IBM® DB2 Universal Database™

Consulta de SQL Volumen 2

Versión 8.2

SC10-3731-01

IBM® DB2 Universal Database™

Consulta de SQL Volumen 2

Versión 8.2

SC10-3731-01

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el apartado *Avisos*.

Esta publicación es la traducción del original inglés *IBM DB2 Universal Database, SQL Reference Volume 2, Version 8.2, (SC09-4845-01).*

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

- Para realizar pedidos de publicaciones en línea, vaya a IBM Publications Center en www.ibm.com/shop/publications/order
- Para encontrar el representante de IBM correspondiente a su localidad, vaya a IBM Directory of Worldwide Contacts en www.ibm.com/planetwide

Para realizar pedidos de publicaciones en márketing y ventas de DB2 de los EE.UU. o de Canadá, llame al número 1-800-IBM-4YOU (426-4968).

Contenido

Acerca de este manual

El manual Consulta de SQL en sus dos volúmenes define el lenguaje de SQL utilizado por DB2 Universal Database Versión 8, e 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 esta estructurado este manual

Este manual contiene información acerca de los siguientes temas principales:

- El apartado "Sentencias", en la página 1 contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos de todas las sentencias de SQL, entre ellas las sentencias de procedimiento de SQL.

Breve visión general del Volumen 1

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 en dicho volumen se describen brevemente aquí:

- "Conceptos" describe 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 las funciones de columna y funciones escalares del SQL.
- "Consultas" describe los distintos formatos de una consulta.
- "Límites de SQL" lista las limitaciones del SQL.
- "Área de comunicaciones del SQL (SQLCA)" describe la estructura de SQLCA.
- "Área de descriptor de SQL (SQLDA)" describe la estructura de SQLDA.
- "Vistas de catálogos" describe las vistas de catálogos de bases de datos.
- "Sistemas federados" describe las opciones y las correlaciones de tipos para sistemas federados.

- "Tablas de base de datos de ejemplo" describe las tablas de ejemplo utilizadas en los ejemplos.
- "Nombres de esquemas reservados y palabras reservadas" contiene los nombres de esquemas reservados y las palabras reservadas correspondientes a los estándares SQL de IBM y SQL00 de ISO/ANSI.
- "Interacción de los activadores y las restricciones" 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 de 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" describe la sentencia CALL que se puede invocar desde una sentencia compilada.
- "Consideraciones acerca del EUC del japonés y del chino tradicional" lista las consideraciones que se deben tener en cuenta al utilizar los juegos de caracteres de código UNIX ampliado (EUC).
- "Especificaciones BNF para DATALINK" contiene las especificaciones del formato BNF (Backus-Naur) para DATALINK.

Cómo se leen los diagramas de sintaxis

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

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

El símbolo ►►— indica el inicio de un diagrama de sintaxis.

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

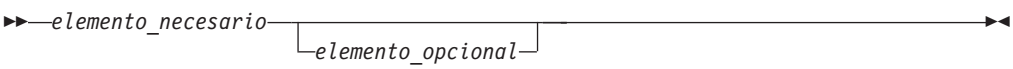
El símbolo ►— indica que la sintaxis es la continuación de la línea anterior.

El símbolo —►◄ indica el final de un diagrama de sintaxis.

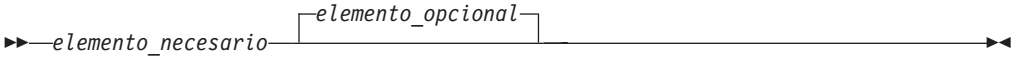
Los fragmentos de sintaxis empiezan con el símbolo ├— y finalizan con el símbolo —┤.

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

Los elementos opcionales aparecen debajo de la línea principal.

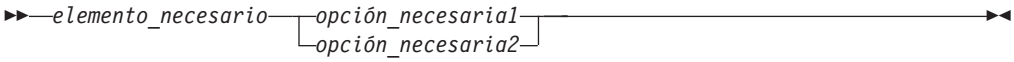


Si aparece un elemento opcional por encima de la línea principal, dicho elemento no tiene ningún efecto en la ejecución y sólo se utiliza para facilitar su lectura.

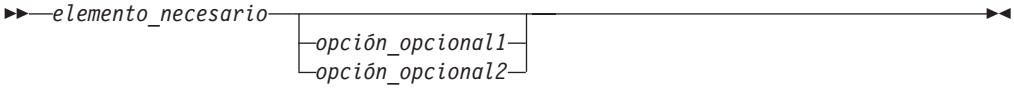


Si puede elegir entre dos o más elementos, aparecen en una pila.

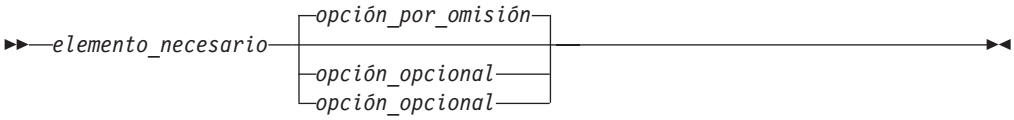
Si *debe* elegir uno de los elementos, un elemento de la pila aparece en la ruta principal.



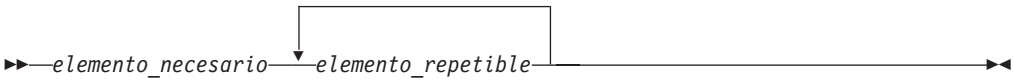
Si la selección de uno de los elementos es opcional, toda la pila aparece por debajo de la línea principal.



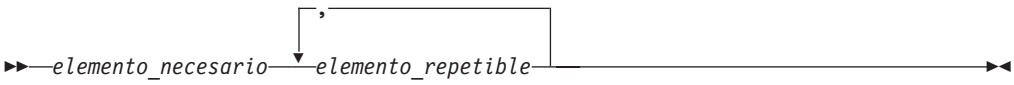
Si uno de los elementos es el valor por omisión, éste aparecerá por encima de la línea principal y el resto de las opciones se mostrarán por debajo de la línea principal.



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



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

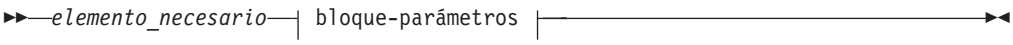


Una flecha de repetición por encima de una pila indica que puede elegir más de una opción de entre los elementos apilados o repetir una sola opción.

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

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

A veces, una única variable representa un fragmento grande de la sintaxis. Por ejemplo, en el diagrama siguiente, la variable bloque-parámetros representa todo el fragmento de sintaxis que se ha etiquetado como **bloque-parámetros**:



bloque-parámetros:

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

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

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

Elementos comunes de la sintaxis

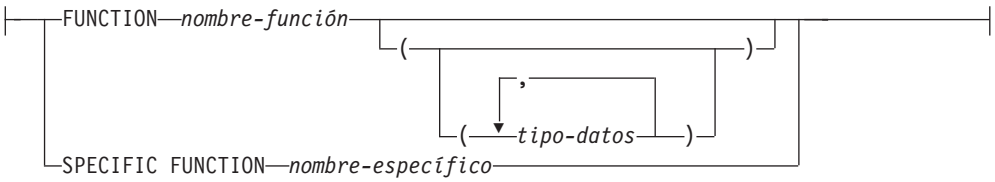
En los apartados siguientes se describen diversos fragmentos de la sintaxis que se utilizan en los diagramas de la sintaxis. A los fragmentos se hace referencia de la forma siguiente:

Designador de función

Un designador de función identifica una única función de forma exclusiva. Por lo general, los designadores de función aparecen en las sentencias de DDL de las funciones (como, por ejemplo, DROP o ALTER).

Sintaxis:

designador-función:



Descripción:

FUNCTION *nombre-función*
Identifica una función en particular y sólo es válido si existe exactamente una instancia de función con el nombre *nombre-función* en el esquema. Para la función identificada puede definirse cualquier número de parámetros. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si existe más de una instancia de la función en el esquema especificado o implícito, se genera un error (SQLSTATE 42725).

FUNCTION *nombre-función (tipo-datos,...)*
Proporciona la signatura de la función, que identifica la función de forma exclusiva. No se utiliza el algoritmo de resolución de función.

nombre-función
Especifica el nombre de la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

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

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

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

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

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

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha definido para *n*, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

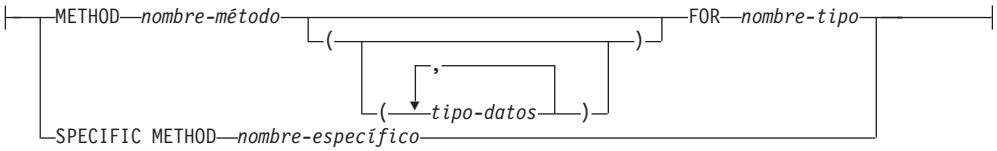
SPECIFIC FUNCTION *nombre-específico*
Identifica una función definida por el usuario en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación de la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

Designador de método

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

Sintaxis:

designador-método:



Descripción:

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

METHOD *nombre-método (tipo-datos,...)*
Proporciona la signatura del método, que identifica de forma exclusiva al método. No se utiliza el algoritmo de resolución de método.

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

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

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En lugar de ello, puede codificarse un

conjunto de paréntesis vacío para indicar que esos atributos deben pasarse por alto durante la búsqueda de una coincidencia de tipo de datos.

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

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

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha definido para *n*, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún método con la signatura especificada para el tipo en el esquema indicado o implícito, se generará un error (SQLSTATE 42883).

FOR *nombre-tipo*
Especifica el nombre del tipo al que va a asociarse el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

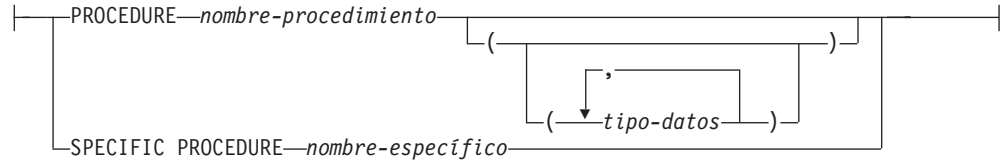
SPECIFIC METHOD *nombre-específico*
Identifica un método en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del método. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de método específica en el esquema indicado o implícito; de lo contrario, se generará un error (SQLSTATE 42704).

Designador de procedimiento

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

Sintaxis:

designador-procedimiento:



Descripción:

PROCEDURE *nombre-procedimiento*
Identifica un procedimiento en particular y sólo es válido si existe exactamente una instancia de procedimiento con el nombre *nombre-procedimiento* en el esquema. Para el procedimiento identificado puede definirse cualquier número de parámetros. En las sentencias de SQL dinámico, el registro especial

CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. Si no existe ningún procedimiento con este nombre en el esquema nombrado o implícito, se genera un error (SQLSTATE 42704). Si existe más de una instancia del procedimiento en el esquema especificado o implícito, se genera un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento (tipo-datos,...)*
Proporciona la signatura del procedimiento, que identifica el procedimiento de forma exclusiva. No se utiliza el algoritmo de resolución de procedimiento.

nombre-procedimiento
Especifica el nombre del procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

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

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

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

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

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

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha definido para *n*, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún procedimiento con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico*
Identifica un procedimiento en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

