
FETCHMAX	IGNORE INTEGER	Altera temporalmente el valor del argumento MAXPAGES en el operador FETCH.
GREEDY	TRUE	Indica que el optimizador ha utilizado el algoritmo ávido para planificar el acceso.
GLOBLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE NO LOCK OBTAINED SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Representa información de intento de bloqueo global para un objeto de tabla particionada.
GROUPBYC	TRUE FALSE	Indica si se proporcionan columnas Agrupar por.
GROUPBYN	Integer	El número de columnas de comparación.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
GROUPBYR	Cada fila de este tipo contendrá: • El valor ordinal de la columna en grupo en clave (seguido por dos puntos y un espacio) • Nombre de columna	Requisito de Agrupar por.
HASHCODE	24 32	Tamaño (en bits) del código de generación aleatoria utilizado para la unión de generación aleatoria.
INNERCOL	Cada fila de este tipo contendrá: • El valor ordinal de la columna en orden (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de orden interno.
INPUTXID	Un identificador de nodos de contexto	INPUTXID identifica el nodo de contexto de entrada que utiliza el operador XSCAN.
ISCANMAX	IGNORE INTEGER	Altera temporalmente el valor del argumento MAXPAGES en el operador ISCAN.
JN INPUT	INNER OUTER	Indica si el operador es el operador que alimenta la parte interna o externa de una unión.
LCKAVOID	TRUE	Prevención de bloqueo: el acceso a filas evitará el bloqueo de los datos confirmados.
LISTENER	TRUE FALSE	Indicador de Cola de tabla receptora.
MAXPAGES	ALL NONE INTEGER	El número máximo de páginas esperadas para la lectura anticipada.
MAXRIDS	NONE INTEGER	El número máximo de Identificadores de fila que se incluirán en cada petición de lectura anticipada por lista.
NUMROWS	INTEGER	El número de filas que se espera clasificar.
ONEFETCH	TRUE FALSE	Un indicador de lectura.
OUTERCOL	Cada fila de este tipo contendrá: • El valor ordinal de la columna en orden (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de orden externo.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	Indicador de unión externa.
PARTCOLS	Nombre de columna	Columnas de partición para el operador.
PREFETCH	LIST NONE SEQUENTIAL	Tipo de lectura anticipada admisible.
REOPT	ALWAYS ONCE	La sentencia se ha optimizado utilizando valores enlazados para los marcadores de parámetro, las variables del lenguaje principal y los registros especiales.
RMTQTEXT	Texto de consulta	Texto de consulta remoto
RNG_PROD	Nombre de función	Función que produce rangos para el acceso ampliado al índice.
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	Intento de bloqueo de fila.
ROWWIDTH	INTEGER	Anchura de fila que se ha de clasificar.
RSUFFIX	Texto de consulta	Sufijo de SQL remoto.
SCANDIR	FORWARD REVERSE	Dirección de exploración.
SCANGRAN	INTEGER	Paralelismo intrapartición, granularidad de la exploración paralela de intrapartición, expresada en SCANUNIT.
SCANSPEED	SLOW FAST	'SLOW' indica que se espera que la exploración progrese lentamente sobre la tabla. Por ejemplo, si la exploración es la parte exterior de una unión de bucle anidado. 'FAST' indica que se espera que la exploración progrese a mayor velocidad. Esta información se utiliza para agrupar las exploraciones y así compartir de forma eficaz los registros de las agrupaciones de almacenamientos intermedios.
SCANTYPE	LOCAL PARALLEL	Paralelismo intrapartición, exploración de Índice o Tabla.
SCANUNIT	ROW PAGE	Paralelismo intrapartición, unidad de granularidad de exploración.
SHARED	TRUE	Paralelismo intrapartición, indicador TEMP compartido.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
SKIP_INS	TRUE	Omisión insertada. El acceso a filas omitirá las filas insertadas que no estén confirmadas. Este comportamiento se habilita con la variable de registro **DB2_SKIPINSERTED** o cuando la semántica confirmada actualmente está en vigor.
SKIPDKEY	TRUE	Omisión de las claves suprimidas. El acceso a filas omitirá las claves suprimidas no confirmadas. Este comportamiento se habilita con la variable de registro **DB2_SKIPDELETED**.
SKIPDROW	TRUE	Omisión de las filas suprimidas. El acceso a filas omitirá las filas suprimidas no confirmadas. Este comportamiento se habilita con la variable de registro **DB2_SKIPDELETED**.
SLOWMAT	TRUE FALSE	Distintivo de materialización lenta.
SNGLPROD	TRUE FALSE	Indicador SORT o TEMP de paralelismo intrapartición generado por un solo agente.
SORTKEY	Cada fila de este tipo contendrá: • El valor ordinal de la columna en clave (seguido por dos puntos y un espacio) • Nombre de columna • Valor de orden **(A)** Ascendente **(D)** Descendente	Columnas de clave de clasificación.
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	Paralelismo intrapartición, tipo SORT.
SRCSEVER	Nombre del servidor	Servidor fuente (enviar a).
SPILLED	INTEGER	El número estimado de páginas del vertido SORT
SQLCA	Información de advertencia	Códigos de advertencia y de razón emitidos durante la operación Explain.
STARJOIN	YES	El operador `IXAND` forma parte de una unión en estrella.
STMTHEAP	INTEGER	Tamaño de la pila de sentencia al iniciar la compilación de una sentencia.
STREAM	TRUE FALSE	La fuente remota es de modalidad continua.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Intento de bloqueo de tabla.
TEMPSIZE	INTEGER	Tamaño de página de la tabla temporal.
THROTTLE	TRUE FALSE	La disminución mejora el rendimiento de otras exploraciones que, de lo contrario, se retrasarían y tendrían que volver a leer las mismas páginas. 'TRUE' si se puede disminuir la exploración. 'FALSE' si no se puede disminuir la exploración.
TMPCMPRS	YESELIGIBLE	El valor YES indica que se aplica la compresión. El valor ELIGIBLE indica que puede aplicarse la compresión si la tabla alcanza un tamaño lo suficientemente grande. La ausencia de TMPCMPRS indica que la tabla temporal no se comprime.
TQDEGREE	INTEGER	Paralelismo intrapartición, número de subagentes que acceden a la Cola de tabla.
TQMERGE	TRUE FALSE	Indicador de Fusión de cola de tabla (clasificada).
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	Propiedad de lectura de Cola de tabla.
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	Propiedad de envío de Cola de tabla.
TQ TYPE	LOCAL	Paralelismo intrapartición, Cola de tabla.
TQ_ORIGIN	ASYNCHRONYXTQ	La razón por la que la Cola de tabla se introdujo en el plan de acceso.
TRUNCTQ	INPUT OUTPUT INPUT AND OUTPUT	Indicador de Cola de tabla truncada. INPUT indica que el truncado se produce en la entrada de la Cola de tabla. OUPUT indica que el truncado se produce en la salida de la Cola de tabla. INPUT y OUTPUT indica que el truncado se produce tanto en la entrada como en la salida de la Cola de tabla.
TRUNCSRT	TRUE	SORT truncado (limita el número de filas generadas).
UNIQUE	TRUE FALSE	Indicador de exclusividad.

Tabla 246. Valores de las columnas ARGUMENT_TYPE y ARGUMENT_VALUE (continuación)

Valor de ARGUMENT_TYPE	Valores de ARGUMENT_VALUE posibles	Descripción
UNIQKEY	Cada fila de este tipo contendrá: • El valor ordinal de la columna en clave (seguido por dos puntos y un espacio) • Nombre de columna	Columnas de claves de unicidad.
UR_EXTRA	TRUE	Aislamiento de lectura sin confirmar, aunque con un proceso adicional para garantizar un aislamiento correcto. Este acceso tiene un bloqueo de nivel de tabla adicional; el mismo que el de la estabilidad del cursor. Además, cuando se ejecuta la sentencia, el nivel de aislamiento se puede actualizar a la estabilidad del cursor; por ejemplo, si hay una carga en línea ejecutándose simultáneamente. Otra parte del plan de ejecución de la sentencia garantizará que el nivel de aislamiento sea correcto, como un operador FETCH en un nivel de aislamiento superior.
VISIBLE	TRUE FALSE	Indica si las exploraciones compartidas son visibles en otras exploraciones compartidas. Una exploración compartida visible puede influir en el comportamiento de otras exploraciones. Entre los comportamientos que pueden resultar afectados se encuentran la ubicación de inicio y la disminución.
VOLATILE	TRUE	Tabla volátil
WRAPPING	TRUE FALSE	Si se permite que una exploración compartida empiece en algún registro de la tabla y se reinicie una vez al alcanzar el último registro. El reinicio permite compartir los registros de agrupaciones de almacenamientos intermedios con otras exploraciones en curso.
XDFOUT	DECIMAL	XDFOUT indica el número esperado de documentos que devolverá el operador XISCAN para cada nodo de contexto.
XLOGID	Un identificador que consiste en un nombre de esquema SQL y el nombre de un índice sobre datos XML	XLOGID identifica el índice sobre datos XML elegido por el optimizador para el operador XISCAN.
XPATH	Una expresión XPATH y el conjunto de resultados en un formato interno	Este argumento indica la evaluación de una expresión XPATH del operador XSCAN.
XPHYID	Un identificador que consiste en un nombre de esquema SQL y el nombre de un índice sobre datos XML físico	XPHYID identifica el índice físico asociado con un índice sobre datos XML que utiliza el operador XISCAN.

Tabla EXPLAIN_DIAGNOSTIC

La tabla EXPLAIN_DIAGNOSTIC contiene una entrada para cada mensaje de diagnóstico producido para una instancia concreta de una sentencia explicada en la tabla EXPLAIN_STATEMENT.

La función de tabla EXPLAIN_GET_MSGS consulta las tablas Explain EXPLAIN_DIAGNOSTIC y EXPLAIN_DIAGNOSTIC_DATA y devuelve mensajes con formato.