Convenios utilizados en este manual

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

Condiciones de error

Una condición de error se indica en el texto del manual listando entre paréntesis el SQLSTATE asociado al error. Por ejemplo:

```
Si hay una signatura duplicada se genera 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 están predefinidos por el sistema.
Cursiva	Indica uno de los siguientes: • Nombres o valores (variables) que debe suministrar el usuario. • Énfasis general. • La presentación de un término nuevo. • Una referencia a otra fuente de información.
`Monoespaciado`	Indica uno de los siguientes: • Archivos y directorios. • Información que se indica al usuario que escriba en un indicador de mandatos o en una ventana. • Ejemplos de valores de datos específicos. • Ejemplos de texto similar a lo que puede mostrar el sistema. • Ejemplos de mensajes del sistema.

Documentación relacionada

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

- *Administration Guide*
  - Contiene la información necesaria para diseñar, implantar y mantener una base de datos a la que se va a acceder de forma local o en un entorno de cliente/servidor.
- *Application Development Guide*
  - Explica el proceso de desarrollo de aplicaciones y la forma de codificar, compilar y ejecutar programas de aplicación que utilizan SQL intercalado y API para acceder a la base de datos.
- *DB2 Universal Database for iSeries SQL Reference*
  - Este manual define el Lenguaje de consulta estructurada (SQL) soportado por DB2 Query Manager y SQL Development Kit en iSeries (AS/400). Contiene información de consulta para las tareas de administración del sistema, administración de la base de datos, programación de aplicaciones y operación. Este manual incluye sintaxis, notas acerca del uso, palabras claves y ejemplos para cada una de las sentencias de SQL utilizadas en sistemas iSeries (AS/400) que ejecutan DB2.
- *DB2 Universal Database for z/OS and OS/390 SQL Reference*

  - Este manual define el Lenguaje de consulta estructurada (SQL) utilizado en DB2 para z/OS (OS/390). Este manual proporciona formatos de consulta, sentencias de SQL, sentencias de procedimientos de SQL, límites de DB2, SQLCA, SQLDA, tablas de catálogos y palabras reservadas para sistemas z/OS (OS/390) 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.
- *Consulta de SQL de IBM*
  - 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 base de datos relacionales 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 1992 de ISO.
- *ISO/IEC 9075-2:1999, Database Language SQL -- Part 2: Foundation (SQL/Foundation)*
  - Contiene una gran parte de la definición de SQL proporcionada por la norma 1999 de ISO.
- *ISO/IEC 9075-4:1999, Database Language SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
  - Contiene la definición de las sentencias de control de los procedimientos SQL, tal como aparece en la norma 1999 de ISO.
- *ISO/IEC 9075-5:1999, Database Language SQL -- Part 4: Host Language Bindings (SQL/Bindings)*
  - Contiene la definición de los enlaces de lenguaje principal y de SQL dinámico, tal como aparece en la norma 1999 de ISO.

Sentencias

Este capítulo contiene diagramas de sintaxis, descripciones semánticas, normas y ejemplos del uso de sentencias de SQL, que incluyen sentencias que constituyen el cuerpo de una rutina de SQL, un activador o una sentencia compuesta dinámica.

Sentencias de SQL soportadas

La tabla siguiente indica las sentencias de SQL soportadas.

Tabla 1. Sentencias de SQL

Sentencia de SQL	Propósito
"ALLOCATE CURSOR" en la página 13	Asigna un cursor para el conjunto de resultados identificados por la variable del localizador de conjunto de resultados.
"ALTER BUFFERPOOL" en la página 15	Cambia la definición de una agrupación de almacenamientos intermedios.
"ALTER DATABASE PARTITION GROUP" en la página 18	Cambia la definición de un grupo de particiones de base de datos.
"ALTER FUNCTION" en la página 22	Modifica una función existente cambiando las propiedades de la función.
"ALTER METHOD" en la página 25	Modifica un método existente cambiando el cuerpo de método que se asocia con el método.
"ALTER NICKNAME" en la página 27	Cambia la definición de un apodo.
"ALTER PROCEDURE" en la página 35	Modifica un procedimiento existente cambiando las propiedades del procedimiento.
"ALTER SEQUENCE" en la página 39	Cambia la definición de una secuencia.
"ALTER SERVER" en la página 43	Cambia la definición de una fuente de datos en un sistema federado.
"ALTER TABLE" en la página 46	Cambia la definición de una tabla.
"ALTER TABLESPACE" en la página 81	Cambia la definición de un espacio de tablas.
"ALTER TYPE (Estructurado)" en la página 89	Cambia la definición de un tipo estructurado.
"ALTER USER MAPPING" en la página 97	Cambia la definición de una correlación de autorizaciones de usuario.
"ALTER VIEW" en la página 99	Cambia la definición de una vista modificando una columna de tipo de referencia para añadir un ámbito.
"ALTER WRAPPER" en la página 101	Actualiza las opciones que, junto con un módulo de reiniciador, se utilizan para acceder a las fuentes de datos de un tipo específico.
"ASSOCIATE LOCATORS" en la página 103	Obtiene el valor del localizador para cada conjunto de resultados devuelto por un procedimiento almacenado.
"BEGIN DECLARE SECTION" en la página 105	Marca el principio de una sección de declaración de variables del lenguaje principal.
"CALL" en la página 107	Invoca un procedimiento almacenado.
"CASE" en la página 113	Selecciona una vía de acceso de ejecución basándose en múltiples condiciones.
"CLOSE" en la página 116	Cierra un cursor.
"COMMENT" en la página 118	Sustituye o añade un comentario a la descripción de un objeto.
"COMMIT" en la página 128	Finaliza una unidad de trabajo y confirma los cambios que esa unidad de trabajo ha realizado en la base de datos.

Sentencias de SQL soportadas

Tabla 1. Sentencias de SQL (continuación)

Sentencia de SQL	Propósito
"SQL compuesto (Dinámico)" en la página 130	Combina una o más sentencias de SQL diferentes en un bloque dinámico.
"SQL compuesto (Incorporado)" en la página 135	Combina una o varias sentencias de SQL para formar un bloque ejecutable.
"SQL compuesto (procedimiento)" en la página 139	Agrupa otras sentencias en un procedimiento de SQL.
"CONNECT (Tipo 1)" en la página 148	Conecta a un servidor de aplicaciones según las normas para una unidad de trabajo remota.
"CONNECT (Tipo 2)" en la página 155	Conecta a un servidor de aplicaciones según las normas para la unidad de trabajo distribuida dirigida por aplicación.
"CREATE ALIAS" en la página 163	Define un alias para una tabla, vista u otro alias.
"CREATE BUFFERPOOL" en la página 166	Crea una nueva agrupación de almacenamientos intermedios.
"CREATE DATABASE PARTITION GROUP" en la página 170	Define un grupo de particiones de base de datos.
"CREATE DISTINCT TYPE" en la página 173	Define un tipo de datos diferenciado.
"CREATE EVENT MONITOR" en la página 179	Especifica sucesos de la base de datos que se deben supervisar.
"CREATE FUNCTION" en la página 197	Registra una función definida por el usuario.
"CREATE FUNCTION (Escalar externa)" en la página 198	Registra una función escalar externa definida por el usuario.
"CREATE FUNCTION (Tabla externa)" en la página 223	Registra una función de tabla externa definida por el usuario.
"CREATE FUNCTION (Tabla externa OLE DB)" en la página 241	Registra una función de tabla externa OLE DB definida por el usuario.
"CREATE FUNCTION (Con origen o plantilla)" en la página 249	Registra una función con origen definida por el usuario.
"CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 259	Registra y define una función SQL definida por el usuario.
"CREATE FUNCTION MAPPING" en la página 268	Define una correlación de funciones.
"CREATE INDEX" en la página 273	Define un índice para una tabla.
"CREATE INDEX EXTENSION" en la página 282	Define un objeto de extensión para su uso con índices sobre tablas con columnas de tipo estructurado o diferenciado.
"CREATE METHOD" en la página 289	Asocia un cuerpo de método con una especificación de método definida previamente.
"CREATE NICKNAME" en la página 295	Define un apodo.
"CREATE PROCEDURE" en la página 307	Registra un procedimiento almacenado.
"CREATE PROCEDURE (Externo)" en la página 308	Registra un procedimiento almacenado externo.
"CREATE PROCEDURE (SQL)" en la página 322	Registra un procedimiento almacenado SQL.
"CREATE SCHEMA" en la página 329	Define un esquema.
"CREATE SEQUENCE" en la página 332	Define una secuencia.
"CREATE SERVER" en la página 337	Define una fuente de datos para una base de datos federada.
"CREATE TABLE" en la página 341	Define una tabla.

Tabla 1. Sentencias de SQL (continuación)

Sentencia de SQL	Propósito
"CREATE TABLESPACE" en la página 405	Define un espacio de tablas.
"CREATE TRANSFORM" en la página 414	Define funciones de transformación.
"CREATE TRIGGER" en la página 422	Define un activador.
"CREATE TYPE (Estructurado)" en la página 433	Define un tipo de datos estructurado.
"CREATE TYPE MAPPING" en la página 458	Define una correlación entre tipos de datos.
"CREATE USER MAPPING" en la página 464	Define una correlación entre autorizaciones de usuario.
"CREATE VIEW" en la página 466	Define una vista de una o más tablas, vistas o apodos.
"CREATE WRAPPER" en la página 481	Registra un reiniciador.
"DECLARE CURSOR" en la página 483	Define un cursor SQL.
"DECLARE GLOBAL TEMPORARY TABLE" en la página 489	Define la Tabla Temporal Global.
"DELETE" en la página 497	Suprime una o más filas de una tabla.
"DESCRIBE" en la página 504	Describe las columnas del resultado de una sentencia SELECT preparada.
"DISCONNECT" en la página 509	Finaliza una o más conexiones cuando no hay ninguna unidad de trabajo activa.
"DROP" en la página 512	Suprime objetos de la base de datos.
"END DECLARE SECTION" en la página 539	Marca el final de una sección de declaración de variables del lenguaje principal.
"EXECUTE" en la página 541	Ejecuta una sentencia de SQL preparada.
"EXECUTE IMMEDIATE" en la página 548	Prepara y ejecuta una sentencia de SQL.
"EXPLAIN" en la página 551	Captura información acerca del plan de acceso elegido.
"FETCH" en la página 556	Asigna valores de una fila a variables del lenguaje principal.
"FLUSH EVENT MONITOR" en la página 559	Graba el almacenamiento intermedio interno activo de un supervisor de sucesos.
"FLUSH PACKAGE CACHE" en la página 560	Elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes.
"FOR" en la página 561	Ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.
"FREE LOCATOR" en la página 564	Elimina la asociación entre una variable localizadora y su valor.
"GET DIAGNOSTICS" en la página 565	Se utiliza para obtener información acerca de la sentencia de SQL ejecutada previamente.
"GOTO" en la página 568	Se utiliza para ramificar a una etiqueta definida por el usuario dentro de un procedimiento de SQL.
"GRANT (Autorizaciones de base de datos)" en la página 570	Otorga autorizaciones sobre toda una base de datos.
"GRANT (Privilegios de índice)" en la página 574	Otorga el privilegio CONTROL en índices en la base de datos.
"GRANT (Privilegios de paquete)" en la página 576	Otorga privilegios para paquetes de la base de datos.
"GRANT (Privilegios de rutina)" en la página 579	Otorga privilegios para una rutina (función, método o procedimiento).

Sentencias de SQL soportadas

Tabla 1. Sentencias de SQL (continuación)

Sentencia de SQL	Propósito
"GRANT (Privilegios de esquema)" en la página 583	Otorga privilegios para un esquema.
"GRANT (Privilegios de secuencia)" en la página 586	Otorga privilegios en una secuencia.
"GRANT (Privilegios de servidor)" en la página 588	Otorga privilegios para consultar una fuente de datos específica.
"GRANT (privilegios de espacio de tabla)" en la página 590	Otorga privilegios para un espacio de tablas.
"GRANT (Privilegios de tabla, vista o apodo)" en la página 592	Otorga privilegios para tablas, vistas y apodos.
"IF" en la página 599	Selecciona una vía de acceso de ejecución basándose en la evaluación de una condición.
"INCLUDE" en la página 601	Inserta código o declaraciones en un programa fuente.
"INSERT" en la página 603	Inserta una o más filas en una tabla.
"ITERATE" en la página 613	Provoca que el flujo de control vuelva al principio de un bucle con etiqueta.
"LEAVE" en la página 614	Transfiere el control del programa fuera de un bucle o de una sentencia compuesta.
"LOCK TABLE" en la página 616	Impide que los procesos simultáneos cambien una tabla o impide que los procesos simultáneos utilicen una tabla.
"LOOP" en la página 618	Repite la ejecución de una sentencia o grupo de sentencias.
"MERGE" en la página 620	Actualiza un destino (tabla o vista) utilizando los datos de una fuente (resultado de una referencia de tabla).
"OPEN" en la página 629	Prepara un cursor que se utilizará para recuperar valores cuando se emita la sentencia FETCH.
"PREPARE" en la página 634	Prepara una sentencia de SQL (con parámetros opcionales) para su ejecución.
"REFRESH TABLE" en la página 644	Renueva los datos de una tabla de consultas materializadas.
"RELEASE (Conexión)" en la página 646	Coloca una o más conexiones en el estado pendiente de liberación.
"RELEASE SAVEPOINT" en la página 648	Libera un punto de salvaguarda dentro de una transacción.
"RENAME" en la página 649	Cambia el nombre de una tabla existente.
"RENAME TABLESPACE" en la página 651	Cambia el nombre de un espacio de tablas existente.
"REPEAT" en la página 652	Ejecuta una sentencia o grupo de sentencias hasta que una condición de búsqueda es verdadera.
"RESIGNAL" en la página 654	Se utiliza para retransmitir una condición de error o aviso.
"RETURN" en la página 656	Se utiliza para volver de una rutina.
"REVOKE (Autorizaciones de base de datos)" en la página 658	Revoca las autorizaciones de toda una base de datos.
"REVOKE (Privilegios de índice)" en la página 662	Revoca el privilegio CONTROL en índices determinados.
"REVOKE (Privilegios de paquete)" en la página 664	Revoca los privilegios de paquetes determinados en la base de datos.
"REVOKE (Privilegios de rutina)" en la página 667	Revoca privilegios para una rutina (función, método o procedimiento).

Tabla 1. Sentencias de SQL (continuación)

Sentencia de SQL	Propósito
"REVOKE (Privilegios de esquema)" en la página 670	Revoca privilegios para un esquema.
"REVOKE (Privilegios de secuencia)" en la página 672	Revoca los privilegios de una secuencia.
"REVOKE (Privilegios de servidor)" en la página 675	Revoca privilegios para consultar una fuente de datos específica.
"REVOKE (Privilegios de espacio de tabla)" en la página 677	Revoca el privilegio de utilización (USE) para un espacio de tablas determinado.
"REVOKE (Privilegios de tabla, vista o apodo)" en la página 679	Revoca privilegios para determinadas tablas, vistas o apodos.
"ROLLBACK" en la página 684	Termina una unidad de trabajo y restituye los cambios realizados por dicha unidad de trabajo.
"SAVEPOINT" en la página 687	Define un punto de salvaguarda dentro de una transacción.
"SELECT INTO" en la página 691	Especifica una tabla de resultados de no más de una fila y asigna los valores a variables del lenguaje principal.
"SET CONNECTION" en la página 693	Cambia el estado de una conexión de inactivo a actual, haciendo que la ubicación especificada sea el servidor actual.
"SET CURRENT DEFAULT TRANSFORM GROUP" en la página 695	Cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP.
"SET CURRENT DEGREE" en la página 697	Cambia el valor del registro especial CURRENT DEGREE.
"SET CURRENT EXPLAIN MODE" en la página 699	Cambia el valor del registro especial CURRENT EXPLAIN MODE.
"SET CURRENT EXPLAIN SNAPSHOT" en la página 702	Cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT.
"SET CURRENT ISOLATION" en la página 705	Cambia el valor del registro especial CURRENT ISOLATION.
"SET CURRENT LOCK TIMEOUT" en la página 706	Cambia el valor del registro especial CURRENT LOCK TIMEOUT.
"SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION" en la página 708	Cambia el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.
"SET CURRENT PACKAGE PATH" en la página 710	Asigna un valor al registro especial CURRENT PACKAGE PATH.
"SET CURRENT PACKAGESET" en la página 714	Establece el nombre de esquema para la selección de paquetes.
"SET CURRENT QUERY OPTIMIZATION" en la página 716	Cambia el valor del registro especial CURRENT QUERY OPTIMIZATION.
"SET CURRENT REFRESH AGE" en la página 719	Cambia el valor del registro especial CURRENT REFRESH AGE.
"SET ENCRYPTION PASSWORD" en la página 721	Establece la contraseña para el cifrado.
"SET EVENT MONITOR STATE" en la página 723	Activa o desactiva un supervisor de sucesos.
"SET INTEGRITY" en la página 725	Establece el estado pendiente de comprobación y comprueba los datos para las violaciones de restricciones.
"SET PASSTHRU" en la página 742	Abre una sesión para someter SQL nativo de fuente de datos directamente a la fuente de datos.

Sentencias de SQL soportadas

Tabla 1. Sentencias de SQL (continuación)

Sentencia de SQL	Propósito
"SET PATH" en la página 744	Cambia el valor del registro especial CURRENT PATH.
"SET SCHEMA" en la página 746	Cambia el valor del registro especial CURRENT SCHEMA.
"SET SERVER OPTION" en la página 748	Establece valores de opciones del servidor.
"SET SESSION AUTHORIZATION" en la página 750	Cambia el valor del registro especial SESSION USER.
"SET Variable" en la página 753	Asigna valores a variables de transición NEW.
"SIGNAL" en la página 758	Se utiliza para transmitir una condición de error o aviso.
"UPDATE" en la página 761	Actualiza los valores de una o varias columnas en una o más filas de una tabla.
"VALUES INTO" en la página 773	Especifica una tabla de resultados de no más de una fila y asigna los valores a variables del lenguaje principal.
"WHENEVER" en la página 775	Define las acciones que se deben tomar sobre la base de los códigos de retorno SQL.
"WHILE" en la página 777	Repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

Cómo se invocan las sentencias de SQL

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

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

- Incorporada en un programa de aplicación
- Incorporada en un procedimiento SQL.
- Preparada y ejecutada dinámicamente
- Emitida interactivamente

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

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

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

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

Incorporación de una sentencia a un programa de aplicación

Las sentencias de SQL pueden incorporarse en un programa fuente que se someterá a un precompilador. Se dice que dichas sentencias se *incorporan* al programa. Una sentencia incorporada se puede colocar en cualquier lugar del programa donde esté permitida una sentencia del lenguaje principal. Cada sentencia incorporada debe ir precedida de las palabras clave EXEC SQL.

Sentencias ejecutables

Una sentencia ejecutable incorporada a un programa de aplicación se ejecuta cada vez que habría de ejecutarse una sentencia del lenguaje del sistema principal si ésta se hubiera especificado en el mismo lugar. Por lo tanto, una sentencia dentro de un bucle se ejecuta cada vez que se ejecuta el bucle, y una sentencia dentro de una construcción condicional sólo se ejecuta cuando se satisface la condición.

Una sentencia incorporada puede contener referencias a variables del lenguaje principal. Una variable del lenguaje principal a la que se hace referencia de esta forma puede utilizarse de dos formas. Puede utilizarse:

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

En particular, todas las referencias a variables del lenguaje principal en expresiones y predicados se sustituyen de manera efectiva por los valores actuales de las variables; es decir, las variables se utilizan como entrada.

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

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

Sentencias no ejecutables

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

Incorporación de una sentencia a un procedimiento de SQL

Se pueden incorporar sentencias al cuerpo de un procedimiento SQL correspondiente a la sentencia CREATE PROCEDURE. Se dice que las sentencias de este tipo son sentencias incorporadas al procedimiento de SQL. Siempre que una descripción de sentencia de SQL haga referencia a una *variable del lenguaje principal*, podrá utilizarse una *variable de SQL* si la sentencia se ha incorporado a un procedimiento de SQL.

Preparación y ejecución dinámicas

Un programa de aplicación puede construir una sentencia de SQL en la forma de una serie de caracteres colocada en una variable de lenguaje principal. En general, la sentencia se construye a partir de algunos datos disponibles para el programa (por ejemplo, la entrada de una estación de trabajo). La sentencia (que no sea una sentencia de selección) construida puede prepararse para su ejecución por medio de la sentencia PREPARE (incorporada) y puede ejecutarse por medio de la sentencia EXECUTE (incorporada). Por otra parte, puede utilizarse una sentencia EXECUTE IMMEDIATE (incorporada) para preparar y para ejecutar la sentencia en un solo paso.

Una sentencia que se va a preparar dinámicamente no debe contener referencias a variables del lenguaje principal. En su lugar puede contener marcadores de parámetros. (Para obtener información acerca de las normas relacionadas con los marcadores de parámetros, consulte “PREPARE”.) Cuando se ejecuta la sentencia preparada, los marcadores de parámetros se sustituyen de manera efectiva por los valores actuales de las variables del lenguaje principal especificadas en la sentencia EXECUTE. Una vez preparada, una sentencia puede ejecutarse varias veces con distintos valores para las variables del lenguaje principal. Los marcadores de parámetros no están permitidos en la sentencia EXECUTE IMMEDIATE.

La ejecución satisfactoria o no satisfactoria de la sentencia se indica por medio de un código de retorno de SQL en la SQLCA tras completarse la sentencia EXECUTE (o EXECUTE IMMEDIATE). El código de retorno de SQL debe comprobarse, tal como se describía anteriormente. Para obtener más información, consulte el apartado “Códigos de retorno de SQL” en la página 9.

Invocación estática de una sentencia de selección

Una sentencia de selección puede incorporarse como parte de la sentencia DECLARE CURSOR (no ejecutable). Una sentencia de este tipo se ejecuta cada vez que se abre el cursor por medio de la sentencia OPEN (incorporada). Después de haberse abierto el cursor, la tabla de resultados puede recuperarse, una fila cada vez, mediante la realización de ejecuciones sucesivas de la sentencia FETCH.

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

Invocación dinámica de una sentencia de selección

Un programa de aplicación puede crear dinámicamente una sentencia de selección en forma de una serie de caracteres que se coloca en una variable del lenguaje principal. En general, la sentencia se construye a partir de algunos datos disponibles para el programa (por ejemplo, una consulta obtenida de una estación de trabajo). La sentencia que se construye de esta forma puede prepararse para su ejecución por medio de la sentencia PREPARE (incorporada) y puede hacerse referencia a ésta por medio de una sentencia DECLARE CURSOR (no ejecutable). Entonces, la sentencia se ejecuta cada vez que se abre el cursor por medio de la sentencia OPEN (incorporada). Después de haberse abierto el cursor, la tabla de resultados puede recuperarse, una fila cada vez, mediante la realización de ejecuciones sucesivas de la sentencia FETCH.

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

Invocación interactiva

Posibilidad de entrar sentencias de SQL desde una estación de trabajo como parte de la arquitectura del gestor de bases de datos. Se dice que una sentencia entrada así se emite interactivamente. Una sentencia de este tipo debe ser una sentencia ejecutable que no contenga marcadores de parámetros o referencias a las variables del lenguaje principal, pues ello sólo tiene sentido en el contexto de un programa de aplicación.

Utilización de SQL con otros sistemas principales

Las sintaxis de las sentencias de SQL presentan pequeñas variaciones entre los distintos tipos de sistemas principales (DB2 para z/OS, DB2 para iSeries, DB2 Universal Database). Con independencia de si las sentencias de SQL de una aplicación son dinámicas o estáticas, es importante — si la aplicación se destina al acceso a distintos sistemas principales de base de datos — asegurarse de que las sentencias de SQL y las opciones de precompilación/enlace reciben soporte en los sistemas de base de datos a los que accederá la aplicación.

Encontrará información adicional acerca de las sentencias de SQL que se utilizan en otros sistemas principales en la publicación *DB2 Universal Database for iSeries SQL Reference* y en la publicación *DB2 Universal Database for OS/390 and z/OS SQL Reference*.

Códigos de retorno de SQL

Un programa de aplicación que contiene sentencias de SQL ejecutables puede utilizar los valores SQLCODE o SQLSTATE para manejar los códigos de retorno de las sentencias de SQL. Hay dos maneras para que la aplicación acceda a estos valores.

- Incluir una estructura llamada SQLCA. La SQLCA incluye una variable entera llamada SQLCODE y una variable de serie de caracteres llamada SQLSTATE. En

REXX, se proporciona automáticamente una SQLCA. En otros lenguajes, la SQLCA se puede obtener utilizando la sentencia INCLUDE SQLCA.

- Si LANGLEVEL SQL92E se ha especificado como opción de precompilación, puede declararse una variable denominada SQLCODE o SQLSTATE en la sección de declaración de SQL del programa. Si no se ha declarado ninguna de estas variables en la sección de declaración de SQL, se da por supuesta la declaración de una variable denominada SQLCODE en algún otro lugar del programa. Con LANGLEVEL SQL92E, el programa no debe tener una sentencia INCLUDE SQLCA.

SQLCODE

El gestor de bases de datos establece una variable SQLCODE tras la ejecución de cada sentencia de SQL. Todos los gestores de bases de datos se ajustan al estándar SQL ISO/ANSI, tal como sigue:

- Si SQLCODE = 0 y SQLWARN0 está en blanco, la ejecución ha sido satisfactoria.
- Si SQLCODE = 100, no se ha encontrado “ningún dato”. Por ejemplo, una sentencia FETCH no ha devuelto ningún dato, porque el cursor estaba situado después de la última fila de la tabla de resultados.
- Si SQLCODE > 0 y no = 100, la ejecución ha sido satisfactoria con un aviso.
- Si SQLCODE = 0 y SQLWARN0 = 'W', la ejecución ha sido satisfactoria, pero se han establecido uno o más indicadores de aviso.
- Si SQLCODE < 0, la ejecución no ha sido satisfactoria.

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

SQLSTATE

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

Comentarios de SQL

Las sentencias de SQL estático pueden incluir comentarios SQL o del lenguaje principal. Los comentarios SQL se introducen mediante dos guiones.

A la utilización de los comentarios de SQL se aplican las normas siguientes:

- Los dos guiones deben estar en la misma línea, no separados por un espacio.
- Los comentarios pueden empezar donde exista un espacio válido (excepto dentro de un símbolo de delimitador o entre 'EXEC' y 'SQL').
- Los comentarios terminan al final de la línea.
- Los comentarios no están permitidos dentro de sentencias que están preparadas dinámicamente (utilizando PREPARE o EXECUTE IMMEDIATE).
- En COBOL, los guiones deben ir precedidos por un espacio.

Ejemplo: Este ejemplo muestra cómo incluir comentarios en una sentencia de SQL de un programa en C:

```
EXEC SQL
   CREATE VIEW PRJ_MAXPER         -- proyectos con más personal de soporte
     AS SELECT PROJNO, PROJNAME   -- número y nombre del proyecto
       FROM PROJECT
       WHERE DEPTNO = 'E21'       --  código dept de soporte de sistemas
       AND PRSTAFF > 1;
```

Información relacionada:

- "Sentencia select" en la publicación *Consulta de SQL, Volumen 1*
- "EXECUTE" en la página 541
- "OPEN" en la página 629
- "PREPARE" en la página 634
- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*

Acerca de las sentencias de control de SQL

Las sentencias de control son sentencias de SQL que permiten que el lenguaje de consulta estructurada pueda utilizarse de forma similar a la escritura de un programa en un lenguaje de programación estructurada. Las sentencias de control de SQL ofrecen la posibilidad de controlar el flujo lógico, declarar y establecer variables y manejar avisos y excepciones. Algunas sentencias de control de SQL incluyen otras sentencias de SQL anidadas. Las sentencias de control de SQL pueden utilizarse en el cuerpo de una rutina, un activador o una sentencia dinámica compuesta.

Puede hacerse referencia a los parámetros de SQL y a las variables de SQL en cualquier punto de la sentencia donde pueda especificarse una expresión o una variable del lenguaje principal. En las rutinas de SQL no pueden especificarse variables del lenguaje principal. Puede hacerse referencia a los parámetros de SQL en cualquier punto de la rutina y pueden calificarse con el nombre de rutina. Puede hacerse referencia a las variables de SQL en cualquier punto de la sentencia compuesta en la que se han declarado y pueden calificarse con el nombre de etiqueta que se ha especificado al principio de la sentencia compuesta.

Se considera que todos los parámetros de SQL y variables de SQL tienen posibilidad de nulos. El nombre de un parámetro de SQL o de una variable de SQL de una rutina de SQL puede ser igual al nombre de una columna de una tabla o vista a la que se haga referencia en la rutina. En este caso, el nombre debe calificarse explícitamente para indicar si es una columna, una variable de SQL o un parámetro de SQL.

Si el nombre no se ha calificado, las normas siguientes describen si el nombre hace referencia a la columna o bien a la variable o al parámetro de SQL:

- Si las tablas y vistas que se han especificado en el cuerpo de la rutina de SQL existen en el momento de crearse la rutina, primero se comprueba si el nombre es un nombre de columna. Si no se encuentra como columna, se comprueba si es un nombre de variable de SQL o de parámetro de SQL.
- Si las tablas o vistas a las que se hace referencia no existen en el momento de crearse la rutina, primero se comprueba si el nombre es un nombre de variable de SQL o de parámetro de SQL. Si no se encuentra, se da por supuesto que es una columna.

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

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

ALLOCATE CURSOR

La sentencia ALLOCATE CURSOR asigna un cursor para el conjunto de resultados identificado por la variable localizadora de conjuntos de resultados. Para obtener más información acerca de las variables de localizador de conjuntos de resultados, consulte la descripción de la sentencia ASSOCIATE LOCATORS.

Invocación:

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

Autorización:

No se necesita.

Sintaxis:

Descripción:

nombre-cursor
Indica el cursor. El nombre no debe identificar un cursor que ya esté declarado en el procedimiento SQL fuente (SQLSTATE 24502).

CURSOR FOR RESULT SET *variable-localizadora-cr*

Indica una variable localizadora de conjuntos de resultados que se ha declarado en el procedimiento SQL fuente, de acuerdo con las normas para las variables del lenguaje principal. Para obtener más información acerca de la declaración de variables de SQL, consulte "Sentencia compuesta (procedimiento)".

La variable localizadora de conjuntos de resultados debe contener un valor válido, tal como lo devuelve la sentencia ASSOCIATE LOCATORS de SQL (SQLSTATE 24501).

Normas:

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

Ejemplos:

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

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

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139
- "ASSOCIATE LOCATORS" en la página 103

ALTER BUFFERPOOL

La sentencia ALTER BUFFERPOOL se utiliza para realizar lo siguiente:

- modificar el tamaño de la agrupación de almacenamientos intermedios en todas las particiones o en una sola partición
- activar o desactivar la utilización del almacenamiento ampliado.
- añadir esta definición de agrupación de almacenamientos intermedios a un nuevo grupo de particiones de base de datos
- modificar el área de bloque de la agrupación de almacenamientos intermedios para la E/S basada en bloques.

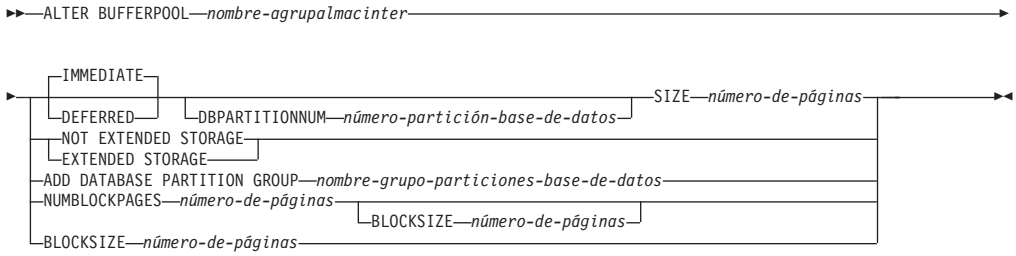
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

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

DBPARTITIONNUM *número-partición-base-de-datos*
Especifica la partición en la que se modifica el tamaño de la agrupación de almacenamientos intermedios. La partición debe estar en uno de los grupos de particiones de base de datos de la agrupación de almacenamientos intermedios (SQLSTATE 42729). Si no se especifica esta cláusula, el tamaño de la agrupación de almacenamientos intermedios se modifica en todas las particiones en las que existe la agrupación de almacenamientos intermedios que utilizaba el tamaño por omisión para la agrupación de almacenamientos intermedios (que no tenía especificado un tamaño en la *cláusula-except-on-db-partitions* de la sentencia CREATE BUFFERPOOL).

SIZE *número-de-páginas*
El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas.

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

DEFERRED
El tamaño de la agrupación de almacenamientos intermedios cambiará cuando vuelva a activarse la base de datos (es necesario desconectar todas las aplicaciones de la base de datos). No se necesita espacio de memoria reservado; durante la activación, DB2 asignará la memoria necesaria del sistema.

NOT EXTENDED STORAGE
Aunque el almacenamiento ampliado esté habilitado, las páginas que se eliminan de esta agrupación de almacenamientos intermedios no se colocan en antememoria en el almacenamiento ampliado.

EXTENDED STORAGE
Si el almacenamiento ampliado está habilitado, puede utilizarse como antememoria secundaria para las páginas que se eliminan de la agrupación de almacenamientos intermedios. (El almacenamiento ampliado se habilita estableciendo los parámetros de configuración de base de datos NUM_ESTORE_SEGS y ESTORE_SEG_SIZE en valores distintos de cero.)

ADD DATABASE PARTITION GROUP *nombre-grupo-particiones-base-de-datos*
Añade este grupo de particiones de base de datos a la lista de grupos de particiones de base de datos a la que se aplica la definición de agrupación de almacenamientos intermedios. Para cualquier partición del grupo de particiones de base de datos que todavía no tenga definida la agrupación de almacenamientos intermedios, la agrupación de almacenamientos intermedios se crea en la partición utilizando el tamaño por omisión que se ha especificado para la agrupación de almacenamientos intermedios. Los espacios de tabla de *nombre-grupo-particiones-base-de-datos* pueden especificar esta agrupación de almacenamientos intermedios. El grupo de particiones de base de datos debe existir actualmente en la base de datos (SQLSTATE 42704).

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

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

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
- Sólo el tamaño de agrupación de almacenamientos intermedios puede cambiarse dinámicamente (inmediatamente). Todos los demás cambios se aplican de forma diferida y sólo entrarán en vigor tras la reactivación de la base de datos.

- Si la sentencia se ejecuta como sentencia diferida, se aplicará lo siguiente: Aunque la definición de agrupación de almacenamientos intermedios sea transaccional y los cambios realizados en la definición de agrupación de almacenamientos intermedios aparezcan reflejados en la tablas de catálogo durante la confirmación, ningún cambio realizado en la agrupación de almacenamientos intermedios real entrará en vigor hasta la próxima vez que se inicie la base de datos. Los atributos actuales de la agrupación de almacenamientos intermedios existirán hasta entonces y no afectará a la agrupación de almacenamientos intermedios mientras tanto. Las tablas que se han creado en los espacios de tabla de los nuevos grupos de particiones de base de datos utilizarán la agrupación de almacenamientos intermedios por omisión. Por omisión, la sentencia es IMMEDIATE cuando se aplica dicha palabra clave.
- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones.
- Una agrupación de almacenamientos intermedios que actualmente está utilizando el almacenamiento ampliado no puede modificarse para que utilice la entrada/salida (E/S) basada en bloques. Una agrupación de almacenamientos intermedios no puede modificarse para que utilice simultáneamente el almacenamiento ampliado y la E/S basada en bloques.

ALTER DATABASE PARTITION GROUP

La sentencia ALTER DATABASE PARTITION GROUP se utiliza para:

- añadir una o más particiones a un grupo de particiones de base de datos
- eliminar una o más particiones de un grupo de particiones de base de datos.

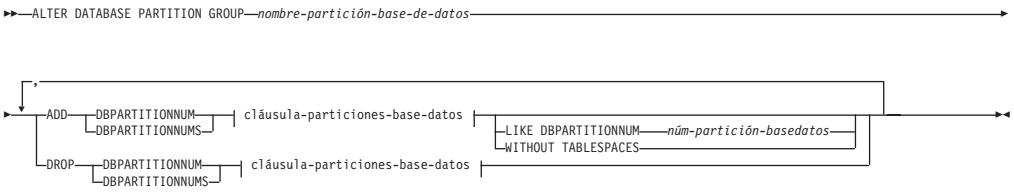
Invocación:

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

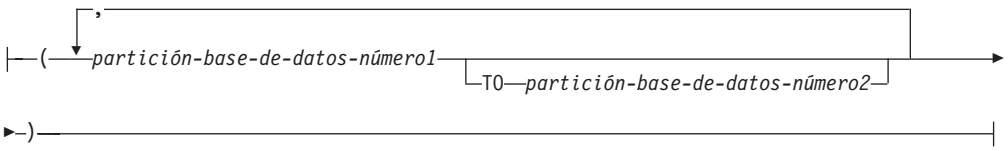
Autorización:

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

Sintaxis:



cláusula-particiones-base-datos:



Descripción:

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

ADD DBPARTITIONNUM
Especifica la partición o particiones específicas que deben añadirse al grupo de particiones de base de datos. DBPARTITIONNUMS es un sinónimo de DBPARTITIONNUM. Cualquier partición que se especifique no deberá estar definida todavía en el grupo de particiones de base de datos (SQLSTATE 42728).

DROP DBPARTITIONNUM
Especifica la partición o particiones específicas que deben eliminarse del grupo de particiones de base de datos. DBPARTITIONNUMS es un sinónimo de

DBPARTITIONNUM. Cualquier partición que se especifique ya deberá estar definida en el grupo de particiones de base de datos (SQLSTATE 42729).

cláusula-particiones-base-de-datos
Especifica la partición o particiones que se deben añadir o eliminar.

partición-base-de-datos-número1
Especifique un número de partición determinado.

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

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

WITHOUT TABLESPACES
Especifica que los espacios de la tabla por omisión no se crean en la partición o particiones que se acaban de añadir. La sentencia ALTER TABLESPACE, con la cláusula FOR DBPARTITIONNUM, debe utilizarse para definir los contenedores que deben utilizarse con los espacios de tabla que se han definido en este grupo de particiones de base de datos. Si no se especifica esta opción, se especificarán los contenedores por omisión en las particiones que acaban de añadirse para cada espacio de tablas que se haya definido en el grupo de particiones de base de datos.

Normas:

- Cada partición que se haya especificado mediante un número deberá definirse en el archivo `db2nodes.cfg` (SQLSTATE 42729).
- Cada *número-partición-base-de-datos* que aparezca en la lista de la cláusula ON DBPARTITIONNUMS deberá corresponder a una partición exclusiva (SQLSTATE 42728).
- Un número de partición válido está comprendido entre 0 y 999 inclusive (SQLSTATE 42729).
- Una partición no puede aparecer en las cláusulas ADD y DROP al mismo tiempo (SQLSTATE 42728).
- Como mínimo, deberá quedar una partición en el grupo de particiones de base de datos. La última partición no puede eliminarse de un grupo de particiones de base de datos (SQLSTATE 428C0).
- Si no se ha especificado la cláusula LIKE DBPARTITIONNUM ni la cláusula WITHOUT TABLESPACES durante la adición de una partición, el valor por omisión consiste en utilizar el número de partición más bajo de las particiones existentes en el grupo de particiones de base de datos (es decir, el 2) y continuar como si se hubiera especificado LIKE DBPARTITIONNUM 2. Para que se utilice una partición existente como valor por omisión, ésta debe tener contenedores definidos para todos los espacios de tabla del grupo de particiones de base de datos (la columna IN_USE de SYSCAT.DBPARTITIONGROUPDEF no es 'T').

Notas:

- ***Compatibilidades***

  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - NODES puede especificarse en lugar de DBPARTITIONNUMS
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
- Cuando se añade una partición a un grupo de particiones de base de datos, se crea una entrada de catálogo para la partición (véase SYSCAT.DBPARTITIONGROUPDEF). El mapa de particionamiento se cambia inmediatamente para que incluya la nueva partición junto con un indicador (IN_USE) de que la partición está en el mapa de particionamiento si se cumple una de estas condiciones:
  - no hay espacios de tabla definidos en el grupo de particiones de base de datos o
  - no hay tablas definidas en los espacios de tabla que se han definido en el grupo de particiones de base de datos y no se ha especificado la cláusula WITHOUT TABLESPACES.

  El mapa de particionamiento no se cambia y se habilita el indicador (IN_USE) para indicar que la partición no se incluye en el mapa de particionamiento si:
  - existen tablas en los espacios de tabla del grupo de particiones de base de datos o
  - existen espacios de tabla en el grupo de particiones de base de datos y se ha especificado la cláusula WITHOUT TABLESPACES.

  Para cambiar la correlación del particionamiento, debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP. Esto redistribuye los datos, cambia el mapa de particionamiento y cambia el indicador. Si se ha especificado la cláusula WITHOUT TABLESPACES, los contenedores de espacios de tabla se deben añadir antes de intentar redistribuir los datos.
- Cuando se elimina una partición de un grupo de particiones de base de datos, se actualiza la entrada de catálogo de la partición (véase SYSCAT.DBPARTITIONGROUPDEF). Si no se ha definido ninguna tabla en los espacios de tabla que se han definido en el grupo de particiones de base de datos, la correlación del particionamiento cambiará inmediatamente para excluir la partición eliminada y se eliminará la entrada de la partición del grupo de particiones de base de datos. Si hay tablas, el mapa de particionamiento no se modifica y se habilita el indicador (IN_USE) para indicar que la partición está esperando su eliminación. Debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP para redistribuir los datos y eliminar la entrada correspondiente a la partición del grupo de particiones de base de datos.

Ejemplo:

Suponga que tiene una base de datos de seis particiones que presenta las particiones siguientes: 0, 1, 2, 5, 7 y 8. Se añaden dos particiones al sistema con los números de partición 3 y 6.

- Supongamos que desea añadir las particiones 3 y 6 a un grupo de particiones de base de datos denominado MAXGROUP y que tiene contenedores de espacio de tablas como los de la partición 2. La sentencia es la siguiente:

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

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

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

Conceptos relacionados:

- "Particionamiento de datos entre múltiples particiones" en la publicación *Consulta de SQL, Volumen 1*

ALTER FUNCTION

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

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema de la función
- Autorización del usuario que define la función, tal como se ha registrado en la columna DEFINER de SYSCAT.ROUTINES

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

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos

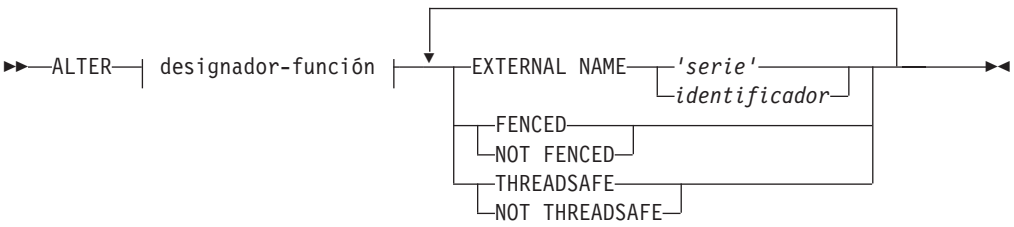
Para modificar una función con el fin de que no esté delimitada, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos

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

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:



Descripción:

designador-función
Identifica de forma exclusiva la función que va a modificarse. Para obtener

más información, consulte el apartado Elementos comunes de la sintaxis "Elementos comunes de la sintaxis" en la página viii.

EXTERNAL NAME *'serie'* **o** *identificador*
Identifica el nombre del código escrito por el usuario que implementa la función. Esta opción sólo puede especificarse cuando se alteran funciones externas (SQLSTATE 42849).

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

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

PRECAUCIÓN:
La utilización de NOT FENCED para las funciones que no se han codificado, revisado y probado de forma adecuada puede comprometer la integridad de DB2. DB2 dispone de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no puede garantizar la integridad completa cuando se utilizan funciones NOT FENCED definidas por el usuario.

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

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

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

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

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

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

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

Notas:

- No es posible modificar una función que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).
- Las funciones declaradas como LANGUAGE SQL, las funciones con origen o las funciones de plantilla no pueden modificarse (SQLSTATE 42917).

Ejemplo:

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

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

Información relacionada:

- "CREATE FUNCTION (Tabla externa OLE DB)" en la página 241
- "CREATE FUNCTION (Escalar externa)" en la página 198
- "CREATE FUNCTION (Tabla externa)" en la página 223
- "Elementos comunes de la sintaxis" en la página viii

ALTER METHOD

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

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
  - Privilegio ALTERIN para el esquema del tipo
  - Autorización del usuario que define el tipo, tal como se ha registrado en la columna DEFINER de SYSCAT.DATATYPES

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:

Descripción:

designador-método
Identifica de forma exclusiva el método que va a modificarse. Para obtener más información, consulte el apartado Elementos comunes de la sintaxis "Elementos comunes de la sintaxis" en la página viii.

EXTERNAL NAME *'serie'* **o** *identificador*
Identifica el nombre del código escrito por el usuario que implementa el método. Esta opción sólo puede especificarse cuando se alteran métodos externos (SQLSTATE 42849).

Notas:

- No es posible modificar un método que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).
- Los métodos declarados como LANGUAGE SQL no pueden modificarse (SQLSTATE 42917).
- Los métodos declarados como LANGUAGE CLR no se pueden modificar (SQLSTATE 42849).
- El método especificado debe tener un cuerpo antes de que pueda modificarse (SQLSTATE 42704).

Ejemplo:

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

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

Información relacionada:

- "CREATE METHOD" en la página 289
- "Elementos comunes de la sintaxis" en la página viii

ALTER NICKNAME

La sentencia ALTER NICKNAME modifica la información de apodo asociada a un objeto de fuente de datos (por ejemplo, una tabla, una vista o un archivo). Esta sentencia modifica la información que se almacena en la base de datos federada mediante:

- La modificación de los nombres de columna locales para las columnas del objeto de fuente de datos
- La modificación de los tipos de datos locales para las columnas del objeto de fuente de datos
- La adición, el establecimiento o la eliminación de opciones de apodo y columna
- La adición o eliminación de una clave primaria
- La adición o eliminación de una o varias restricciones de comprobación, de referencia o exclusivas
- La modificación de uno o varios atributos de restricción de comprobación o de referencia

Invocación:

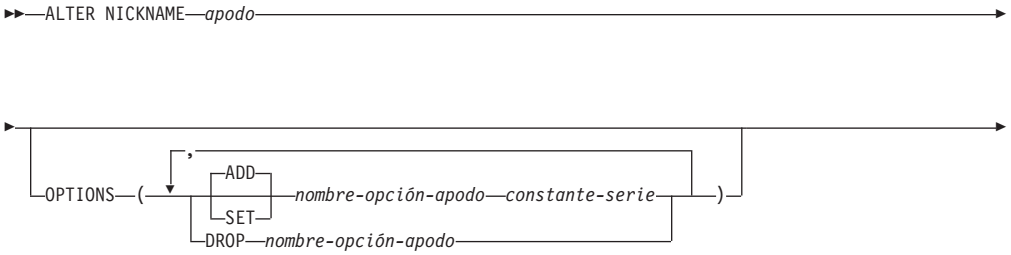
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

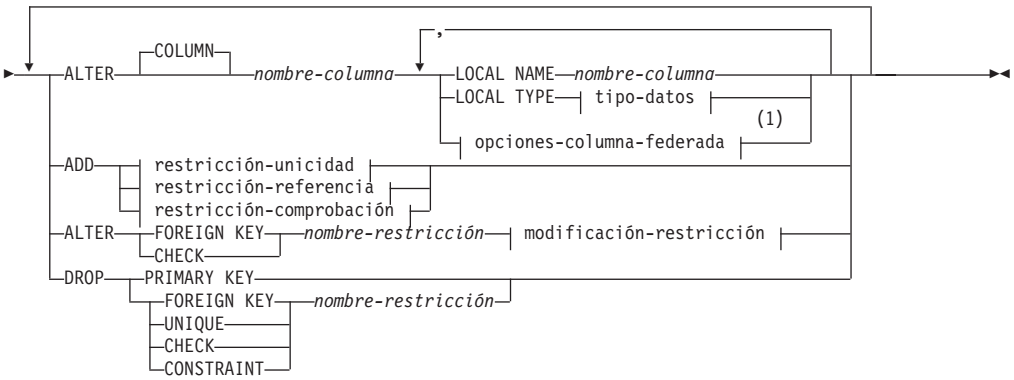
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio ALTER para el apodo especificado en la sentencia
- Privilegio CONTROL para el apodo especificado en la sentencia
- Privilegio ALTERIN para el esquema, si existe el nombre de esquema del apodo
- Definidor del apodo, tal como está registrado en la columna DEFINER de la vista del catálogo SYSCAT.TABLES
- Autorización SYSADM o DBADM

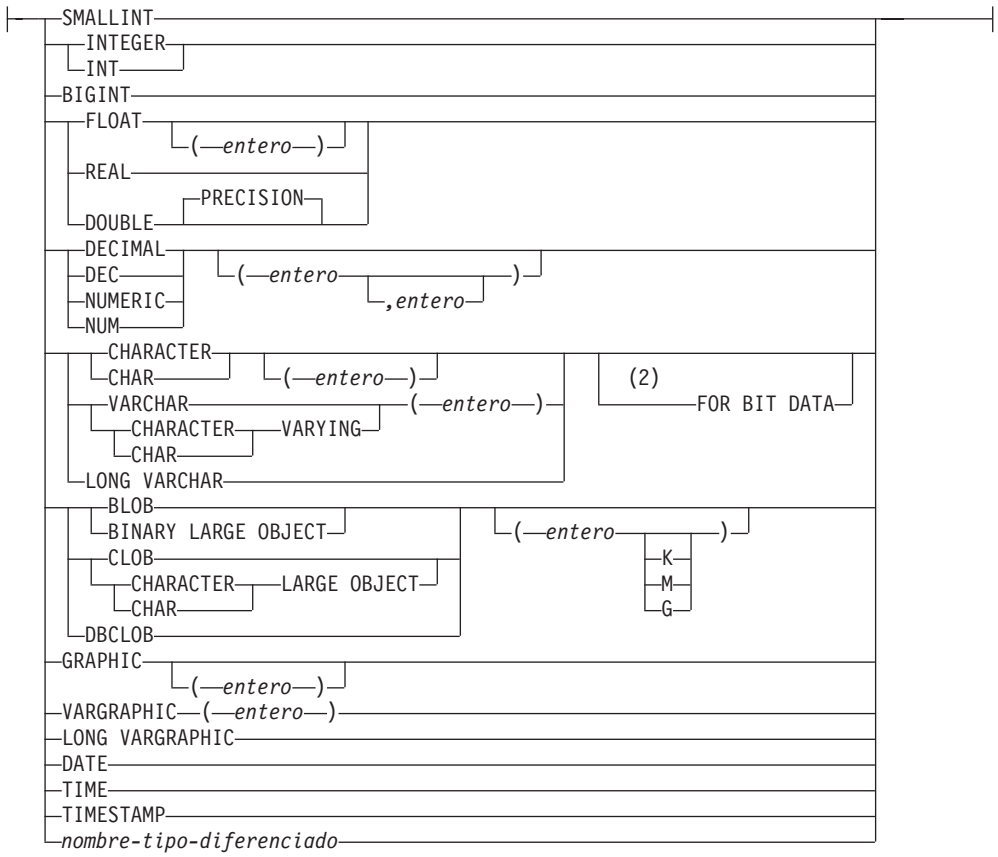
Sintaxis:



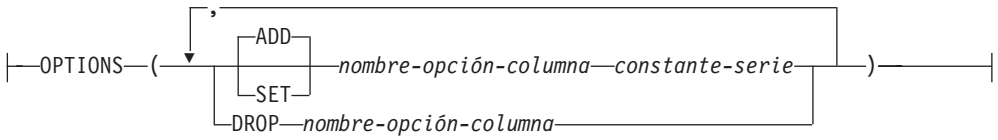
ALTER NICKNAME



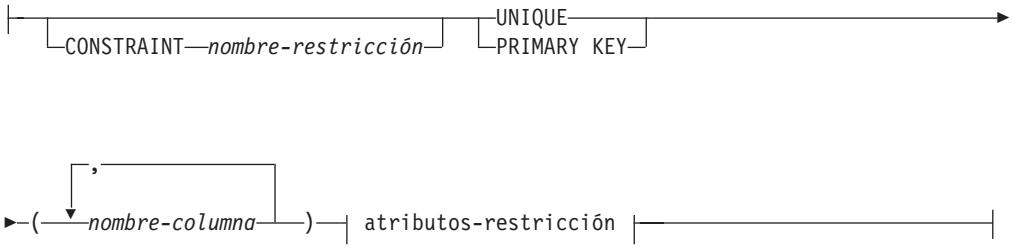
tipo-datos:



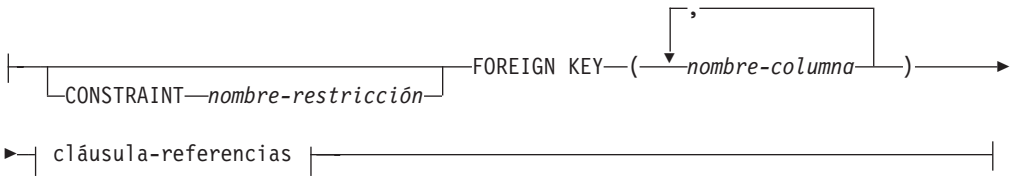
opciones-columna-federada:



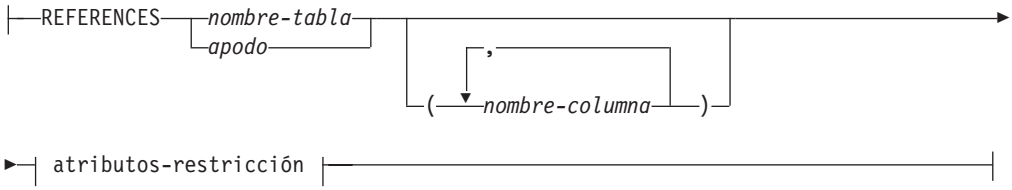
restricción-unicidad:



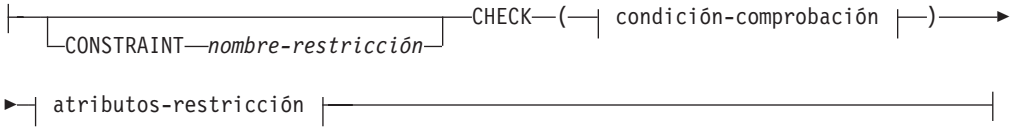
restricción-referencia:



cláusula-referencias:

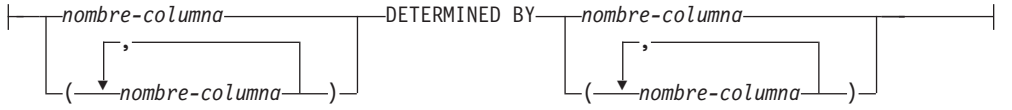


restricción-comprobación:

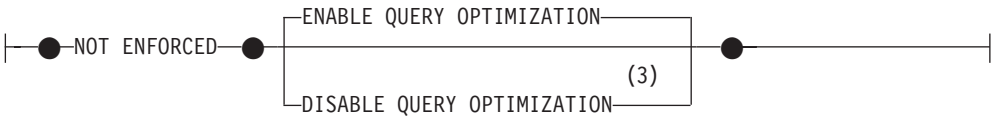


condición-comprobación:

dependencia-funcional:



atributos-restricción:



modificación-restricción:

Notas:

1 Si debe especificar la cláusula opciones-columna-federada además del parámetro LOCAL NAME, el parámetro LOCAL TYPE o ambos, debe especificar la cláusula opciones-columna-federada en último lugar.

2 La cláusula FOR BIT DATA se puede especificar en cualquier orden con las restricciones de columna siguientes.

3 No se da soporte a DISABLE QUERY OPTIMIZATION para una restricción de clave primaria o exclusiva.

Descripción:

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

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

ADD
Añade una opción de apodo.

SET
Cambia el valor de una opción de apodo.

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

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

DROP *nombre-opción-apodo*
Desactiva una opción de apodo.

ALTER COLUMN *nombre-columna*
Los nombres de la columna que se deben modificar. El *nombre-columna* es el nombre actual del servidor federado para la columna de la tabla o la vista de la fuente de datos. El *nombre-columna* debe identificar una columna existente del apodo (SQLSTATE 42703). No puede hacer referencia al mismo nombre de columna varias veces en la misma sentencia ALTER NICKNAME (SQLSTATE 42711).

LOCAL NAME *nombre-columna*
Especifica un nuevo nombre, *nombre-columna*, por el que el servidor federado debe hacer referencia a la columna que se va a modificar. El nuevo nombre no

puede estar calificado y no se puede utilizar el mismo nombre para más de una columna del apodo (SQLSTATE 42711).

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

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

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

ADD
Añade una opción de columna.

SET
Cambia el valor de una opción de columna.

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

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

DROP *nombre-opción-columna*
Desactiva una opción de columna.

ADD *restricción-unicidad*
Define una restricción exclusiva. Consulte la descripción de la sentencia "CREATE NICKNAME".

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

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

ALTER FOREIGN KEY *nombre-restricción*
Altera los atributos de restricción de la restricción de referencia *nombre-restricción*. Consulte la descripción de la sentencia "CREATE NICKNAME". El *nombre-restricción* debe identificar una restricción de referencia existente (SQLSTATE 42704).

ALTER CHECK *nombre-restricción*
Altera los atributos de restricción de la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente (SQLSTATE 42704).

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

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

DISABLE QUERY OPTIMIZATION
La restricción no puede utilizarse para la optimización de la consulta.

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

DROP FOREIGN KEY *nombre-restricción*
Elimina la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia existente, definida en el apodo.

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

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

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

Normas:

- Si se utiliza un apodo en una vista, un método de SQL o una función de SQL, o contiene definiciones de restricciones informativas, la sentencia ALTER NICKNAME no se puede utilizar para cambiar los nombres locales ni los tipos de datos para las columnas del apodo (SQLSTATE 42893). Sin embargo, la sentencia se puede utilizar para añadir, establecer o descartar opciones de columna, opciones de apodo o restricciones informativas.
- Si una definición de tabla de consultas materializadas hace referencia a un apodo, la sentencia ALTER NICKNAME no se puede utilizar para cambiar los nombres locales, los tipos de datos, las opciones de columna ni las opciones de apodo (SQLSTATE 42893). Sin embargo, la sentencia se puede utilizar para añadir, modificar o descartar restricciones informativas.
- No se puede especificar una opción de columna más de una vez en la misma sentencia ALTER NICKNAME (SQLSTATE 42853). Cuando se habilita, restaura o desactiva una opción de columna, no afecta a ninguna otra opción de columna que se esté utilizando.
- Para apodos relacionales, la sentencia ALTER NICKNAME de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Para apodos no relacionales, la sentencia ALTER NICKNAME de una unidad de trabajo (UOW) determinada no puede procesarse bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW

- Una sentencia SELECT de la misma UOW ya hace referencia a un apodo al que se hace referencia en esta sentencia
- Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia

Notas:

- Si se utiliza la sentencia ALTER NICKNAME para cambiar el nombre local de una columna de un apodo, las consultas deben hacer referencia a esa columna por su nuevo nombre.
- Cuando se cambia la especificación local del tipo de datos de una columna, el gestor de bases de datos invalida cualquier estadística (HIGH2KEY, LOW2KEY, etcétera) reunida para esa columna.

Ejemplos:

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

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

Ejemplo 2: El apodo EMPLOYEE hace referencia a una tabla DB2 UDB para z/OS y OS/390 denominada EMP. Además, SALARY es el nombre local que hace referencia a EMP_SAL, una de las columnas de esta tabla. El tipo de datos de la columna, FLOAT, se correlaciona con el tipo de datos local, DOUBLE. Cambie la correlación para que FLOAT se correlacione con DECIMAL (10, 5).