Tabla 247. Tabla EXPLAIN_DIAGNOSTIC. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK, FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK, FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK, FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK, FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	PK, FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	PK, FK	Nivel de información de Explain para el que esta fila es aplicable. Los valores válidos son: **O** Texto original (introducido por el usuario) **P** PLAN SELECTION
STMTNO	INTEGER	No	PK, FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain. Establecido en 1 para las sentencias de SQL dinámico de Explain. En el caso de sentencias de SQL estático, este valor es el mismo que el utilizado para la vista de catálogo del sistema SYSCAT.STATEMENTS.
SECTNO	INTEGER	No	PK, FK	El número de sección en el paquete que contiene esta sentencia de SQL. En el caso de sentencias de SQL dinámico de Explain, éste es el número de sección utilizado para mantener la sección de esta sentencia en tiempo de ejecución. En el caso de sentencias de SQL estático, este valor es el mismo que el utilizado para la vista de catálogo del sistema SYSCAT.STATEMENTS.
DIAGNOSTIC_ID	INTEGER	No	PK	ID del diagnóstico para una instancia concreta de una sentencia en la tabla EXPLAIN_STATEMENT.
CODE	INTEGER	No	No	Número exclusivo asignado a cada mensaje de diagnóstico. Una API de mensajes puede utilizar el número para recuperar el texto completo del mensaje de diagnóstico.

Tabla EXPLAIN_DIAGNOSTIC_DATA

La tabla EXPLAIN_DIAGNOSTIC_DATA contiene símbolos para mensajes de diagnóstico determinados que se registran en la tabla EXPLAIN_DIAGNOSTIC. Los símbolos de mensajes ofrecen información adicional específica para la ejecución de la sentencia SQL que ha generado el mensaje.

La función de tabla EXPLAIN_GET_MSGS consulta las tablas Explain EXPLAIN_DIAGNOSTIC y EXPLAIN_DIAGNOSTIC_DATA, y devuelve los mensajes con formato.

Tabla 248. Tabla EXPLAIN_DIAGNOSTIC_DATA. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable. Los valores válidos son: **O** Texto original (introducido por el usuario) **P** PLAN SELECTION
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain. Establecido en 1 para las sentencias de SQL dinámico de Explain. En el caso de sentencias de SQL estático, este valor es el mismo que el utilizado para la vista de catálogo del sistema SYSCAT.STATEMENTS.
SECTNO	INTEGER	No	FK	El número de sección en el paquete que contiene esta sentencia de SQL. En el caso de sentencias de SQL dinámico de Explain, éste es el número de sección utilizado para mantener la sección de esta sentencia en tiempo de ejecución. En el caso de sentencias de SQL estático, este valor es el mismo que el utilizado para la vista de catálogo del sistema SYSCAT.STATEMENTS.
DIAGNOSTIC_ID	INTEGER	No	PK	ID del diagnóstico para una instancia concreta de una sentencia en la tabla EXPLAIN_STATEMENT.
ORDINAL	INTEGER	No	No	Posición del símbolo en el texto de mensaje completo.
TOKEN	VARCHAR(1000)	Sí	No	Símbolo de mensaje que insertar en el texto de mensaje completo; es posible que esté truncado.
TOKEN_LONG	BLOB(3M)	Sí	No	Más información detallada, si está disponible.

Tabla EXPLAIN_INSTANCE

La tabla EXPLAIN_INSTANCE es la tabla principal de control para toda la información de Explain. Cada fila de datos de las tablas de Explain se enlaza explícitamente con una fila exclusiva de esta tabla. La tabla EXPLAIN_INSTANCE proporciona la información básica acerca de la fuente de las sentencias de SQL que se están explicando, así como la información acerca del entorno en el que ha tenido lugar la explicación.

Tabla 249. Tabla EXPLAIN_INSTANCE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	PK	Versión de la fuente de la petición de Explain.
EXPLAIN_OPTION	CHAR(1)	No	No	Indica que se ha pedido la información de Explain para esta petición. Los valores posibles son: **P** PLAN SELECTION **S** Sección Explain
SNAPSHOT_TAKEN	CHAR(1)	No	No	Indica si se ha tomado una instantánea de Explain para esta petición. Los valores posibles son: **S** Sí, se ha(n) tomado una(s) instantánea(s) de Explain y se ha(n) almacenado en la tabla EXPLAIN_STATEMENT. También se ha capturado información de Explain normal. **N** No se ha tomado ninguna instantánea de Explain. Se ha capturado información de Explain normal. **O** Sólo se ha tomado una instantánea de Explain. No se ha capturado información de Explain normal.
DB2_VERSION	CHAR(7)	No	No	Número de release del producto DB2 que procesó esta petición de explicación. El formato es *vv.rr.m*, donde: **vv** Número de versión **rr** Número de release **m** Número de release de mantenimiento

Tabla 249. Tabla EXPLAIN_INSTANCE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
SQL_TYPE	CHAR(1)	No	No	Indica si la instancia de Explain era para SQL dinámico o estático. Los valores posibles son: **S** SQL estático **D** SQL dinámico
QUERYOPT	INTEGER	No	No	Indica la clase de optimización de consulta que ha utilizado el Compilador SQL en el momento de la invocación de Explicar. El valor indica qué nivel de optimización de consulta ha efectuado el Compilador SQL para las sentencias de SQL que se explican.
BLOCK	CHAR(1)	No	No	Indica qué tipo de bloqueo de cursor se ha utilizado al compilar las sentencias de SQL. Para obtener información, consulte la columna BLOCK de SYSCAT.PACKAGES. Los valores posibles son: **N** Sin bloqueo **U** Bloqueo de cursores no ambiguos **B** Bloqueo de todos los cursores
ISOLATION	CHAR(2)	No	No	Indica qué tipo de aislamiento se ha utilizado al compilar las sentencias de SQL. Para obtener más información, consulte la columna ISOLATION en SYSCAT.PACKAGES. Los valores posibles son: **RR** Lectura repetible **RS** Estabilidad de lectura **CS** Estabilidad del cursor **UR** Lectura no confirmada
BUFFPAGE	INTEGER	No	No	Contiene el valor de configuración de base de datos BUFFPAGE en el momento de la invocación de Explicar.
AVG_APPLS	INTEGER	No	No	Contiene el valor del parámetro de configuración **avg_appls** en el momento de la invocación de Explicar.
SORTHEAP	INTEGER	No	No	Contiene el valor del parámetro de configuración de la base de datos **sortheap** en el momento de la invocación de Explicar.
LOCKLIST	INTEGER	No	No	Contiene el valor del parámetro de configuración de la base de datos **locklist** en el momento de la invocación de Explicar.
MAXLOCKS	SMALLINT	No	No	Contiene el valor del parámetro de configuración de la base de datos **maxlocks** en el momento de la invocación de Explicar.

Tabla EXPLAIN_INSTANCE

Tabla 249. Tabla EXPLAIN_INSTANCE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
LOCKS_AVAIL	INTEGER	No	No	Contiene el número de bloqueos que el optimizador supone que están disponibles para cada usuario. (Derivado de **locklist** y **maxlocks**.)
CPU_SPEED	DOUBLE	No	No	Contiene el valor del parámetro de configuración del gestor de base de datos **cpuspeed** en el momento de la invocación de Explicar.
REMARKS	VARCHAR(254)	Sí	No	Comentario suministrado por el usuario.
DBHEAP	INTEGER	No	No	Contiene el valor del parámetro de configuración de la base de datos **dbheap** en el momento de la invocación de Explicar.
COMM_SPEED	DOUBLE	No	No	Contiene el valor del parámetro de configuración de la base de datos **comm_bandwidth** en el momento de la invocación de Explicar.
PARALLELISM	CHAR(2)	No	No	Los valores posibles son: • N = Sin paralelismo • P = Paralelismo intrapartición • IP = Paralelismo interparticiones • BP = Paralelismo intrapartición y paralelismo interparticiones
DATAJOINER	CHAR(1)	No	No	Los valores posibles son: • N = Plan de sistemas no federados • Y = Plan de sistemas federados

Tabla EXPLAIN_OBJECT

La tabla EXPLAIN_OBJECT identifica los objetos de datos que necesita el plan de acceso generado para satisfacer la sentencia de SQL.

Tabla 250. Tabla EXPLAIN_OBJECT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OBJECT_SCHEMA	VARCHAR(128)	No	No	Esquema al que pertenece este objeto.
OBJECT_NAME	VARCHAR(128)	No	No	Nombre del objeto.
OBJECT_TYPE	CHAR(2)	No	No	Etiqueta descriptiva del tipo de objeto.
CREATE_TIME	TIMESTAMP	Sí	No	Hora de creación del objeto; nulo si es una función de tabla.
STATISTICS_TIME	TIMESTAMP	Sí	No	Última vez que se actualizaron las estadísticas para este objeto; nulo si no existen estadísticas para este objeto.
COLUMN_COUNT	SMALLINT	No	No	El número de columnas en este objeto.
ROW_COUNT	INTEGER	No	No	El número estimado de filas en este objeto.
WIDTH	INTEGER	No	No	La anchura media del objeto en bytes. Se establece en -1 para un índice.
PAGES	BIGINT	No	No	El número estimado de páginas que ocupa el objeto en la agrupación de almacenamientos intermedios. Establecido en -1 para una función de tabla.
DISTINCT	CHAR(1)	No	No	Indica si las filas del objeto son diferenciadas (es decir, si hay duplicados) Los valores posibles son: **S** Sí **N** No
TABLESPACE_NAME	VARCHAR(128)	Sí	No	Nombre del espacio de tablas en el que está almacenado el objeto; se establece en nulo si no está implicado ningún espacio de tablas.