```
ALTER NICKNAME EMPLOYEE
   ALTER COLUMN SALARY
   LOCAL TYPE DECIMAL(10,5)
```

Ejemplo 3: Indica que en una tabla Oracle, una columna con el tipo de datos VARCHAR no tiene blancos de cola. El apodo para la tabla es NICK2 y el nombre local para la columna es COL1.

```
ALTER NICKNAME NICK2
  ALTER COLUMN COL1
  OPTIONS (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

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

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

Tareas relacionadas:

- "Altering a nickname" en la publicación *Federated Systems Guide*

Información relacionada:

- "CREATE NICKNAME" en la página 295

- "Nickname column options for federated systems" en la publicación *Federated Systems Guide*

ALTER PROCEDURE

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

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema del procedimiento
- Autorización del usuario que define el procedimiento, tal como se ha registrado en la columna DEFINER de SYSCAT.ROUTINES

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

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos

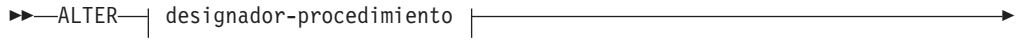
Para modificar un procedimiento con el fin de que no esté delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las siguientes:

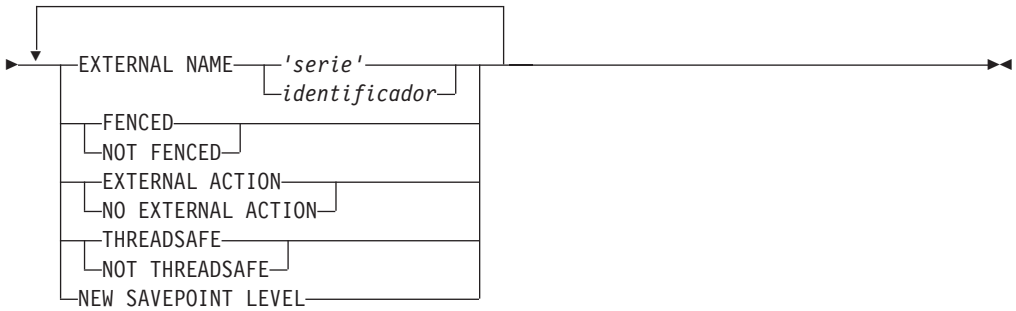
- Autorización SYSADM o DBADM
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos

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

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:





Descripción:

designador-procedimiento
Identifica de forma exclusiva el procedimiento que va a modificarse. Para obtener más información, consulte el apartado Elementos comunes de la sintaxis "Elementos comunes de la sintaxis" en la página viii.

EXTERNAL NAME *'serie'* **o** *identificador*
Identifica el nombre del código escrito por el usuario que implementa el procedimiento. Esta opción sólo puede especificarse cuando se alteran procedimientos externos (SQLSTATE 42849). La cláusula EXTERNAL NAME no se puede modificar en los procedimientos que se han declarado como LANGUAGE SQL (SQLSTATE 42917).

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

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

PRECAUCIÓN:
La utilización de NOT FENCED para los procedimientos que no se han codificado, revisado y probado de forma adecuada puede comprometer la integridad de DB2. DB2 toma algunas precauciones contra muchas de las anomalías más habituales debido a descuidos, pero no puede garantizar la completa integridad de los datos cuando se utilizan procedimientos almacenados NOT FENCED.

Esta opción sólo puede especificarse cuando se alteran procedimientos externos (SQLSTATE 42849).

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

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

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

Las cláusulas FENCED o NOT FENCED no se pueden modificar en los procedimientos que se han declarado como LANGUAGE SQL (SQLSTATE 42917).

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

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

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

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

Esta opción sólo puede especificarse cuando se alteran procedimientos externos (SQLSTATE 42849).

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

Las cláusulas THREADSAFE o NOT THREADSAFE no se pueden modificar en procedimientos que se han declarado como LANGUAGE SQL (SQLSTATE 42917).

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

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

Normas:

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

Ejemplo:

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

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

Información relacionada:

- “CREATE PROCEDURE” en la página 307
- “Elementos comunes de la sintaxis” en la página viii

ALTER SEQUENCE

La sentencia ALTER SEQUENCE puede utilizarse para cambiar una secuencia en cualquiera de estos modos:

- Reiniciando la secuencia
- Cambiando el incremento entre valores de secuencia futuros
- Estableciendo o eliminando los valores mínimo y máximo
- Cambiando los números de secuencia almacenados en antememoria
- Cambiando el atributo que determina si la secuencia puede realizar un ciclo o no
- Cambiando la posibilidad de que los números de secuencia deban generarse en orden de petición

Invocación:

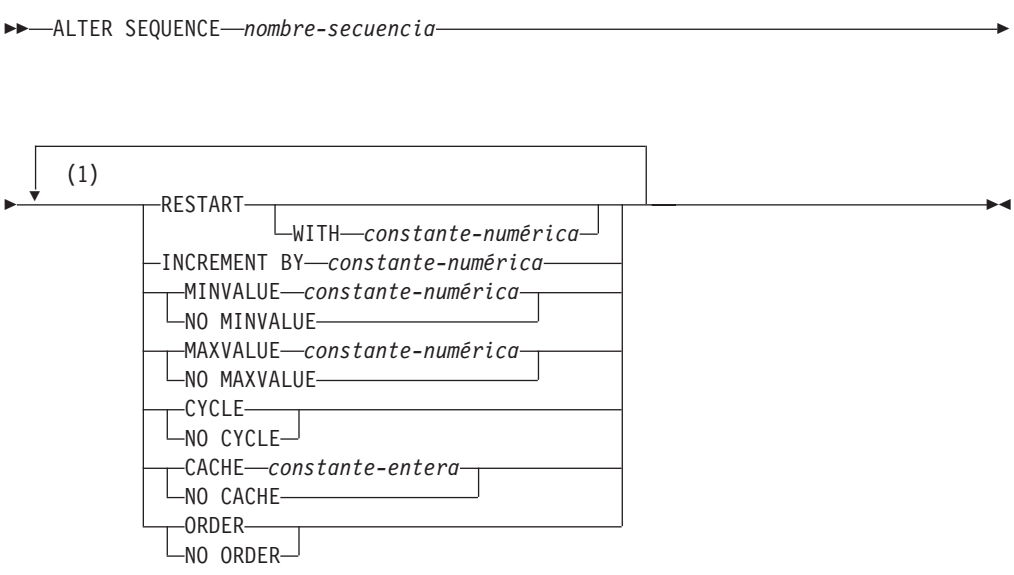
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

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

Sintaxis:



Notas:

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

Descripción:

nombre-secuencia
Identifica la secuencia que va a cambiarse. El nombre (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre en el esquema especificado explícita o implícitamente, se devuelve un error (SQLSTATE 42704). El *nombre-secuencia* no debe ser una secuencia generada por el sistema para una columna de identidad (SQLSTATE 428FB).

RESTART
Reinicia la secuencia. Si no se especifica *constante-numérica*, la secuencia se reinicia en el valor especificado implícita o explícitamente como el valor inicial en la sentencia CREATE SEQUENCE que ha creado originalmente la secuencia.

WITH *constante-numérica*
Reinicia la secuencia con el valor especificado. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

INCREMENT BY *constante-numérica*
Especifica el intervalo entre valores consecutivos de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820) y sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, se trata de una secuencia descendente. Si este valor es 0 o positivo, es una secuencia ascendente tras la sentencia ALTER.

MINVALUE o **NO MINVALUE**
Especifica el valor mínimo en el que una secuencia descendente pasa por un ciclo o deja de generar valores o en el que una secuencia ascendente pasa por un ciclo después de alcanzar el valor máximo.

MINVALUE *constante-numérica*
Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor que o igual al valor máximo (SQLSTATE 42815).

NO MINVALUE
Para una secuencia ascendente, el valor es el valor de inicio original. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos asociado con la secuencia.

MAXVALUE o **NO MAXVALUE**
Especifica el valor máximo en el que una secuencia ascendente pasa por un ciclo o deja de generar valores o en el que una secuencia descendente pasa por un ciclo después de alcanzar el valor mínimo.

MAXVALUE *constante-numérica*
Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser mayor que o igual al valor mínimo (SQLSTATE 42815).

NO MAXVALUE
Para una secuencia ascendente, el valor es el valor máximo del tipo de datos asociado con la secuencia. Para una secuencia descendente, el valor es el valor de inicio original.

CYCLE o **NO CYCLE**
Especifica si la secuencia debe continuar generando valores después de alcanzar el valor máximo o el valor mínimo. El límite de la secuencia puede alcanzarse con el siguiente valor que coincida exactamente con la condición de límite o excediendo el valor.

CYCLE
Especifica que se continúan generando valores para esta secuencia después de haber alcanzado el valor máximo o mínimo. Si se utiliza esta opción, cuando una secuencia ascendente haya alcanzado su valor máximo, generará su valor mínimo; o cuando una secuencia descendente haya alcanzado su valor mínimo, generará su valor máximo. Los valores máximo y mínimo para la secuencia determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, DB2 puede generar valores duplicados para la secuencia.

NO CYCLE
Especifica que no se generarán valores para la secuencia una vez que se haya alcanzado el valor máximo o mínimo para la secuencia.

CACHE o **NO CACHE**
Especifica si se deben mantener algunos valores preasignados en memoria para obtener un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste.

CACHE *constante-entera*
Especifica el número máximo de valores de secuencia que se preasignan y se mantienen en memoria. La preasignación y el almacenamiento de valores en la antememoria reducen la E/S síncrona en las anotaciones cronológicas cuando se generan valores para la secuencia.

En el caso de producirse una anomalía del sistema, todos los valores de secuencia almacenados en antememoria que no se han utilizando en sentencias confirmadas se pierden (es decir, no se utilizarán nunca). El valor especificado para la opción CACHE es el número máximo de valores de secuencia que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42815).

NO CACHE
Especifica que los valores de la secuencia no se deben preasignar. Asegura que no haya ninguna pérdida de valores en el caso de producirse una anomalía del sistema, un cierre o una desactivación de la base de datos. Cuando se especifica esta opción, los valores de la secuencia no se almacenan en la antememoria. En este caso, cada petición de un valor nuevo para la secuencia produce E/S síncrona en las anotaciones cronológicas.

ORDER o **NO ORDER**
Especifica si los números de secuencia deben generarse según el orden de petición.

ORDER
Especifica que los números de secuencia se generan según el orden de petición.

NO ORDER
Especifica que los números de secuencia no necesitan generarse según el orden de petición.

Notas:

- *Compatibilidades*
  - Para mantener la compatibilidad con las versiones anteriores de DB2 y por coherencia:
    - Puede utilizarse una coma para separar varias opciones en una secuencia.
  - También recibe soporte la sintaxis siguiente:
    - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER
- Sólo los números de secuencia futuros se ven afectados por la secuencia ALTER SEQUENCE.
- El tipo de datos de una secuencia no se puede cambiar. En lugar de ello, elimine la secuencia y vuelva a crear otra especificando el tipo de datos deseado para la secuencia nueva.
- Todos los valores almacenados en antememoria se pierden cuando se modifica una secuencia.
- Después de reiniciar una secuencia o de cambiar a CYCLE, es posible que los números de secuencia sean valores duplicados de los números generados por la secuencia anteriormente.

Ejemplos:

Ejemplo 1: Una posible razón para especificar RESTART sin un valor numérico puede ser para restablecer la secuencia en el valor START WITH. En este ejemplo, el objetivo es generar los números del 1 hasta el número de filas de tabla y, a continuación, insertar los números en una columna añadida a la tabla utilizando tablas temporales. También se podría utilizar para obtener resultados en los que todas las filas resultantes estén numeradas:

```
ALTER SEQUENCE ORG_SEQ RESTART
  SELECT NEXT VALUE FOR ORG_SEQ, ORG.* FROM ORG
```

Ejemplos relacionados:

- "DbSeq.java -- How to create, alter and drop a sequence in a database (JDBC)"
- "DbSeq.out -- HOW TO USE A SEQUENCE IN A DATABASE. Connect to 'sample' database using JDBC type 2 driver (JDBC)"

ALTER SERVER

La sentencia ALTER SERVER se utiliza para:

- Modificar la definición de una fuente de datos específica o la definición de una categoría de fuentes de datos.
- Realizar cambios en la configuración de una fuente de datos específica o en la configuración de una categoría de fuentes de datos—cambios que persisten a través de múltiples conexiones a la base de datos federada.

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

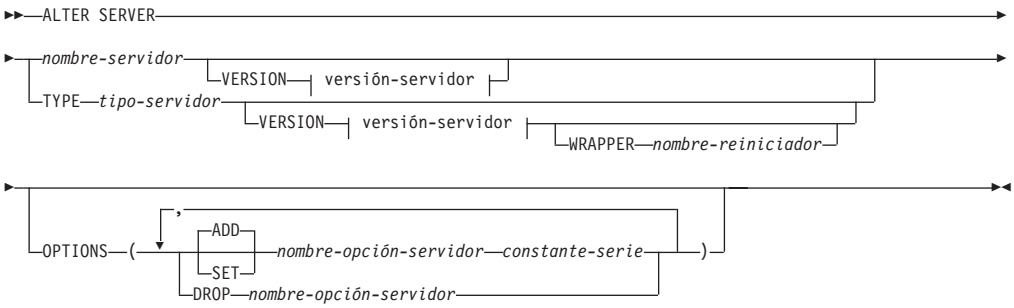
Invocación:

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

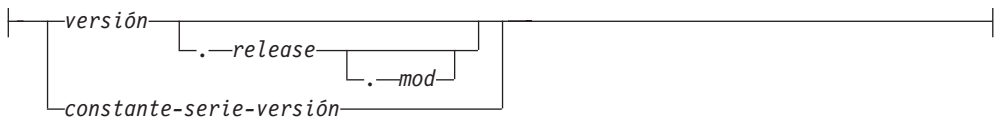
Autorización:

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

Sintaxis:



versión-servidor:



Descripción:

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

VERSION
Después de *nombre-servidor*, VERSION y su parámetro especifican una nueva versión de la fuente de datos que indica *nombre-servidor*.

versión
Especifica el número de versión. El valor debe ser un entero.

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

mod
Especifica el número de la modificación del release indicado por *release*. El valor debe ser un entero.

constante-serie-versión
Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i') o puede estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

TYPE *tipo-servidor*
Especifica el tipo de fuente de datos al que se deben aplicar los cambios que se están solicitando.

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

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

OPTIONS
Indica las opciones de servidor que se deben habilitar, restaurar o descartar para la fuente de datos indicada por *nombre-servidor* o para la categoría de fuentes de datos indicada por *tipo-servidor* y sus parámetros asociados.

ADD
Habilita una opción de servidor.

SET
Cambia el valor de una opción de servidor.

nombre-opción-servidor
Nombra una opción de servidor que se debe habilitar o restaurar.

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

DROP *nombre-opción-servidor*
Desactiva una opción de servidor.

Notas:

- Una opción de servidor no se puede especificar más de una vez en la misma sentencia ALTER SERVER (SQLSTATE 42853). Cuando se habilita, restaura o descarta una opción de servidor, no afecta a ninguna otra opción de servidor que se esté utilizando.
- Una sentencia ALTER SERVER de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:

  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- Si la opción de servidor se establece en un valor para un tipo de fuente de datos y en otro valor para una instancia de este tipo, el segundo valor altera temporalmente el primero para la instancia. Por ejemplo, supongamos que PLAN_HINTS se establece en 'Y' para el tipo de servidor ORACLE y en 'N' para una fuente de datos de Oracle denominado DELPHI. Esta configuración provoca que se habiliten las indicaciones de planes en todas las fuentes de datos Oracle excepto en DELPHI.
- Sólo se puede ejecutar ALTER SET o ALTER DROP en opciones de servidor de una categoría de fuentes de datos que se ha habilitado antes por una operación de opción de servidor ALTER ADD (SQLSTATE 42704).

Ejemplos:

Ejemplo 1: Asegúrese de que cuando se envían ID de autorización a las fuentes de datos Oracle 8.0.3, no se cambian las mayúsculas y minúsculas de los ID. Suponga también que la CPU del servidor federado local es el doble de rápida que la CPU de fuente de datos. Informe al optimizador de esta estadística.

```
ALTER SERVER
   TYPE ORACLE
   VERSION 8.0.3
   OPTIONS
     ( ADD FOLD_ID 'N',
     SET CPU_RATIO '2.0')
```

Ejemplo 2: Indique que la fuente de datos Documentum denominada DCTM_SVR_ASIA se ha cambiado a la Versión 4.

```
ALTER SERVER DCTM_SVR_ASIA
  VERSION 4
```

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*

Información relacionada:

- "Server options for federated systems" en la publicación *Federated Systems Guide*

ALTER TABLE

La sentencia ALTER TABLE modifica la definición de una tabla.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio ALTER para la tabla que se debe modificar
- Privilegio CONTROL para la tabla que se debe modificar
- Privilegio ALTERIN para el esquema de la tabla
- Autorización SYSADM o DBADM.

Para crear o eliminar una clave foránea, el ID de autorización de la sentencia debe tener uno de los privilegios siguientes para la tabla padre:

- Privilegio REFERENCES para la tabla
- Privilegio REFERENCES para todas las columnas de la clave padre especificada
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

Para eliminar una clave primaria o una restricción de unicidad de la tabla T, el ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes para cada tabla que sea dependiente de esta clave padre T:

- Privilegio ALTER para la tabla
- Privilegio CONTROL para la tabla
- Privilegio ALTERIN para el esquema de la tabla
- Autorización SYSADM o DBADM.

Para modificar una tabla con el fin de convertirla en una tabla de consultas materializadas (utilizando una selección completa), entre los privilegios del ID de autorización de la sentencia debe incluir, como mínimo, uno de los siguientes:

- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM;

y, como mínimo, uno de los siguientes para cada tabla o vista que se identifique en la selección completa:

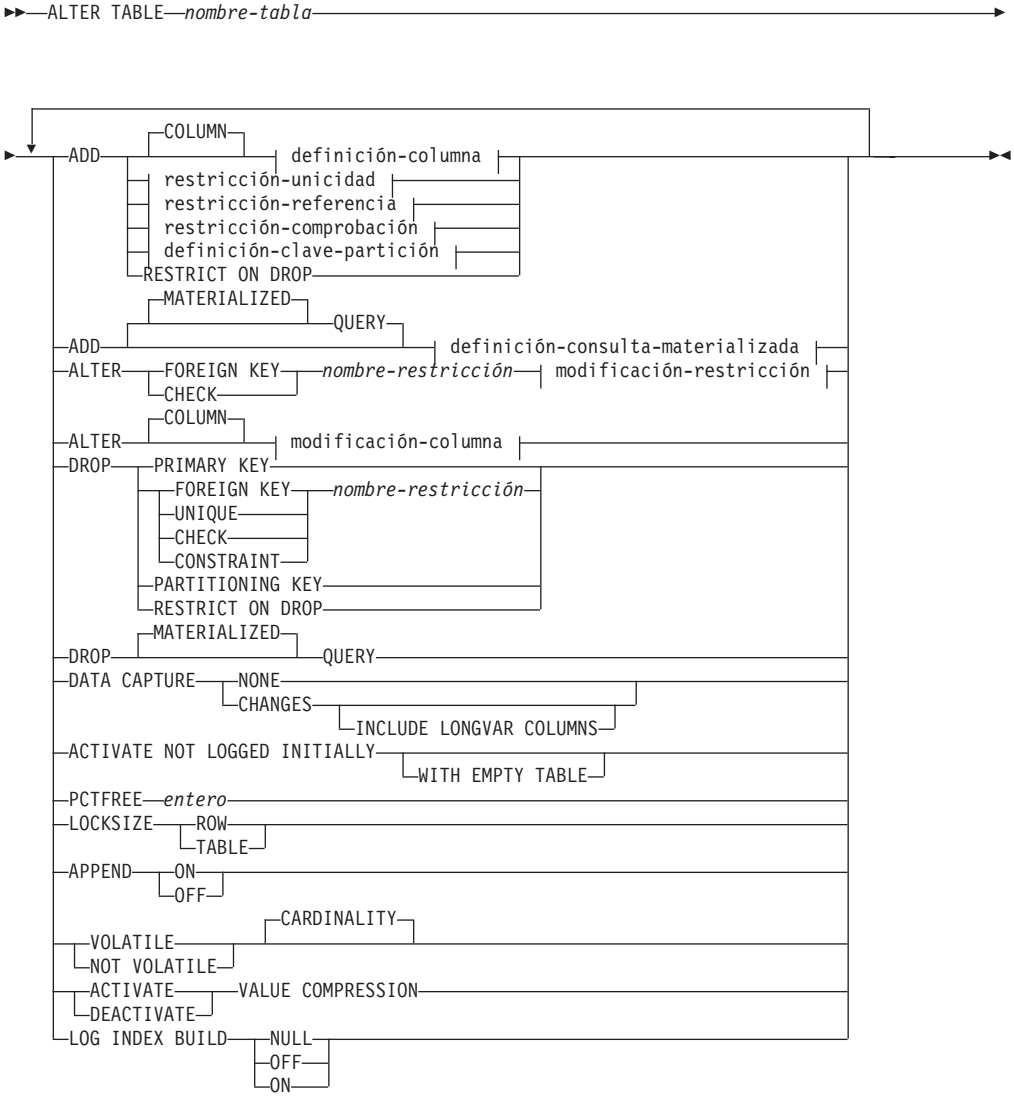
- Privilegio SELECT y ALTER para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Privilegio SELECT para la tabla o vista y privilegio ALTERIN para el esquema de la tabla o vista
- Autorización SYSADM o DBADM.

Para modificar una tabla con el fin de que ya no sea una tabla de consultas materializadas, entre los privilegios del ID de autorización de la sentencia debe

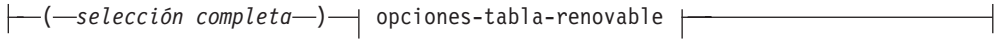
incluir, como mínimo, uno de los siguientes para cada tabla o vista que se identifique en la selección completa utilizada para definir la tabla de consultas materializadas:

- Privilegio ALTER para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Privilegio ALTERIN para el esquema de la tabla o vista
- Autorización SYSADM o DBADM

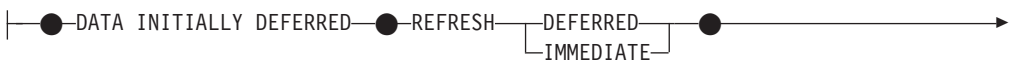
Sintaxis:



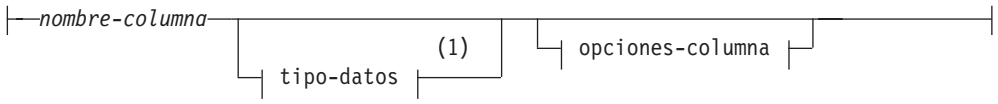
definición-consulta-materializada:



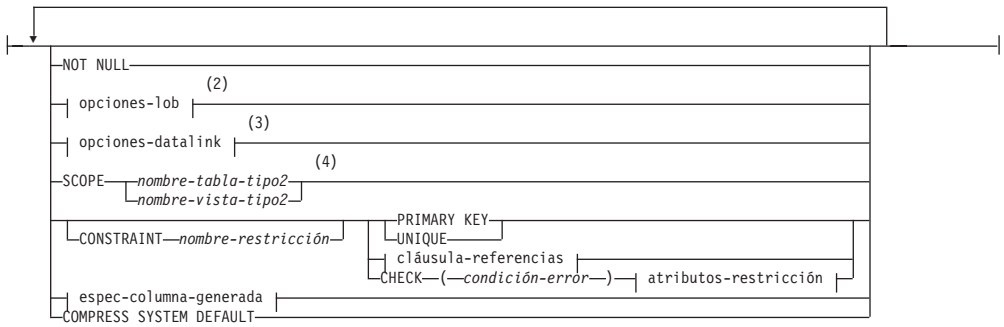
opciones-tabla-renovable:



definición-columna:



opciones-columna:



Notas:

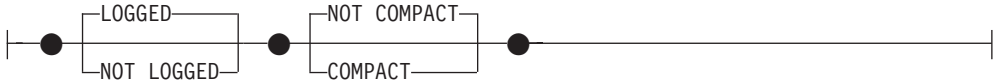
1 Si la primera opción de columna elegida es la especificación de columna generada, se puede omitir el tipo de datos y ser calculado por la expresión de generación.