Tabla 250. Tabla EXPLAIN_OBJECT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
OVERHEAD	DOUBLE	No	No	Actividad general total estimada, en milisegundos, para una sola operación de E/S aleatoria para un espacio de tablas especificado. Incluye la actividad general del controlador, la búsqueda de disco y los tiempos de latencia. Se establece en -1 si no está implicado ningún espacio de tablas.
TRANSFER_RATE	DOUBLE	No	No	Tiempo estimado para leer una página de datos, en milisegundos, del espacio de tablas especificado. Se establece en -1 si no está implicado ningún espacio de tablas.
PREFETCHSIZE	INTEGER	No	No	El número de páginas de datos que se han de leer cuando se efectúa una lectura anticipada. Establecido en -1 para una función de tabla.
EXTENTSIZE	INTEGER	No	No	El tamaño de extensión, en páginas de datos. Este volumen de páginas se escribe en un contenedor individual del espacio de tablas antes de cambiar al contenedor siguiente. Establecido en -1 para una función de tabla.
CLUSTER	DOUBLE	No	No	Nivel de clúster de los datos con el índice. Si >= 1, es el CLUSTERRATIO. Si >= 0 y < 1, es el CLUSTERFACTOR. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
NLEAF	BIGINT	No	No	El número de páginas que ocupan los valores de estos objetos de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
NLEVELS	INTEGER	No	No	El número de niveles de índice del árbol de este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
FULLKEYCARD	BIGINT	No	No	El número de valores de clave completa diferenciada contenidos en este objeto de índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
OVERFLOW	BIGINT	No	No	El número total de registros de desbordamiento en la tabla. Se establece en -1 para un índice, una función de tabla o si no está disponible esta estadística.
FIRSTKEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
FIRST2KEYCARD	BIGINT	No	No	El número de valores de primera clave diferenciada utilizando las primeras columnas {2,3,4} del índice. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
FIRST3KEYCARD	BIGINT	No	No	
FIRST4KEYCARD	BIGINT	No	No	

Tabla 250. Tabla EXPLAIN_OBJECT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
SEQUENTIAL_PAGES	BIGINT	No	No	El número de páginas ubicadas en disco por orden de clave de índice con pocos o ningún vacío entre ellas. Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
DENSITY	INTEGER	No	No	Proporción de SEQUENTIAL_PAGES en relación al número de páginas del rango de páginas ocupado por el índice, expresada como porcentaje (entero entre 0 y 100). Se establece en -1 para una tabla, una función de tabla o si no está disponible esta estadística.
STATS_SRC	CHAR(1)	No	No	Indica la fuente para las estadísticas. Se establece en 1 si se trata de un solo nodo.
AVERAGE_SEQUENCE_GAP	DOUBLE	No	No	Espacio entre secuencias.
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	No	No	Espacio entre las secuencias al captar utilizando el índice.
AVERAGE_SEQUENCE_PAGES	DOUBLE	No	No	Promedio de páginas del índice accesibles en secuencia.
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	No	No	Promedio de páginas de la tabla accesibles en secuencia al captar utilizando el índice.
AVERAGE_RANDOM_PAGES	DOUBLE	No	No	Promedio de páginas del índice aleatorias entre los accesos a páginas secuenciales.
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	No	No	Promedio de páginas de la tabla aleatorias entre los accesos a páginas secuenciales al captar utilizando el índice.
NUMRIDS	BIGINT	No	No	Número total de identificadores de filas del índice.
NUMRIDS_DELETED	BIGINT	No	No	Número total de identificadores de filas pseudosuprimidos del índice.
NUM_EMPTY_LEAFS	BIGINT	No	No	Número total de páginas hojas vacías del índice.
ACTIVE_BLOCKS	BIGINT	No	No	Número total de bloques de clúster multidimensional (MDC) activos de la tabla.
NUM_DATA_PART	INTEGER	No	No	Número de particiones de datos para una tabla particionada. Se establece en 1 si la tabla no está particionada.

Tabla 251. Valores de OBJECT_TYPE posibles

Valor	Descripción
IX	Índice
NK	Apodo
RX	Índice RCT
DP	Tabla de partición de datos
TA	Tabla
TF	Función de tabla

Tabla 251. Valores de OBJECT_TYPE posibles (continuación)

Valor	Descripción
+A	Alias de referencia del compilador
+C	Restricción de referencia del compilador
+F	Función de referencia del compilador
+G	Activador de referencia del compilador
+N	Apodo de referencia del compilador
+T	Tabla de referencia del compilador
+V	Vista de referencia del compilador
XI	Índice XML lógico
PI	Índice XML físico
LI	Índice particionado
LX	Índice XML lógico particionado
LP	Índice XML físico particionado

Tabla EXPLAIN_OPERATOR

La tabla EXPLAIN_OPERATOR contiene todos los operadores necesarios para que el compilador de consultas satisfaga la sentencia de consulta.

Tabla 252. Tabla EXPLAIN_OPERATOR. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	PK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	PK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	PK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	PK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	PK	ID exclusivo para este operador en esta consulta.
OPERATOR_TYPE	CHAR(6)	No	No	Etiqueta descriptiva para el tipo de operador.
TOTAL_COST	DOUBLE	No	No	Coste total acumulado estimado (en timerons) de la ejecución del plan de acceso elegido hasta este operador inclusive.
IO_COST	DOUBLE	No	No	Coste de E/S acumulado estimado (en E/S de páginas de datos) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
CPU_COST	DOUBLE	No	No	Coste de CPU acumulado estimado (en las instrucciones) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
FIRST_ROW_COST	DOUBLE	No	No	Coste acumulado estimado (en timerons) de la lectura de la primera fila para el plan de acceso hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria.
RE_TOTAL_COST	DOUBLE	No	No	Coste acumulado estimado (en timerons) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.
RE_IO_COST	DOUBLE	No	No	Coste de E/S acumulado estimado (en E/S de páginas de datos) de la lectura de la siguiente fila del plan de acceso elegido hasta este operador inclusive.
RE_CPU_COST	DOUBLE	No	No	Coste de CPU acumulado estimado (en instrucciones) de la lectura de la siguiente fila para el plan de acceso elegido hasta este operador inclusive.

Tabla 252. Tabla EXPLAIN_OPERATOR (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
COMM_COST	DOUBLE	No	No	Coste de comunicación acumulado estimado (en tramas TCP/IP) de la ejecución del plan de acceso elegido hasta este operador (inclusive).
FIRST_COMM_COST	DOUBLE	No	No	Coste de comunicaciones acumulado estimado (en TCP/IP) de la lectura de la primera fila para el plan de acceso elegido hasta este operador inclusive. Este valor incluye cualquier actividad general inicial necesaria.
BUFFERS	DOUBLE	No	No	Requisitos estimados de almacenamiento intermedio para este operador y sus entradas.
REMOTE_TOTAL_COST	DOUBLE	No	No	Coste total acumulado estimado (en timerons) de la ejecución de operación(es) en base(s) de datos remota(s).
REMOTE_COMM_COST	DOUBLE	No	No	Coste de comunicación acumulado estimado de la ejecución del plan de acceso remoto elegido hasta este operador (inclusive).

Tabla 253. Valores de OPERATOR_TYPE

Valor	Descripción
DELETE	Suprimir
EISCAN	Exploración de índice ampliada
FETCH	Leer
FILTER	Filtrar filas
GENROW	Generar fila
GRPBY	Agrupar por
HSJOIN	Unión de generación aleatoria
INSERT	Insertar
IXAND	Aplicación de AND de Índice de mapas de bits dinámico
IXSCAN	Exploración del índice relacional
MSJOIN	Fusionar unión de exploración
NLJOIN	Unión de bucle anidado
RETURN	Resultado
RIDSCN	Exploración de identificador de fila (RID)
RPD	Desplazamiento descendente remoto
SHIP	Enviar consulta a sistema remoto
SORT	Clasificar
TBSCAN	Exploración de tabla
TEMP	Construcción de tabla temporal
TQ	Cola de tabla
UNION	Unión
UNIQUE	Eliminación de duplicados
UPDATE	Actualizar

Tabla 253. Valores de OPERATOR_TYPE (continuación)

Valor	Descripción
XISCAN	Exploración del índice sobre datos XML
XSCAN	Exploración mediante navegación de documento XML
XANDOR	Añadir AND u OR al índice sobre datos XML

Tabla EXPLAIN_PREDICATE

La tabla EXPLAIN_PREDICATE identifica los predicados que aplica un operador específico.

Tabla 254. Tabla EXPLAIN_PREDICATE. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
OPERATOR_ID	INTEGER	No	No	ID exclusivo para este operador en esta consulta.
PREDICATE_ID	INTEGER	No	No	ID exclusivo de este predicado para el operador especificado. Se muestra el valor ″-1″ en los predicados de operador creados mediante la herramienta Explain que no son objetos del optimizador y que no existen en el plan de éste.
HOW_APPLIED	CHAR(10)	No	No	La forma en que el operador especificado utiliza el predicado.

Tabla 254. Tabla EXPLAIN_PREDICATE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
WHEN_EVALUATED	CHAR(3)	No	No	Indica cuándo se evalúa la subconsulta utilizada en este predicado. Los valores posibles son: **en blanco** Este predicado no contiene ninguna subconsulta. **EAA** La subconsulta utilizada en este predicado se evalúa en la aplicación (EAA). Es decir, se vuelve a evaluar para cada fila procesada por el operador especificado, cuando se aplica el predicado. **EAO** La subconsulta utilizada en este predicado se evalúa en la apertura (EAO). Es decir, se vuelve a evaluar sólo una vez para el operador especificado y sus resultados se vuelven a utilizar en la aplicación del predicado para cada fila. **MUL** Hay más de una subconsulta en este predicado.
RELOP_TYPE	CHAR(2)	No	No	El tipo de operador relacional utilizado en este predicado.
SUBQUERY	CHAR(1)	No	No	Si es necesaria una corriente de datos de una subconsulta o no para este predicado. Puede ser necesarias múltiples corrientes de subconsultas. Los valores posibles son: **N** No es necesaria ninguna corriente de subconsulta **S** Son necesarias una o varias corrientes de subconsultas
FILTER_FACTOR	DOUBLE	No	No	La fracción estimada de filas que este predicado calificará. Se muestra el valor ″-1″ cuando no se puede aplicar FILTER_FACTOR. FILTER_FACTOR no se puede aplicar en los predicados de operador creados mediante la herramienta Explain que no son objetos del optimizador y no existen en el plan de éste.

Tabla EXPLAIN_PREDICATE

Tabla 254. Tabla EXPLAIN_PREDICATE (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
PREDICATE_TEXT	CLOB(2M)	Sí	No	El texto del predicado tal como se ha vuelto a crear a partir de la representación interna de la sentencia de SQL o XQuery. Si se utiliza el valor de una variable del lenguaje principal, un registro especial o un marcador de parámetro durante la compilación de la sentencia, este valor aparecerá al final del texto del predicado, en un comentario. El valor sólo se almacenará en la tabla EXPLAIN_PREDICATE si la sentencia la ejecuta un usuario que cuente con autorización DBADM o si la variable de registro de DB2 DB2_VIEW_REOPT_VALUES está definida en YES; en caso contrario, aparecerá un comentario vacío al final del texto del predicado. Nulo si no está disponible.
RANGE_NUM	INTEGER	Sí	No	Rango de predicados de eliminación de particiones de datos, que permite la agrupación según el rango de los predicados que se utilizan para la eliminación de particiones de datos. Valor nulo para todos los demás tipos de predicado.