2 La cláusula opciones-lob sólo se aplica a tipos de gran objeto (BLOB, CLOB y DBCLOB) y a los tipos diferenciados basados en tipos de gran objeto.

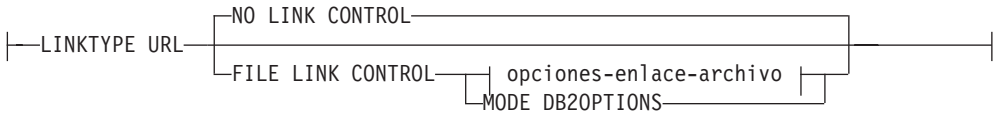
3 La cláusula opciones-datalink sólo se aplica al tipo DATALINK y a los tipos diferenciados basados en el tipo DATALINK.

4 La cláusula SCOPE sólo se aplica al tipo REF.

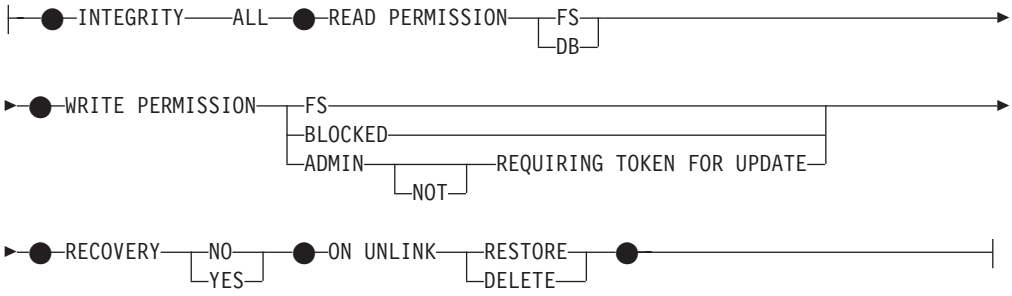
opciones-lob:



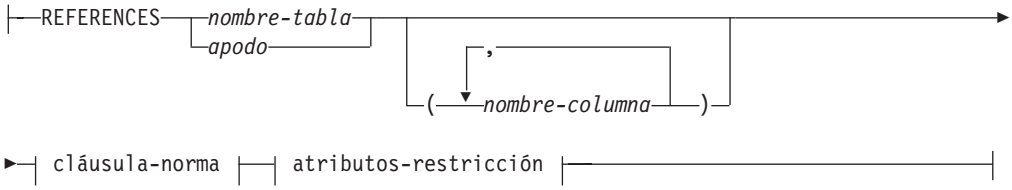
opciones-datalink:



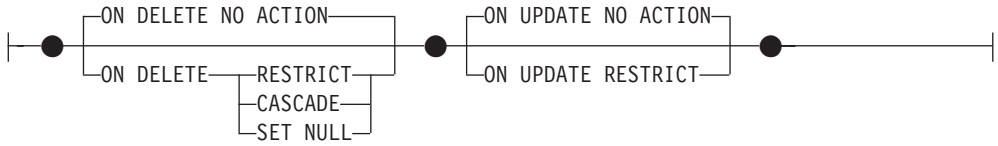
opciones-enlace-archivo:



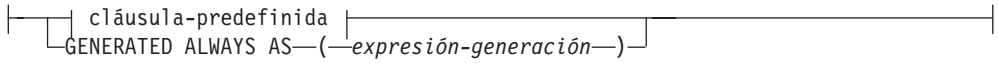
cláusula-referencias:



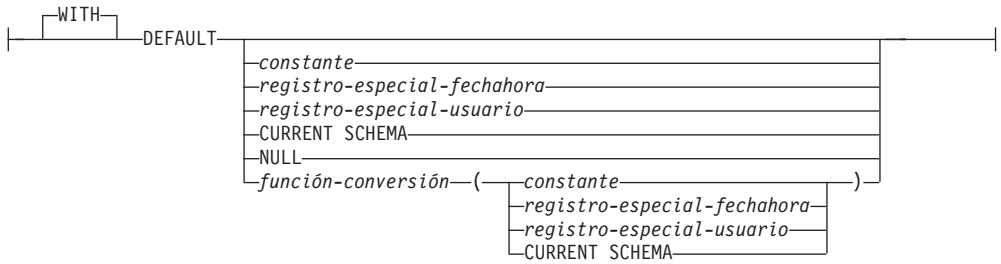
cláusula-norma:



espec-columna-generada:

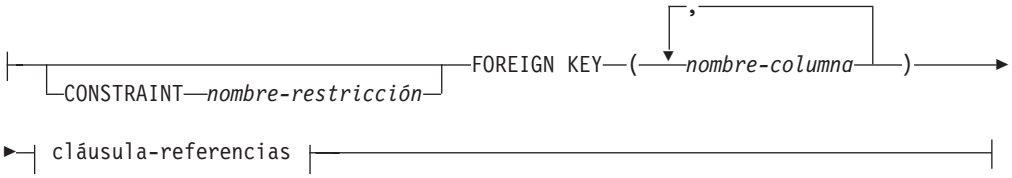


cláusula-predefinida:

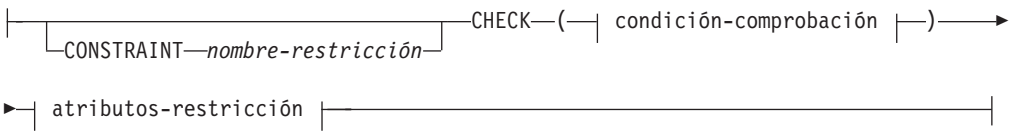


restricción-unicidad:

restricción-referencia:



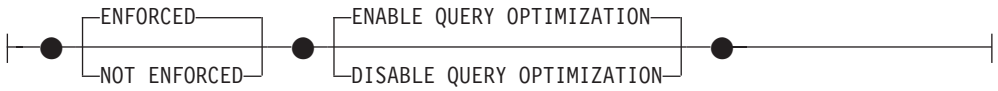
restricción-comprobación:



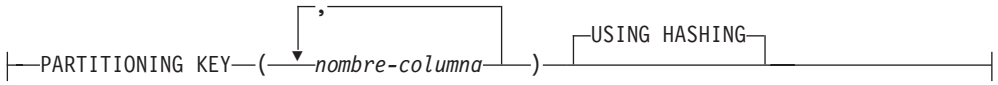
condición-comprobación:

dependencia-funcional:

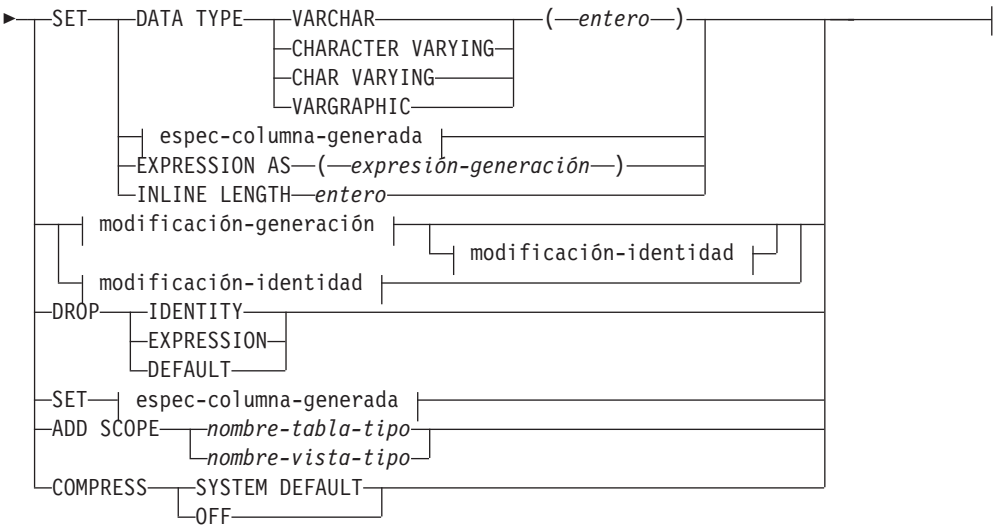
atributos-restricción:



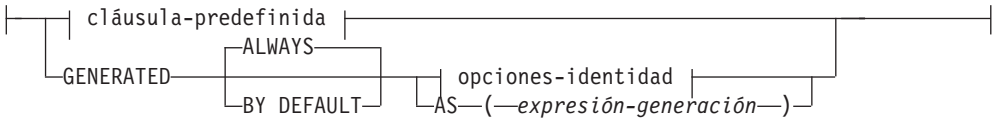
definición-clave-partición:



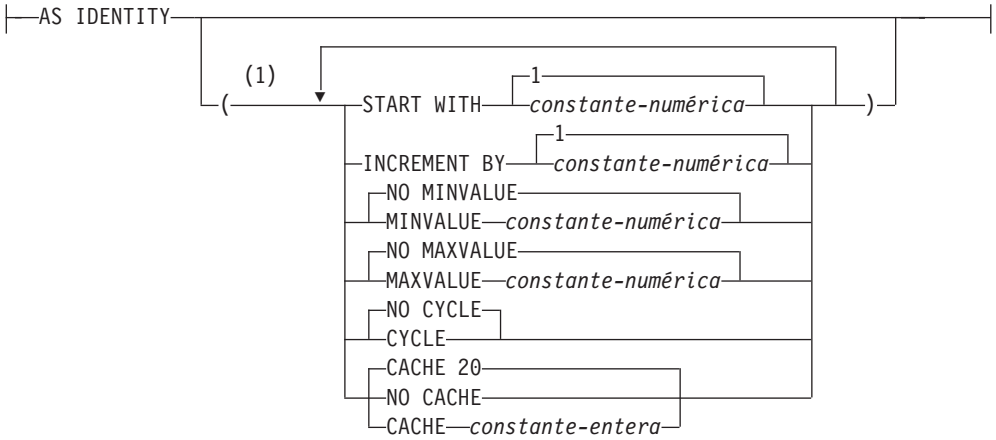
modificación-columna:



espec-columna-generada:

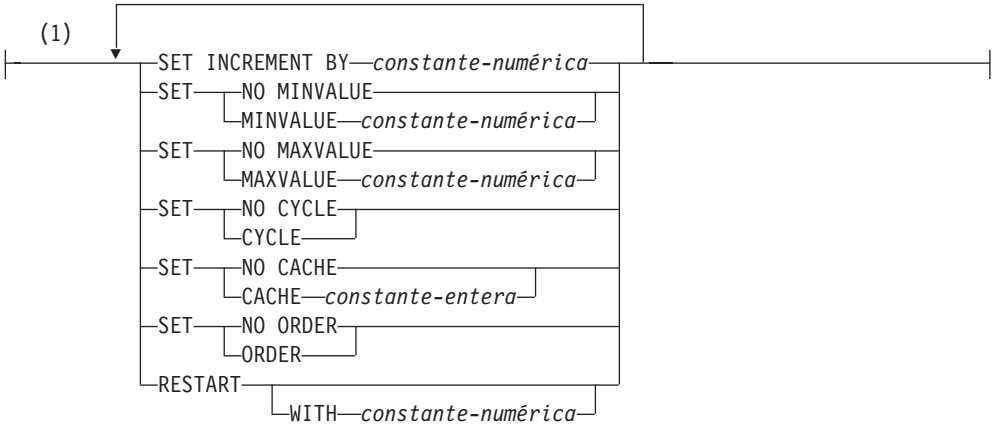


opciones-identidad:

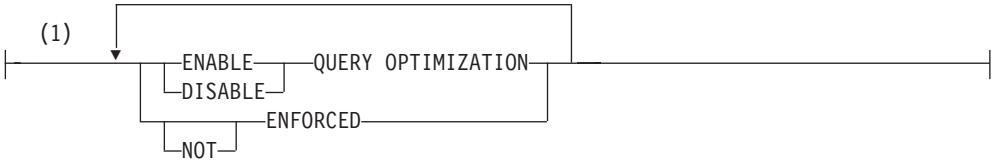


modificación-generación:

modificación-identidad:



modificación-restricción:



Notas:

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

Descripción:

nombre-tabla
El *nombre-tabla* debe identificar una tabla que exista en el servidor actual. No puede ser un apodo (SQLSTATE 42809) ni tampoco una vista, una tabla de catálogo ni una tabla temporal declarada (SQLSTATE 42995).

Si *nombre-tabla* identifica una tabla de consultas materializadas, las modificaciones se limitan a añadir o descartar la tabla de consultas materializadas, a activar NOT LOGGED INITIALLY, a añadir o descartar RESTRICT ON DROP y a cambiar PCTFREE, LOCKSIZE, APPEND o VOLATILE.

Si el *nombre-tabla* identifica una tabla de clúster de rango, las modificaciones se limitan a añadir, cambiar o descartar restricciones, a activar NOT LOGGED INITIALLY, a añadir o descartar RESTRICT ON DROP, a cambiar el tamaño de bloqueo, a capturar datos o volátil y a establecer los valores por omisión de columna.

ADD *definición-columna*
Añade una columna a la tabla. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH). Para todas las filas existentes en la tabla, el valor de la nueva columna se establece en su valor por omisión. La nueva columna es la última columna de la tabla; es decir, si inicialmente existen *n* columnas, la columna añadida será la columna *n*+1.

La adición de la nueva columna no debe dar lugar a que el número total de bytes de todas las columnas exceda el tamaño de registro máximo.

nombre-columna
Es el nombre de la columna que se va a añadir a la tabla. El nombre no

puede estar calificado. No pueden utilizarse nombres de columna existentes en la tabla (SQLSTATE 42711).

tipo-datos
Es uno de los tipos de datos que se listan en "CREATE TABLE".

NOT NULL
Evita que la columna contenga valores nulos. También debe especificarse la *cláusula-predefinida* (SQLSTATE 42601).

opciones-lob
Especifica opciones para los tipos de datos LOB. Consulte *opciones-lob* en "CREATE TABLE".

opciones-datalink
Especifica opciones para los tipos de datos DATALINK. Consulte *opciones-datalink* en "CREATE TABLE".

SCOPE
Especifica un ámbito para una columna de tipo de referencia.

nombre-tabla-tipo2
El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo2* (SQLSTATE 428DM). No se realiza ninguna comprobación del valor por omisión de *nombre-columna* para asegurarse de si realmente el valor hace referencia a una fila existente de *nombre-tabla-tipo2*.

nombre-vista-tipo2
El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo2* (SQLSTATE 428DM). No se realiza ninguna comprobación del valor por omisión de *nombre-columna* para asegurarse de si realmente el valor hace referencia a una fila existente de *nombre-vista-tipo2*.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar una restricción que ya se ha especificado dentro de la misma sentencia ALTER TABLE, o como el nombre de cualquier otra restricción existente en la tabla (SQLSTATE 42710).

Si el usuario no ha especificado el nombre de restricción, el sistema genera un identificador exclusivo de 18 caracteres dentro de los identificadores de las restricciones existentes definidas en la tabla. (El identificador se compone de la palabra ″SQL″ seguida de una secuencia de 15 caracteres numéricos que genera una función basada en la indicación de la hora.)