Tabla 255. Valores de HOW_APPLIED posibles

Valor	Descripción
BSARG	Evaluado como un predicado comparable una vez para cada bloque
DPSTART	Predicado de clave de inicio utilizado en la eliminación de particiones de datos
DPSTOP	Predicado de clave de parada utilizado en la eliminación de particiones de datos
JOIN	Utilizado para unir tablas
RESID	Evaluado como un predicado residual
SARG	Evaluado como un predicado comparable para un índice o página de datos
START	Utilizado como una condición de inicio
STOP	Utilizado como una condición de detención

Tabla 256. Valores de RELOP_TYPE posibles

Valor	Descripción
blancos	No aplicable
EQ	Igual
GE	Mayor o igual que
GT	Mayor que
IN	En lista
LE	Menor o igual que
LK	Igual
LT	Menor que
NE	Diferente a

Tabla 256. Valores de RELOP_TYPE posibles (continuación)

Valor	Descripción
NL	Es nulo
NN	No es nulo

Tabla EXPLAIN_STATEMENT

La tabla EXPLAIN_STATEMENT contiene el texto de la sentencia de SQL tal como existe para los diferentes niveles de información de Explain. La sentencia de SQL original se almacena tal como la entra el usuario, en esta tabla junto con la versión utilizada (por el optimizador) para elegir el plan de acceso para satisfacer la sentencia de SQL. La última versión puede parecerse poco a la original ya que puede haberse vuelto a escribir y/o mejorar con predicados adicionales tal como lo determina el Compilador SQL.Además, si se habilita el concentrador de sentencias y la sentencia se ha cambiado como resultado del concentrador de sentencias, la sentencia de SQL efectiva también se almacenará en esta tabla. Esta sentencia será similar a la sentencia original, excepto en el hecho de que los valores literales se sustituirán con marcadores de parámetro con nombre generados. La información del plan se basará en la sentencia efectiva en este caso.

Tabla 257. Tabla EXPLAIN_STATEMENT. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	PK, FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	PK, FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	PK, FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	PK, FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	PK	Nivel de información de Explain para el que esta fila es aplicable. Los valores válidos son: **E** Texto de SQL efectivo **O** Texto original (tal como lo ha entrado el usuario) **P** PLAN SELECTION **S** Sección Explain
STMTNO	INTEGER	No	PK	El número de sentencia en el paquete con el que está relacionado esta información de Explain. Establecido en 1 para las sentencias de SQL dinámico de Explain. Para las sentencias de SQL estático, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.
SECTNO	INTEGER	No	PK	El número de sección en el paquete que contiene esta sentencia de SQL. En las sentencias de SQL dinámico de Explain, es el número de sección utilizada para contener la sección para esta sentencia en tiempo de ejecución. Para las sentencias de SQL estático, este valor es igual al valor utilizado para la vista de catálogo SYSCAT.STATEMENTS.

Tabla 257. Tabla EXPLAIN_STATEMENT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
QUERYNO	INTEGER	No	No	Identificador numérico para la sentencia de SQL explicada. Para sentencias de SQL dinámico (excluyendo la sentencia EXPLAIN SQL) emitidas a través de CLP o CLI, el valor por omisión es un valor incrementado secuencialmente. De lo contrario, el valor por omisión es el valor de STMTNO para sentencias de SQL estático y 1 para sentencias de SQL dinámico.
QUERYTAG	CHAR(20)	No	No	Distintivo identificador para cada sentencia de SQL explicada. Para sentencias de SQL dinámico emitidas a través de CLP (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLP'. Para sentencias de SQL dinámico emitidas a través de CLI (excluida la sentencia EXPLAIN SQL), el valor por omisión es 'CLI'. De lo contrario, el valor por omisión utilizado es blancos.
STATEMENT_TYPE	CHAR(2)	No	No	Etiqueta descriptiva para el tipo de consulta que se está explicando. Los valores posibles son: **CL** Llamada **CP** SQL compuesto (Dinámico) **D** Suprimir **DC** Supresión en la ubicación actual del cursor **I** Insertar **M** Fusionar **S** Selección **SI** Establecer integridad o Renovar tabla **U** Actualizar **UC** Actualización en la ubicación actual del cursor
UPDATABLE	CHAR(1)	No	No	Indica si esta sentencia se considera actualizable. Es relevante en particular para las sentencias SELECT que pueden determinarse como actualizables en potencia. Los valores posibles son: **' '** No aplicable (blanco) **N** No **S** Sí

Tabla EXPLAIN_STATEMENT

Tabla 257. Tabla EXPLAIN_STATEMENT (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
DELETABLE	CHAR(1)	No	No	Indica si esta sentencia se considera suprimible. Es relevante en particular para las sentencias SELECT que pueden determinarse como suprimibles en potencia. Los valores posibles son: ' ' No aplicable (blanco) **N** No **S** Sí
TOTAL_COST	DOUBLE	No	No	Coste calculado total (en timerons) de la ejecución del plan de acceso elegido para esta sentencia; establézcalo en 0 (cero) si EXPLAIN_LEVEL es `0` o `E` (texto original o efectivo) ya que no hay ningún plan de acceso elegido en este momento.
STATEMENT_TEXT	CLOB(2M)	No	No	Texto o fragmento del texto de la sentencia de SQL que se está explicando. El texto que se muestra para los niveles de Explain de Selección del plan o Sección se ha reconstruido a partir de la representación interna y es de tipo SQL por naturaleza; es decir, no se garantiza que la sentencia reconstruida siga la sintaxis de SQL correcta.
SNAPSHOT	BLOB(10M)	Sí	No	Instantánea de la representación interna de esta sentencia de SQL en el Nivel_Explain mostrado. Esta columna está pensada para utilizarla con DB2 Visual Explain. La columna se establece en NULL si EXPLAIN_LEVEL no es `P` (sentencia Plan), pues no se había elegido ningún plan de acceso en el momento en que se capturó esta versión específica de la sentencia.
QUERY_DEGREE	INTEGER	No	No	Indica el grado de paralelismo intrapartición en el momento de la invocación de Explicar. Para la sentencia original, contiene el grado dirigido de paralelismo intrapartición. Si no, contiene el grado de paralelismo intrapartición generado para que lo utilice el plan.

Tabla EXPLAIN_STREAM

La tabla EXPLAIN_STREAM representa las corrientes de datos de entrada y de salida entre operadores individuales y objetos de datos. Los objetos de datos en sí se representan en la tabla EXPLAIN_OBJECT. Los operadores implicados en una corriente de datos se han de encontrar en una tabla EXPLAIN_OPERATOR.

Tabla 258. Tabla EXPLAIN_STREAM. PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
EXPLAIN_REQUESTER	VARCHAR(128)	No	FK	ID de autorización del iniciador de esta petición de Explain.
EXPLAIN_TIME	TIMESTAMP	No	FK	Hora de inicio de la petición de Explain.
SOURCE_NAME	VARCHAR(128)	No	FK	Nombre del paquete que se ejecutaba cuando se ha explicado la sentencia dinámica o nombre del archivo fuente cuando se ha explicado la sentencia de SQL estático.
SOURCE_SCHEMA	VARCHAR(128)	No	FK	Esquema, o calificador, de la fuente de la petición de Explain.
SOURCE_VERSION	VARCHAR(64)	No	FK	Versión de la fuente de la petición de Explain.
EXPLAIN_LEVEL	CHAR(1)	No	FK	Nivel de información de Explain para el que esta fila es aplicable.
STMTNO	INTEGER	No	FK	El número de sentencia en el paquete con el que está relacionado esta información de Explain.
SECTNO	INTEGER	No	FK	El número de sección en el paquete con el que está relacionada esta información de Explain.
STREAM_ID	INTEGER	No	No	ID exclusivo para esta corriente de datos en el operador especificado.
SOURCE_TYPE	CHAR(1)	No	No	Indica la fuente de la corriente de datos: **O** Operador **D** Objeto de datos
SOURCE_ID	SMALLINT	No	No	ID exclusivo para el operador dentro de esta consulta que es la fuente de esta corriente de datos. Se establece en -1 si SOURCE_TYPE es 'D'.
TARGET_TYPE	CHAR(1)	No	No	Indica el destino de la corriente de datos: **O** Operador **D** Objeto de datos
TARGET_ID	SMALLINT	No	No	ID exclusivo para el operador dentro de esta consulta que es el destino de esta corriente de datos. Se establece en -1 si TARGET_TYPE es 'D'.
OBJECT_SCHEMA	VARCHAR(128)	Sí	No	El esquema al que pertenece el objeto de datos afectado. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.
OBJECT_NAME	VARCHAR(128)	Sí	No	Nombre del objeto que es el sujeto de la corriente de datos. Se establece en nulo si SOURCE_TYPE y TARGET_TYPE son 'O'.
STREAM_COUNT	DOUBLE	No	No	Cardinalidad estimada de la corriente de datos.
COLUMN_COUNT	SMALLINT	No	No	El número de columnas en la corriente de datos.

Tabla EXPLAIN_STREAM

Tabla 258. Tabla EXPLAIN_STREAM (continuación). PK significa que la columna forma parte de una clave primaria; FK significa que la columna forma parte de una clave foránea.

Nombre de columna	Tipo de datos	¿Posibil. de nulos?	¿Clave?	Descripción
PREDICATE_ID	INTEGER	No	No	Si la corriente forma parte de una subconsulta para un predicado, el ID del predicado se reflejará aquí, de lo contrario la columna se establece en -1.
COLUMN_NAMES	CLOB(2M)	Sí	No	Esta columna contiene los nombres y la información de ordenación de las columnas implicadas en esta corriente. Estos nombres estarán en el formato de: `NOMBRE1(A)+NOMBRE2(D)+NOMBRE3+NOMBRE4` Donde *(A)* indica una columna por orden ascendente, *(D)* indica una columna por orden descendente y ninguna información de ordenación indica que la columna no está ordenada o que el orden no es relevante.
PMID	SMALLINT	No	No	ID de correlación de distribución.
SINGLE_NODE	CHAR(5)	Sí	No	Indica si este flujo de datos está en una partición de base de datos única o en varias: **MULT** En varias particiones de base de datos **COOR** En nodo del coordinador **HASH** Dirigido utilizando generación aleatoria **RID** Dirigido utilizando el ID de fila **FUNC** Dirigido utilizando una función (HASHEDVALUE() o DBPARTITIONNUM()) **CORR** Dirigido utilizando un valor de correlación **Numérico** Dirigido hacia un nodo individual predeterminado
PARTITION_COLUMNS	CLOB(2M)	Sí	No	Lista de las columnas en las que este flujo de datos se distribuye.
SEQUENCE_SIZES	CLOB(2M)	Sí	No	Muestra el tamaño de la secuencia que se esperaba para las columnas XML o muestra ″NA″ (no aplicable) para las columnas que no sean XML del flujo de datos. Establézcalo como nulo si al menos no hay una columna XML en el flujo de datos.

Apéndice J. Valores de los registros de EXPLAIN

A continuación se proporciona una descripción de la interacción de los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT, entre sí y con los mandatos PREP y BIND.

Con SQL dinámico, los valores de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT interactúan de la siguiente manera:

Tabla 259. Interacción de los valores de los registros especiales EXPLAIN (SQL dinámico)

Valores de EXPLAIN SNAPSHOT	Valores de EXPLAIN MODE					
	NO	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
NO	• Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de Explain. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.
YES	• Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.

Tabla 259. Interacción de los valores de los registros especiales EXPLAIN (SQL dinámico) (continuación)

Valores de EXPLAIN SNAPSHOT	Valores de EXPLAIN MODE					
	NO	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
EXPLAIN	• Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan).	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de Explain. • Se toma una Instantánea de explicación. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.
REOPT	• Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan).	• Las tablas de Explain se rellenan cuando una sentencia está calificada para su reoptimización durante la ejecución. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se rellenan las tablas de Explain. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan). • Se recomiendan los índices.	• Se rellenan las tablas de Explain. • Instantánea de explicación tomada cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas o de vinculación incremental no se ejecutan). • Se evalúan los índices.

El registro especial CURRENT EXPLAIN MODE interactúa con la opción de vinculación EXPLAIN de la siguiente manera en SQL dinámico.

Tabla 260. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	**YES**	**REOPT**	**ALL**
NO	• Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.
YES	• Las tablas de Explain se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Se devuelven los resultados de la consulta.

Tabla 260. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE (continuación)

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	YES	REOPT	ALL
EXPLAIN	• Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).
REOPT	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.

Tabla 260. Interacción de la opción de vinculación EXPLAIN y CURRENT EXPLAIN MODE (continuación)

Valores de EXPLAIN MODE	Valores de la opción de vinculación EXPLAIN			
	NO	**YES**	**REOPT**	**ALL**
RECOMMEND INDEXES	• Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se recomiendan los índices.
EVALUATE INDEXES	• Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Las tablas de Explain se rellenan para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.	• Las tablas de Explain se rellenan para SQL estático. • Las tablas de Explain se rellenan para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan). • Se evalúan los índices.

El registro especial CURRENT EXPLAIN SNAPSHOT interactúa con la opción de vinculación EXPLSNAP de la siguiente manera para SQL dinámico.

Tabla 261. Interacción de la opción de vinculación EXPLSNAP y CURRENT EXPLAIN SNAPSHOT

Valores de EXPLAIN SNAPSHOT	Valores de la opción de vinculación EXPLSNAP			
	NO	**YES**	**REOPT**	**ALL**
NO	• Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.
YES	• Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Se devuelven los resultados de la consulta.

Tabla 261. Interacción de la opción de vinculación EXPLSNAP y CURRENT EXPLAIN SNAPSHOT (continuación)

Valores de EXPLAIN SNAPSHOT	Valores de la opción de vinculación EXPLSNAP			
	NO	YES	REOPT	ALL
EXPLAIN	• Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).	• Se toma una Instantánea de explicación para SQL estático. • Se toma una Instantánea de explicación para SQL dinámico. • Los resultados de la consulta no se devuelven (las sentencias dinámicas no se ejecutan).
REOPT	• Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.	• Se toma una Instantánea de explicación para SQL estático cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se toma una Instantánea de explicación para SQL dinámico cuando una sentencia está calificada para su reoptimización durante la ejecución. • Se devuelven los resultados de la consulta.

Apéndice K. Tablas de excepciones

Las tablas de excepciones son tablas creadas por el usuario que imitan la definición de las tablas cuya comprobación se especifica utilizando SET INTEGRITY con la opción IMMEDIATE CHECKED. Se utilizan para almacenar copias de las filas que violan las restricciones de las tablas que se están comprobando.

Las tablas de excepciones que utiliza el programa de utilidad de carga son idénticas a las que se describen aquí y, por lo tanto, se pueden volver a utilizar durante la comprobación de la sentencia SET INTEGRITY.

Normas para crear una tabla de excepciones

Las normas para crear una tabla de excepciones son las siguientes:

- Si la tabla está protegida por una política de seguridad, la tabla de excepciones debe protegerse mediante la misma política de seguridad.
- Las primeras "n" columnas de la tabla de excepciones son iguales que las columnas de la tabla que se está comprobando. Todos los atributos de columna, inclusive el nombre, el tipo de datos y la longitud, deben ser idénticos. En el caso de las columnas protegidas, la etiqueta de seguridad que protege a la columna debe ser la misma en ambas tablas.
- Todas las columnas de la tabla de excepciones deben estar libres de restricciones y activadores. Las restricciones incluyen la integridad referencial, las restricciones de comprobación, así como las restricciones de índice exclusivo que podrían causar errores en la inserción.
- La columna "(n+1)" de la tabla de excepciones es una columna TIMESTAMP opcional. Esto sirve para identificar las invocaciones sucesivas de la comprobación que efectúa la sentencia SET INTEGRITY en la misma tabla, si las filas de la tabla de excepciones no se han suprimido antes mediante la emisión de la sentencia SET INTEGRITY para comprobar los datos.La precisión de la indicación de fecha y hora puede ser cualquier valor de 0 a 12 y el valor asignado será el resultado del registro especial CURRENT TIMESTAMP
- La columna "(n+2)" debe ser de tipo CLOB(32K) o mayor. Esta columna es opcional pero se recomienda incluirla y se utilizará para proporcionar los nombres de las restricciones que violan los datos de la fila. Si no se proporciona esta columna (como pasaría si, por ejemplo, la tabla original tuviese el número máximo de columnas permitido), sólo se copia la fila en la que se ha detectado la violación de restricción.
- La tabla de excepciones se debe crear con las columnas "(n+1)" y "(n+2)".
- No se impone ningún nombre en particular para las columnas adicionales anteriores. No obstante, debe seguirse exactamente la especificación del tipo.
- No se permiten columnas adicionales.
- Si la tabla original tiene columnas generadas (incluida la propiedad IDENTITY), las columnas correspondientes de la tabla de excepciones no deben especificar la propiedad generada.
- Los usuarios que invocan la sentencia SET INTEGRITY para comprobar datos deben tener el privilegio SET INSERT en las tablas de excepciones.
- La tabla de excepciones no puede ser una tabla particionadas de datos, una tabla de agrupación en clúster de rangos ni una tabla desenlazada.

- La tabla de excepciones no puede ser una tabla de consulta materializada ni una tabla de etapas.
- La tabla de excepciones no puede tener ninguna tabla de consulta materializada de renovación inmediata dependiente ni ninguna tabla de etapas de propagación inmediata dependiente.

La información de la columna "mensaje" tiene la estructura siguiente:

Tabla 262. Estructura de la columna de mensajes de la tabla de excepciones

Número de campo	Contenido	Tamaño	Comentarios
1	Número de violaciones de restricción	5 bytes	Justificada por la derecha rellenada con '0'
2	Tipo de la primera violación de restricción	1 byte	'K' - Violación de restricción de comprobación 'F' - Violación de clave foránea 'G' - Violación de columna generada 'I' - Violación de índice de unicidad[a] 'D' - Violación de supresión de cascada 'P' - Violación de particionamiento de datos 'S' - Etiqueta de seguridad de fila no válida 'L' - Violación de normas de grabación de LBAC de DB2 'X' - Violación del índice de valores XML[d]
3	Longitud de restricción/columna[b] /ID de índice[c]	5 bytes	Justificada por la derecha rellenada con '0'
4	Nombre de restricción/Nombre de columna[b]/ID de índice[c]	longitud del campo anterior	
5	Separador	3 bytes	<espacio><dos puntos><espacio>
6	Tipo de la siguiente violación de restricción	1 byte	'K' - Violación de restricción de comprobación 'F' - Violación de clave foránea 'G' - Violación de columna generada 'I' - Violación de índice de unicidad 'D' - Violación de supresión de cascada 'P' - Violación de particionamiento de datos 'S' - Etiqueta de seguridad de fila no válida 'L' - Violación de normas de grabación de LBAC de DB2 'X' - Violación del índice de valores XML[d]
7	Longitud de restricción/columna/ID de índice	5 bytes	Justificada por la derecha rellenada con '0'
8	Nombre de restricción/Nombre de columna/ID de índice	longitud del campo anterior	
.....	.....	.....	Repita del Campo 5 al 8 para cada violación

Tabla 262. Estructura de la columna de mensajes de la tabla de excepciones (continuación)

Número de campo	Contenido	Tamaño	Comentarios

- [a] No se producirán violaciones de índices exclusivos en la comprobación si se utiliza la sentencia SET INTEGRITY, a no ser que se realice después de una operación de enlace. Sin embargo, se informará de esto, cuando se ejecute LOAD si se elige la opción FOR EXCEPTION. Por otra parte, LOAD no informará de las violaciones de restricción de comprobación, de columna generada, de clave foránea, de supresión de cascada ni de particionamiento de datos ocurridas en las tablas de excepciones.
- [b] Para recuperar la expresión de una columna generada a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es MYSCHEMA.MYTABLE.GEN_1, entonces SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; devuelve los primeros 50 bytes de la expresión, en el formato "AS (<expresión>)"
- [c] Para recuperar un ID de índice a partir de las vistas de catálogo, utilice una sentencia de selección. Por ejemplo, si el campo 4 es 1234, entonces SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234.
- [d] Para las violaciones del índice de valores XML, el nombre de restricción, el nombre de columna o el campo ID de índice identifican la columna XML que contiene una violación de integridad en uno de sus índices. No identifica el índice que contiene la violación de la integridad. Sólo identifica el nombre de la columna XML en la que se produce la violación del índice. Por ejemplo, el valor 'X00006XTCOL2' en la columna del mensaje indica que se ha producido una violación de índice en uno de los índices en la columna XTCOL2.

Gestión de las filas en una tabla de excepciones

La información de las tablas de excepciones se puede procesar de varias formas. Se pueden corregir datos y volver a insertar filas en las tablas originales.

Si no hay ningún activador INSERT en la tabla original, transfiera las filas corregidas emitiendo la sentencia INSERT con una subconsulta en la tabla de excepciones.

Si hay activadores INSERT y desea completar la operación de carga con las filas corregidas de las tablas de excepciones sin disparar los activadores:

- Diseñe los activadores INSERT para que se disparen dependiendo del valor de una columna definida explícitamente para esta finalidad.
- Descargue los datos de las tablas de excepciones y añádalos utilizando el programa de utilidad de carga. En ese caso, si desea volver a comprobar los datos, tenga en cuenta que la comprobación de restricciones no está confinada a las filas añadidas.
- Guarde el texto de definición del activador de la vista de catálogos del sistema relevante. Después elimine el activador INSERT y utilice INSERT para transferir las filas corregidas de las tablas de excepciones. Finalmente, vuelva a crear el activador utilizando la definición del activador guardada.

No se realiza una provisión explícita para evitar que se disparen los activadores cuando se insertan filas desde las tablas de excepciones.

Sólo se informa de una violación por fila para las violaciones de índices exclusivos.

Si en la tabla hay valores con LONG VARCHAR, LONG VARGRAPHIC o tipos de datos LOB, los valores no se insertan en la tabla de excepciones si se producen violaciones de índices exclusivos.

Consulta de las tablas de excepciones

La estructura de la columna de mensajes de una tabla de excepciones es una lista concatenada de nombres de restricciones, longitudes y delimitadores, tal como se describe antes. Esta información se puede consultar.

Por ejemplo, para recuperar una lista de todas las violaciones, repitiendo cada fila con sólo el nombre de la restricción junto a ella, supongamos que la tabla T1 original tiene dos columnas: C1 y C2. Supongamos también que la tabla de excepciones correspondiente, E1, tiene las columnas C1 y C2, que se corresponden con las de la T1, así como una columna de mensajes, MSGCOL. La siguiente consulta utiliza la recurrencia para enumerar un nombre de restricción por fila (que pertenece a las filas que tienen más de una violación):

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
     CHAR(SUBSTR(MSGCOL, 12,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
     1,
     15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV;
```

Para mostrar todas las filas que han violado una restricción concreta, la consulta anterior se puede ampliar de la siguiente manera:

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, 12,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
      1,
      15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'nombrerestricción';
```

La consulta siguiente puede utilizarse para obtener todas las violaciones de restricciones de comprobación:

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, 6, 1)),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, J, 1)),
       I+1,
       J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

Apéndice L. Sentencias de SQL que se permiten en rutinas

La tabla siguiente indica si se permite o no que la sentencia de SQL (especificada en la primera columna) se ejecute en una rutina que tenga especificada la indicación de acceso a datos SQL. Si se encuentra una sentencia de SQL ejecutable en una rutina definida con NO SQL, se devuelve SQLSTATE 38001. Para otros contextos de ejecución, las sentencias de SQL que no se soportan en ningún contexto devuelven SQLSTATE 38003. Para otras sentencias de SQL no permitidas en un contexto CONTAINS SQL, se devuelve SQLSTATE 38004. En un contexto READS SQL DATA, se devuelve SQLSTATE 38002. Durante la creación de una rutina SQL, una sentencia que no coincida con la indicación de acceso a datos SQL haría que se devolviera SQLSTATE 42985.

Si una sentencia invoca una rutina, la indicación de acceso a datos SQL efectiva para la sentencia será el mayor de:

- La indicación de acceso a datos SQL de la sentencia de la tabla siguiente.
- La indicación de acceso a datos SQL de la rutina especificada al crear la rutina.

Por ejemplo, la sentencia CALL tiene una indicación de acceso a datos SQL de CONTAINS SQL. Sin embargo, si se llama a un procedimiento almacenado definido como READS SQL DATA, la indicación de acceso a datos SQL efectiva para la sentencia CALL es READS SQL DATA.

Cuando una sentencia invoca una sentencia de SQL, la indicación de acceso a datos SQL efectiva para la sentencia no debe sobrepasar la indicación de acceso a datos SQL declarada para la rutina. Por ejemplo, una función definida como READS SQL DATA no podría llamar a un procedimiento almacenado definido como MODIFIES SQL DATA.

Tabla 263. Sentencia de SQL e indicación del acceso a datos SQL

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	S
AUDIT	N	N	N	S
BEGIN DECLARE SECTION	S(1)	S	S	S
CALL	N	S	S	S
CLOSE	N	N	S	S
COMMENT ON	N	N	N	S
COMMIT	N	N(4)	N(4)	N(4)
COMPOUND SQL	N	S	S	S
CONNECT(2)	N	N	N	N
CREATE...	N	N	N	S
DECLARE CURSOR	S(1)	S	S	S
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	S
DELETE	N	N	N	S
DESCRIBE	N	S	S	S

Tabla 263. Sentencia de SQL e indicación del acceso a datos SQL (continuación)

Sentencia de SQL	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DISCONNECT(2)	N	N	N	N
DROP ...	N	N	N	S
END DECLARE SECTION	S(1)	S	S	S
EXECUTE	N	S(3)	S(3)	S
EXECUTE IMMEDIATE	N	S(3)	S(3)	S
EXPLAIN	N	N	N	S
FETCH	N	N	S	S
FREE LOCATOR	N	S	S	S
FLUSH EVENT MONITOR	N	N	N	S
GRANT ...	N	N	N	S
INCLUDE	S(1)	S	S	S
INSERT	N	N	N	S
LOCK TABLE	N	S	S	S
MERGE	N	N	N	S
OPEN	N	N	S(5)	S
PREPARE	N	S	S	S
REFRESH TABLE	N	N	N	S
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	S
RENAME TABLE	N	N	N	S
REVOKE ...	N	N	N	S
ROLLBACK	N	N(4)	N(4)	N(4)
ROLLBACK TO SAVEPOINT	N	N	N	S
SAVEPOINT	N	N	N	S
SELECT INTO	N	N	S(5)	S
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	S
SET registro especial	N	S	S	S
SET variable	N	S(6)	S(5)	S
TRANSFER OWNERSHIP	N	N	N	S
TRUNCATE	N	N	N	S
UPDATE	N	N	N	S
VALUES INTO	N	N	S	S
WHENEVER	S(1)	S	S	S

Nota:

1. Aunque la opción NO SQL implica que no puede especificarse ninguna sentencia de SQL, las sentencias no ejecutables no están restringidas.

2. Las sentencias de gestión de conexiones no están permitidas en ningún contexto de ejecución de rutinas.
3. Depende de la sentencia que se ejecute. La sentencia especificada para la sentencia EXECUTE debe ser una sentencia que esté permitida en el contexto del nivel de acceso SQL que esté vigente. Por ejemplo, si el nivel acceso SQL en vigor es READS SQL DATA, la sentencia no deber ser INSERT, UPDATE ni DELETE.
4. Es posible utilizar la sentencia COMMIT y la sentencia ROLLBACK sin la cláusula TO SAVEPOINT en un procedimiento almacenado, pero sólo si se llama al procedimiento almacenado directamente desde una aplicación o indirectamente mediante llamadas a procedimientos almacenados anidados desde una aplicación. (Si alguna sentencia de activador, función, método o compuesto atómico está en la cadena de llamada al procedimiento almacenado, no se permite realizar COMMIT o ROLLBACK de una unidad de trabajo).
5. Si el nivel de acceso SQL en vigor es READS SQL DATA, no se puede incorporar ninguna sentencia de cambio de datos de SQL en la sentencia SELECT INTO en el cursor al que hace referencia la sentencia OPEN ni en la expresión de la parte derecha de la sentencia SET variable.
6. Si el nivel de acceso SQL en vigor es CONTAINS SQL, no se puede incorporar ninguna selección completa escalar en la expresión de la parte derecha de la sentencia SET variable.

Apéndice M. CALL invocada desde una sentencia compilada

Invoca un procedimiento almacenado en la ubicación de una base de datos. Un procedimiento, por ejemplo, se ejecuta en la ubicación de la base de datos y devuelve datos a la aplicación cliente.

Los programas que utilizan la sentencia de SQL CALL se diseñan para ejecutarse en dos partes, una en el cliente y la otra en el servidor. El procedimiento del servidor en la base de datos se ejecuta en la misma transacción que la aplicación cliente. Si la aplicación cliente y el procedimiento están en la misma partición de base de datos, el procedimiento almacenado se ejecuta localmente.

Nota: Esta forma de la sentencia CALL ha quedado obsoleta y sólo se suministra por motivos de compatibilidad con las versiones anteriores de DB2.

Invocación

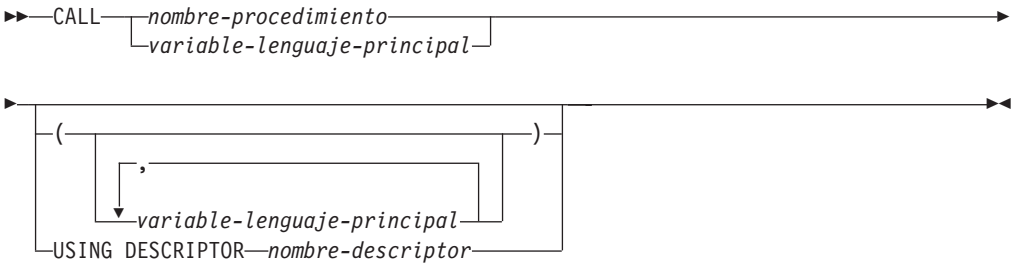
Esta forma de la sentencia CALL sólo puede incorporarse en un programa de aplicación precompilado con la opción CALL_RESOLUTION DEFERRED. No puede invocar un procedimiento federado. No puede utilizarse en activadores, procedimientos de SQL ni ningún otro contexto que no sea de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente. No obstante, el nombre de procedimiento puede especificarse mediante una variable del lenguaje principal y esto, junto con el uso de la cláusula USING DESCRIPTOR, permite proporcionar tanto el nombre de procedimiento como la lista de parámetros en tiempo de ejecución, con lo que se consigue un efecto similar al de una sentencia preparada dinámicamente.

Autorización

Entre los privilegios que el ID de autorización de la sentencia necesita poseer *durante la ejecución* debe incluirse uno de los siguientes:

- Privilegio EXECUTE para el paquete asociado al procedimiento; no se comprueba el privilegio EXECUTE en el procedimiento
- Privilegio CONTROL para el paquete asociado al procedimiento
- Autorización SYSADM o DBADM

Sintaxis



Descripción

nombre-procedimiento **o** *variable-lenguaje-principal*
Identifica el procedimiento que se va a llamar. El nombre de procedimiento puede especificarse directamente o dentro de una variable del lenguaje principal. El procedimiento identificado debe existir en el servidor actual (SQLSTATE 42724).

Si se especifica el *nombre-procedimiento*, debe ser un identificador normal que no sobrepase los 254 bytes. Como sólo puede ser un identificador normal, no puede contener blancos ni caracteres especiales. El valor se convierte a mayúsculas. Si es necesario utilizar nombres en minúsculas, blancos o caracteres especiales, el nombre debe especificarse mediante una *variable-lenguaje-principal*.

Si se especifica *variable-lenguaje-principal*, debe ser una variable CHAR o VARCHAR con un atributo de longitud que no sobrepase los 254 bytes y no debe incluir una variable de indicador. El valor *no* se convierte a mayúsculas. La serie de caracteres debe estar justificada por la izquierda.

El nombre de procedimiento puede tener uno de estos formatos.

nombre-procedimiento
El nombre (sin extensión) del procedimiento que se va a ejecutar. El procedimiento que se invoca se determina de la manera siguiente.

1. Se utiliza el *nombre-procedimiento* para buscar un procedimiento que coincida en los procedimientos definidos (en SYSCAT.ROUTINES). Un procedimiento que coincida se determina utilizando los pasos siguientes.
   a. Busque los procedimientos (ROUTINETYPE es 'P') del catálogo (SYSCAT.ROUTINES), donde ROUTINENAME coincida con el *nombre-procedimiento* especificado y ROUTINESCHEMA sea un nombre de esquema en la vía de acceso de SQL (registro especial CURRENT PATH). Si el nombre de esquema está especificado explícitamente, la vía de acceso de SQL se ignora y sólo se tienen en cuenta los procedimientos con el nombre de esquema especificado.
   b. Después, elimine cualquiera de estos procedimientos que no tengan el mismo número de parámetros que el número de argumentos especificados en la sentencia CALL.
   c. Elija el procedimiento restante que esté antes en la vía de acceso de SQL.

   Si se selecciona un procedimiento, DB2 invocará el procedimiento definido por el nombre externo.
2. Si no se encuentra ningún procedimiento que coincida, se utiliza el *nombre-procedimiento* como el nombre de la biblioteca de procedimientos y el nombre de función dentro de dicha biblioteca. Por ejemplo, si el *nombre-procedimiento* es `proclib`, el servidor DB2 cargará la biblioteca de procedimientos `proclib` y ejecutará la rutina de función `proclib()` dentro de esa biblioteca.

   En sistemas UNIX, el directorio por omisión para las bibliotecas de procedimiento es `sqllib/function`. El directorio por omisión para los procedimientos no delimitados es `sqllib/function/unfenced` .

   En los sistemas basados en Windows, el directorio por omisión para las bibliotecas de procedimientos es `sqllib\function`. El directorio por omisión para los procedimientos no delimitados es `sqllib\function\unfenced`.

Si no se ha encontrado la biblioteca o función, se devuelve un error (SQLSTATE 42884).

biblioteca-procedimiento!nombre-función
El carácter de admiración (!) actúa como delimitador entre el nombre de biblioteca y el nombre de función del procedimiento. Por ejemplo, si se especifica `proclib!func`, se carga `proclib` en memoria y se ejecuta la función `func` desde esa biblioteca. Esto permite que se coloquen múltiples funciones en la misma biblioteca de procedimientos.

La biblioteca de procedimientos está ubicada en los directorios o especificada en la variable LIBPATH, tal como se describe en el *procedimiento-nombre*.

vía-acceso-absoluta!nombre-función
La *vía-acceso-absoluta* especifica la vía de acceso completa a la biblioteca de procedimientos almacenados.

En un sistema UNIX, por ejemplo, si se especifica `/u/terry/proclib!func`, se obtiene la biblioteca de procedimientos `proclib` del directorio `/u/terry` y se ejecuta la función `func` de dicha biblioteca.

En todos estos casos, la longitud total del nombre de procedimiento, incluida su vía de acceso completa implícita o explícita, no debe tener una longitud superior a 254 bytes.

(variable-lenguaje-principal,...)
Cada especificación de *variable-lenguaje-principal* es un parámetro de la sentencia CALL. El parámetro*n* de CALL corresponde al parámetro *n* del procedimiento del servidor.

Se supone que se utiliza cada *variable-lenguaje-principal* para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe proporcionar una variable de indicador con cada parámetro y establecer el indicador en -1 si el parámetro no se utiliza para transmitir datos al procedimiento. El procedimiento debe establecer la variable de indicador en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Si el servidor de bases de datos es DB2 9.1, los parámetros deben tener tipos de datos coincidentes en el programa cliente y servidor.

USING DESCRIPTOR *nombre-descriptor*
Identifica una SQLDA que debe contener una descripción válida de las variables del lenguaje principal. El elemento SQLVAR *n* corresponde al parámetro *n* del procedimiento del servidor.

Antes de que se procese la sentencia CALL, la aplicación debe definir los campos siguientes de la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables. Deben inicializarse los siguientes campos de cada elemento pasado de SQLVAR base:
  - SQLTYPE

- SQLLEN
- SQLDATA
- SQLIND

Deben inicializarse los siguientes campos de cada elemento pasado de la SQLVAR secundaria:

- LEN.SQLLONGLEN
- SQLDATALEN
- SQLDATATYPE_NAME

Se supone que la SQLDA se utiliza para intercambiar datos en ambas direcciones entre cliente y servidor. Para evitar enviar datos innecesarios entre cliente y servidor, la aplicación cliente debe establecer el campo SQLIND en -1 si el parámetro no se utiliza para transmitir datos al procedimiento. El procedimiento debe establecer el campo SQLIND en -128 para cualquier parámetro que no se utilice para devolver datos a la aplicación cliente.

Notas

- ***Utilización de tipos de datos de objeto grande (LOB)***:

  Si la aplicación cliente y servidora tiene que especificar datos LOB de una SQLDA, asigne el doble al número de entradas SQLVAR.

  Desde DB2 Versión 2, los procedimientos soportan los tipos de datos LOB. Estos tipos de datos no están soportados por todos los clientes o servidores de versiones anteriores.

- ***Recuperación de DB2_RETURN_STATUS desde un procedimiento SQL***:

  Si un procedimiento SQL emite satisfactoriamente una sentencia RETURN junto con un valor de estado, este valor se coloca en el primer campo SQLERRD de la SQLCA. Si la sentencia CALL se emite en un procedimiento SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de DB2_RETURN_STATUS. El valor es -1 cuando SQLSTATE indica un error.

- ***Devolución de conjuntos de resultados de los procedimientos***:

  Si el programa de aplicación cliente se escribe utilizando CLI, los conjuntos de resultados pueden devolverse directamente a la aplicación cliente. 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:

  - Por cada cursor que se ha dejado abierto, se devuelve un conjunto resultante a la aplicación.
  - Si se deja abierto más de un cursor, los conjuntos del resultado se devuelven en el orden en que se han abierto sus cursores.
  - Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto de resultados de un cursor tiene 500 filas, y el procedimiento ha leído 150 de dichas filas una vez que finaliza el procedimiento, se devolverán a la aplicación desde la fila 151 a la fila 500.

- ***Manejo de registros especiales***:

  Los valores de los registros especiales utilizados para el llamador los hereda el procedimiento durante la invocación y se restauran al devolver el control al llamador. Los registros especiales se pueden modificar dentro de un procedimiento, pero estos cambios no tienen efecto en el llamador. Esto no es cierto para los procedimientos de versiones anteriores (aquéllos definidos con el

estilo de parámetro DB2DARI o situados en la biblioteca por omisión), en los que los cambios efectuados en los registros especiales de un procedimiento se convierten en los valores del llamador.

- *Compatibilidades*

  Existe una forma nueva, preferida de la sentencia CALL que puede incluirse en una aplicación (precompilando la aplicación con la opción CALL_RESOLUTION IMMEDIATE) o que puede prepararse de forma dinámica.

Ejemplos

Ejemplo 1:

En C, invoque un procedimiento denominado TEAMWINS en la biblioteca ACHIEVE, pasándole un parámetro almacenado en la variable del lenguaje principal HV_ARGUMENT.

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

Ejemplo 2:

En C, invoque un procedimiento denominado :SALARY_PROC, utilizando la SQLDA denominada INOUT_SQLDA.

```
struct sqlda *INOUT_SQLDA;
/* El código de configuración para variables SQLDA va aquí */
CALL :SALARY_PROC
USING DESCRIPTOR :*INOUT_SQLDA;
```

Ejemplo 3:

Un procedimiento 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.prepareCall( sql ) ; /* con es la conexión */

stpCall.setInt( 1, variable1 ) ;
stpCall.setBigDecimal( 2, variable2 ) ;
stpCall.setInt( 3, variable3 ) ;

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

variable2 = stpCall.getBigDecimal(2) ;
variable3 = stpCall.getInt(3) ;
...
```

CALL invocada desde una sentencia compilada

Este fragmento de código de aplicación invocará el método Java *onhand* en 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'.

Apéndice N. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- Centro de información de DB2
  - Temas (Tareas, concepto y temas de consulta)
  - Ayuda para herramientas de DB2
  - 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 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 cuando estén disponibles, o consulte el Centro de información de DB2 en 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. 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. 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/shop/publications/order. Los manuales de DB2 Versión 9.7 en inglés y las versiones traducidas en formato PDF se pueden descargar del sitio web www.ibm.com/support/docview.wss?rs=71 &uid=swg2700947.

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 264. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
Consulta de las API administrativas	SC11-3912-01	Sí	Noviembre de 2009
Rutinas y vistas administrativas	SC11-3909-01	No	Noviembre de 2009
Call Level Interface Guide and Reference, Volume 1	SC27-2437-01	Sí	Noviembre de 2009
Call Level Interface Guide and Reference, Volume 2	SC27-2438-01	Sí	Noviembre de 2009
Consulta de mandatos	SC11-3914-01	Sí	Noviembre de 2009
Data Movement Utilities Guide and Reference	SC27-2440-00	Sí	Agosto de 2009
Data Recovery and High Availability Guide and Reference	SC27-2441-01	Sí	Noviembre de 2009
Database Administration Concepts and Configuration Reference	SC27-2442-01	Sí	Noviembre de 2009
Database Monitoring Guide and Reference	SC27-2458-01	Sí	Noviembre de 2009
Database Security Guide	SC27-2443-01	Sí	Noviembre de 2009
Guía de DB2 Text Search	SC11-3927-01	Sí	Noviembre de 2009
Developing ADO.NET and OLE DB Applications	SC27-2444-01	Sí	Noviembre de 2009
Developing Embedded SQL Applications	SC27-2445-01	Sí	Noviembre de 2009
Desarrollo de aplicaciones Java	SC11-3907-01	Sí	Noviembre de 2009

Tabla 264. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails	SC11-3908-00	No	Agosto de 2009
Developing User-defined Routines (SQL and External)	SC27-2448-01	Sí	Noviembre de 2009
Getting Started with Database Application Development	GI11-9410-01	Sí	Noviembre de 2009
Iniciación a la instalación y administración de DB2 en Linux y Windows	GI11-8640-00	Sí	Agosto de 2009
Globalization Guide	SC27-2449-00	Sí	Agosto de 2009
Instalación de servidores DB2	SC11-3916-01	Sí	Noviembre de 2009
Instalación de clientes de servidor de datos de IBM	SC11-3917-00	No	Agosto de 2009
Consulta de mensajes Volumen 1	SC11-3922-00	No	Agosto de 2009
Consulta de mensajes Volumen 2	SC11-3923-00	No	Agosto de 2009
Net Search Extender Guía de administración y del usuario	SC11-3926-01	No	Noviembre de 2009
Partitioning and Clustering Guide	SC27-2453-01	Sí	Noviembre de 2009
pureXML Guide	SC27-2465-01	Sí	Noviembre de 2009
Query Patroller Administration and User's Guide	SC27-2467-00	No	Agosto de 2009
Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta	SC11-3925-00	No	Agosto de 2009
SQL Procedural Languages: Application Enablement and Support	SC27-2470-01	Sí	Noviembre de 2009
Consulta de SQL, Volumen 1	SC11-3910-01	Sí	Noviembre de 2009
Consulta de SQL, Volumen 2	SC11-3911-01	Sí	Noviembre de 2009
Troubleshooting and Tuning Database Performance	SC27-2461-01	Sí	Noviembre de 2009
Actualización a DB2 Versión 9.7	SC11-3915-01	Sí	Noviembre de 2009

Tabla 264. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Guía de aprendizaje de Visual Explain	SC11-3924-00	No	Agosto de 2009
Novedades en DB2 Versión 9.7	SC11-3921-01	Sí	Noviembre de 2009
Workload Manager Guide and Reference	SC27-2464-01	Sí	Noviembre de 2009
XQuery Reference	SC27-2466-01	No	Noviembre de 2009

Tabla 265. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
Instalación y configuración de DB2 Connect Personal Edition	SC11-3919-01	Sí	Noviembre de 2009
Instalación y configuración de servidores DB2 Connect	SC11-3920-01	Sí	Noviembre de 2009
Guía del usuario de DB2 Connect	SC11-3918-01	Sí	Noviembre de 2009

Tabla 266. Información técnica de Information Integration

Nombre	Número de documento	Copia impresa disponible	Última actualización
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	Sí	Agosto de 2009
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC11-3900-04	Sí	Agosto de 2009
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	No	Agosto de 2009
Information Integration: SQL Replication Guide and Reference	SC11-3899-02	Sí	Agosto de 2009
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	Sí	Agosto de 2009

Pedido de manuales de DB2 en copia impresa

Si necesita manuales de DB2 en copia impresa, puede comprarlos en línea en varios países o regiones, pero no en todos. Siempre puede hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM. Recuerde que algunas publicaciones en copia software del DVD *Documentación en*

PDF de DB2 no están disponibles en copia impresa. Por ejemplo, no está disponible la publicación *Consulta de mensajes de DB2* en copia impresa.

Las versiones impresas de muchas de las publicaciones de DB2 disponibles en el DVD de Documentación en PDF de DB2 se pueden solicitar a IBM por una cantidad. Dependiendo desde dónde realice el pedido, podrá solicitar manuales en línea, desde el Centro de publicaciones de IBM. Si la realización de pedidos en línea no está disponible en su país o región, siempre puede hacer pedidos de manuales de DB2 en copia impresa al representante local de IBM. Tenga en cuenta que no todas las publicaciones del DVD de Documentación en PDF de DB2 están disponibles en copia impresa.

Nota: La documentación más actualizada y completa de DB2 se conserva en el Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2luw/v9r7.

Para hacer pedidos de manuales de DB2 en copia impresa:

- Para averiguar si puede hacer pedidos de manuales de DB2 en copia impresa en línea en su país o región, consulte el Centro de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Debe seleccionar un país, región o idioma para poder acceder a la información sobre pedidos de publicaciones y, a continuación, seguir las instrucciones sobre pedidos para su localidad.
- Para hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM:
  1. Localice la información de contacto de su representante local desde uno de los siguientes sitios Web:
     - El directorio de IBM de contactos en todo el mundo en el sitio www.ibm.com/planetwide
     - El sitio Web de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Tendrá que seleccionar su país, región o idioma para acceder a la página de presentación de las publicaciones apropiadas para su localidad. Desde esta página, siga el enlace "Acerca de este sitio".
  2. Cuando llame, indique que desea hacer un pedido de una publicación de DB2.
  3. Proporcione al representante los títulos y números de documento de las publicaciones que desee solicitar. Si desea consultar los títulos y los números de documento, consulte el apartado “Biblioteca técnica de DB2 en copia impresa o en formato PDF” en la página 1164.

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.

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

Para los temas de la versión 9.7 de DB2, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Para los temas de la versión 9.5 de DB2, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para los temas de la versión 9.1 de DB2, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Para los temas de la versión 8 de DB2 vaya al URL del *Centro de información de DB2* en el sitio: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Visualización de temas en su idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias del navegador. Si un tema no se ha traducido al idioma preferido, el Centro de información de DB2 visualiza dicho tema en inglés.

- Para visualizar temas en su idioma preferido en el navegador Internet Explorer:
  1. En Internet Explorer, pulse en el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...**. Se abrirá la ventana Preferencias de idioma.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**.

       Nota: La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Renueve la página para que aparezca el Centro de información de DB2 en su idioma preferido.
- Para visualizar temas en su idioma preferido en un navegador Firefox o Mozilla:
  1. Seleccione el botón en la sección **Idiomas** del diálogo **Herramientas** —> **Opciones** —> **Avanzado**. Se visualizará el panel Idiomas en la ventana Preferencias.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Añadir...** a fin de seleccionar un idioma en la ventana Añadir idiomas.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Renueve la página para que aparezca el Centro de información de DB2 en su idioma preferido.

En algunas combinaciones de navegador y sistema operativo, también debe cambiar los valores regionales del sistema operativo al entorno local y al idioma de su elección.

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 9.7 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:

- Actualizaciones automáticas: actualiza las funciones y los idiomas del Centro de información existentes. Una ventaja adicional de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo mínimo mientras se realiza la actualización. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Actualizaciones manuales: se deben utilizar si se quieren añadir funciones o idiomas durante el proceso de actualización. 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.

Procedimiento

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".

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/V9.7`.
   b. Navegue desde el directorio de instalación al directorio `doc/bin`.
   c. Ejecute el script `ic-update`:

```
ic-update
```

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\Centro de información de DB2\Versión 9.7, 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 ic-update.bat:

      ```
      ic-update.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, puede obtener las actualizaciones de la documentación de IBM e instalarlas.

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 para estaciones 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/db2icdv97 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\Version 9.7, 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/V9.7.
     b. Navegue desde el directorio de instalación al directorio doc/bin.
     c. Ejecute el script help_start:
        ```
        help_start
        ```

   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 de instalación `doc/bin` 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/db2icdv97 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 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/db2help/.

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.

"Visual Explain" en la *Guía de aprendizaje de Visual Explain*
: Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

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 *DB2 Troubleshooting Guide* o en la sección Conceptos fundamentales sobre bases de datos del Centro de información de *DB2*. En ellas encontrará información sobre cómo aislar e identificar problemas utilizando herramientas y programas de utilidad de diagnóstico de DB2, soluciones a algunos de los problemas más habituales y otros consejos sobre cómo solucionar problemas que podría encontrar en los productos DB2.

Sitio web de soporte técnico de DB2
Consulte el sitio Web de soporte técnico de DB2 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 sitio Web de soporte técnico de DB2 en la dirección http://www.ibm.com/software/data/db2/support/db2_9/

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes 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 derivativos de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

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.

Apéndice O. 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.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 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

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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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 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 sección Copyright and trademark information de la web 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 todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.
- 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, Intel® Centrino®, el logotipo de Intel Centrino, Celeron®, Intel® Xeon®, 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

Caracteres Especiales

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

SC11-3910-01

Spine information:

IBM DB2 9.7 para Linux, UNIX y Windows **Versión 9 Release 7** **Consulta de SQL, Volumen 1**