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* puede utilizarse como el nombre de un índice que se ha creado para dar soporte a la restricción. Consulte el apartado "Notas" en la página 73 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. Por lo tanto, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada. La columna no puede contener valores nulos, de manera que también debe especificarse NOT NULL (SQLSTATE 42831).

Consulte el apartado PRIMARY KEY en la descripción de la *restricción-unicidad* más abajo.

UNIQUE
Proporciona un método abreviado de definir una clave de unicidad compuesta de una sola columna. Por lo tanto, si se especifica UNIQUE KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula UNIQUE(C) como cláusula separada.

Consulte el apartado UNIQUE en la descripción de la *restricción-unicidad* más abajo.

cláusula-referencias
Proporciona un método abreviado de definir una clave externa compuesta de una sola columna. Así, si se especifica una cláusula-referencias en la definición de la columna C, el efecto es el mismo que si se especificase esa cláusula-referencias como parte de una cláusula FOREIGN KEY en la que C fuera la única columna identificada.

Consulte *cláusula-referencias* en "CREATE TABLE".

CHECK (*condición-error***)**
Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Consulte *condición-error* en "CREATE TABLE".

espec-columna-generada
Para ver detalles sobre la generación de columnas, consulte "CREATE TABLE".

cláusula-predefinida
Especifica un valor por omisión para la columna.

WITH
Palabra clave opcional.

DEFAULT
Proporciona un valor por omisión en el caso de que no se suministre ningún valor en INSERT o se especifique uno como DEFAULT en INSERT o UPDATE. Si no se especifica un valor por omisión específico a continuación de la palabra clave DEFAULT, el valor por omisión depende del tipo de datos de la columna, tal como se muestra en la Tabla 2. Si una columna está definida como DATALINK o tipo estructurado, no puede especificarse una cláusula por omisión.

Si se define una columna utilizando un tipo diferenciado, el valor por omisión de la columna es el valor por omisión del tipo de datos fuente convertido al tipo diferenciado.

Tabla 2. Valores por omisión (cuando no se especifica ningún valor)

Tipo de datos	Valor por omisión
Numérico	0
Serie de caracteres de longitud fija	Blancos
Serie de caracteres de longitud variable	Una serie de longitud 0
Serie gráfica de longitud fija	Blancos de doble byte
Serie gráfica de longitud variable	Una serie de longitud 0
Fecha	Para las filas existentes, fecha que corresponde al 1 de enero de 0001. Para las filas añadidas, la fecha actual.

Tabla 2. Valores por omisión (cuando no se especifica ningún valor) (continuación)

Tipo de datos	Valor por omisión
Hora	Para filas existentes, una hora que corresponde a las 0 horas, 0 minutos y 0 segundos. Para filas añadidas, la hora actual.
Indicación de la hora	Para las filas existentes, una fecha que corresponde al 1 de enero, 0001 y una hora que corresponde a las 0 horas, 0 minutos, 0 segundos y 0 microsegundos. Para filas añadidas, la indicación de la hora actual.
Serie binaria (blob)	Una serie de longitud 0

La omisión de DEFAULT en una *definición-columna* da como resultado la utilización del valor nulo como valor por omisión para la columna.

Los tipos específicos de los valores que pueden especificarse con la palabra clave DEFAULT son los siguientes.

constante
Especifica la constante como el valor por omisión para la columna. La constante especificada debe:

- representar un valor que pueda asignarse a la columna de acuerdo con las normas de asignación descritas en el Capítulo 3
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- no tener dígitos diferentes de cero fuera de la escala del tipo de datos de la columna si la constante es decimal (por ejemplo, 1,234 no puede ser el valor por omisión para una columna DECIMAL(5,2))
- estar expresada con un máximo de 254 caracteres, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificado y paréntesis, cuando la constante es el argumento de una *función-conversión*.

registro-especial-fechahora
Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser el tipo de datos que corresponde al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE). Para las filas existentes, el valor es la fecha, hora o indicación de la hora actuales en el momento de procesar la sentencia ALTER TABLE.

registro-especial-usuario
Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar

USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER. Para las filas existentes, el valor es CURRENT USER, SESSION_USER o SYSTEM_USER de la sentencia ALTER TABLE.

CURRENT SCHEMA
Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Para las filas existentes, se procesa el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse la sentencia ALTER TABLE.

NULL
Especifica NULL como valor por omisión para la columna. Si se ha especificado NO NULL, DEFAULT NULL no debe especificarse en la misma definición de columna.

función-conversión
Esta forma de valor por omisión sólo puede utilizarse con las columnas definidas como tipo de datos diferenciado, BLOB o de indicación de fecha y hora (DATE, TIME o TIMESTAMP). Para el tipo diferenciado, a excepción de los tipos diferenciados basados en tipos BLOB o de indicación de fecha y hora, el nombre de la función debe coincidir con el nombre del tipo diferenciado de la columna. Si está calificado con un nombre de esquema, debe ser el mismo que el nombre de esquema del tipo diferenciado. Si no está calificado, el nombre de esquema procedente de la resolución de la función debe ser el mismo que el nombre de esquema del tipo diferenciado. Para un tipo diferenciado basado en un tipo de indicación de fecha y hora, en el que el valor por omisión es una constante, debe utilizarse una función y el nombre de ésta debe coincidir con el nombre del tipo de fuente del tipo diferenciado, con un nombre de esquema implícito o explícito de SYSIBM. Para las demás columnas de indicación de fecha y hora, también puede utilizarse la correspondiente función de indicación de fecha y hora. Para un BLOB o tipo diferenciado basado en BLOB, debe utilizarse una función, y el nombre de la función debe ser BLOB, junto con SYSIBM como nombre de esquema implícito o explícito.

constante
Especifica una constante como argumento. La constante debe cumplir las normas de una constante para el tipo de fuente del tipo diferenciado, o para el tipo de datos si no es un tipo diferenciado. Si la función-conversión es BLOB, la constante debe ser de tipo serie.

registro-especial-fechahora
Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario
Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de fuente del tipo diferenciado de la

columna debe ser el tipo de datos serie con una longitud mínima de 8 bytes. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

CURRENT SCHEMA
Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la función-conversión es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

GENERATED
Especifica que DB2 generará los valores para la columna.

ALWAYS
Especifica que DB2 generará siempre un valor para la columna cuando se inserte una fila en la tabla o cuando cambie el valor del resultado de *expresión-generación*. El resultado de la expresión se almacena en la tabla. GENERATED ALWAYS es la opción recomendada, a menos que estén realizándose operaciones de propagación o de descarga y de recarga. GENERATED ALWAYS es la opción obligatoria para las columnas generadas.

BY DEFAULT
Especifica que DB2 generará un valor para la columna cuando se inserte una fila en la tabla, o cuando se actualice especificando DEFAULT para la columna, a menos que se especifique un valor explícito. BY DEFAULT es la opción recomendada cuando se utiliza la propagación de datos o cuando se realizan operaciones de descarga y recarga.

opciones-identidad
No se puede especificar esta cláusula cuando se añade una columna a una tabla existente.

AS (*expresión-generación***)**
Especifica que la definición de la columna se basa en una expresión. Es necesario poner la tabla en estado pendiente de comprobación, utilizando la sentencia SET INTEGRITY. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY con FORCE GENERATED para actualizar y comprobar todos los valores de esa columna con la nueva expresión. Para ver los detalles acerca de la especificación de una columna con una *expresión-generación*, consulte "CREATE TABLE".

COMPRESS SYSTEM DEFAULT
Especifica que los valores por omisión del sistema (es decir, los valores por omisión que se utilizan para los tipos de datos cuando no se ha especificado ningún valor específico) van a almacenarse utilizando el espacio mínimo. Si no se especifica la cláusula VALUE COMPRESSION, se devuelve un mensaje de aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el espacio mínimo.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

El tipo de datos base no debe ser DATE, TIME ni TIMESTAMP (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto. Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

ADD *restricción-unicidad*
Define una restricción de clave primaria o de unicidad. No puede añadirse una restricción de unicidad o de clave primaria a una tabla que sea una subtabla (SQLSTATE 429B3). Si la tabla es la supertabla del principio de la jerarquía, la restricción se aplica a la tabla y a todas sus subtablas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de clave primaria o de unicidad. Para obtener más información, consulte *nombre-restricción* en "CREATE TABLE".

UNIQUE (*nombre-columna***...,)**
Define una clave de unicidad compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez. El nombre no puede estar calificado. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024. No puede utilizarse ningún tipo LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado como parte de una clave exclusiva, incluso si el atributo de longitud de la columna es lo suficientemente pequeño como para caber dentro del límite de 1024 bytes (SQLSTATE 54008). El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.) Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515).

Se realiza una comprobación para determinar si un índice existente coincide con la definición de clave de unicidad (ignorando cualquier columna INCLUDE del índice). Una definición de índice coincide si identifica el mismo conjunto de columnas sin tener en cuenta el orden de las columnas ni las especificaciones de dirección (ASC/DESC). Si se encuentra una definición de índice coincidente, la descripción del índice se cambia para indicar que el sistema lo necesita y se cambia por de unicidad (después de asegurar la exclusividad) si no era un índice de unicidad. Si la tabla tiene más de un índice coincidente, se selecciona un índice de unicidad existente (la selección es arbitraria). Si no se encuentra ningún índice coincidente, se creará automáticamente un índice de unicidad para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado "Notas" en la página 73 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

CLAVE PRIMARIA *...(nombre-columna,)*
Define una clave primaria formada por las columnas identificadas. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez. El nombre no puede estar calificado. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024. La tabla no debe tener una clave primaria y las columnas identificadas deben definirse como NOT NULL. No puede utilizarse ningún tipo LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipo diferenciado basado

en cualquiera de estos tipos o tipo estructurado como parte de una clave primaria, incluso si el atributo de longitud de la columna es lo suficientemente pequeño como para caber dentro del límite de 1024 bytes (SQLSTATE 54008). El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.) Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515).

Se realiza una comprobación para determinar si un índice existente coincide con la definición de clave primaria (ignorando cualquier columna INCLUDE del índice). Una definición de índice coincide si identifica el mismo conjunto de columnas sin tener en cuenta el orden de las columnas ni las especificaciones de dirección (ASC/DESC). Si se encuentra una definición de índice coincidente, la descripción del índice se cambia para que indique que es el índice principal, tal como necesita el sistema, y se cambia al de unicidad (después de asegurar la exclusividad) si no era un índice de unicidad. Si la tabla tiene más de un índice coincidente, se selecciona un índice de unicidad existente (la selección es arbitraria). Si no se encuentra ningún índice coincidente, se creará automáticamente un índice de unicidad para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado "Notas" en la página 73 para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

En una tabla sólo se puede definir una clave primaria.

ADD *restricción-referencia*
Define una restricción de referencia. Consulte *restricción-referencia* en "CREATE TABLE".

ADD *restricción-comprobación*
Define una restricción de comprobación o una dependencia funcional. Consulte *restricción-comprobación* en "CREATE TABLE".

ADD *definición-clave-partición*
Define una clave de particionamiento. La tabla debe estar definida en un espacio de tablas de un grupo de particiones de base de datos de una única partición y no debe tener todavía una clave de particionamiento. Si ya existe una clave de particionamiento para la tabla, se debe suprimir la clave existente antes de añadir la nueva clave de particionamiento.

No puede añadirse una clave de particionamiento a una tabla que sea una subtabla (SQLSTATE 428DH).

PARTITIONING KEY (*nombre-columna***...)**
Define una clave de particionamiento utilizando las columnas especificadas. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez. El nombre no puede estar calificado. Una columna no se puede utilizar como parte de una clave de particionamiento si el tipo de datos de la columna es un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado de cualquiera de estos tipos o un tipo estructurado.

USING HASHING
Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

ADD RESTRICT ON DROP
Especifica que la tabla no puede eliminarse y que el espacio de tablas que contiene la tabla no puede eliminarse.

ADD MATERIALIZED QUERY

definición-consulta-materializada
Cambia una tabla normal por una tabla de consultas materializadas para utilizarla durante la optimización de la consulta. La tabla especificada por *nombre-tabla*:

- No debe haberse definido anteriormente como una tabla de consultas materializadas
- No debe ser una tabla con tipo
- No debe tener ninguna restricción, índice de unicidad ni activador definidos
- No debe existir una referencia a ésta en la definición de otra tabla de consultas materializadas.

Si *nombre-tabla* no cumple estos criterios, se devolverá un error (SQLSTATE 428EW).

selección completa
Define la consulta en la que se basa la tabla. Las columnas de la tabla existente:

- deben tener el mismo número de columnas
- deben tener exactamente los mismos tipos de datos
- deben tener los mismos nombres de columna en las mismas posiciones de orden

que las columnas resultantes de la *selección completa* (SQLSTATE 428EW). Para obtener información detallada acerca de la especificación de la *selección completa* para un tabla de consultas materializadas, consulte "CREATE TABLE". Una restricción adicional es que *nombre-tabla* no puede estar referenciada, directa o indirectamente, en la selección completa.

opciones-tabla-renovable
Especifica las opciones que pueden renovarse para modificar una tabla de consultas materializadas.

DATA INITIALLY DEFERRED
Los datos de la tabla deben validarse utilizando la sentencia REFRESH TABLE o SET INTEGRITY.

REFRESH
Indica cómo se mantienen los datos de la tabla.

DEFERRED
Los datos de la tabla pueden renovarse en cualquier momento mediante la sentencia REFRESH TABLE. Los datos de la tabla sólo reflejan el resultado de la consulta en forma de instantánea en el momento en que se procesa la sentencia REFRESH TABLE. Las tablas de consultas materializadas que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

IMMEDIATE
Los cambios que se han realizado en las tablas subyacentes como parte de una sentencia DELETE, INSERT o UPDATE se

aplican en cascada a la tabla de consultas materializadas. En este caso, el contenido de la tabla, en cualquier momento, es igual como si se procesara la subselección especificada. Las tablas de consultas materializadas que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION
Las tablas de consultas materializadas pueden utilizarse para la optimización de la consulta.

DISABLE QUERY OPTIMIZATION
La tabla de consultas materializadas no se utilizará para la optimización de la consulta. La tabla todavía puede consultarse directamente.

MAINTAINED BY
Especifica quién mantiene los datos de la tabla de consultas materializadas: el sistema, el usuario o una herramienta de duplicación.

SYSTEM
Especifica que el sistema mantiene los datos de la tabla de consultas materializadas.

USER
Especifica que el usuario mantiene los datos de la tabla de consultas materializadas. El usuario puede realizar operaciones de actualización, supresión o inserción en las tablas de consultas materializadas mantenidas por el usuario. La sentencia REFRESH TABLE, que se utiliza para las tablas de consultas materializadas mantenidas por el sistema, no puede invocarse para las tablas de consultas materializadas mantenidas por el usuario. Sólo una tabla de consultas materializadas REFRESH DEFERRED puede definirse como MAINTAINED BY USER.

FEDERATED_TOOL
Especifica que la herramienta de duplicación mantiene los datos de la tabla de consultas materializadas. La sentencia REFRESH TABLE, que se utiliza para las tablas de consultas materializadas mantenidas por el sistema, no se puede invocar para las tablas de consultas materializadas mantenidas por herramientas federadas. Sólo se puede definir una única tabla de consultas materializadas REFRESH DEFERRED como MAINTAINED BY FEDERATED_TOOL.

ALTER FOREIGN KEY *nombre-restricción*
Altera los atributos de restricción de la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia existente (SQLSTATE 42704).

ALTER CHECK *nombre-restricción*
Modifica los atributos de la restricción de comprobación o de la dependencia funcional *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente o una dependencia funcional (SQLSTATE 42704).

modificación-restricción
Opciones para cambiar los atributos que se asocian a las restricciones de referencia o de comprobación.

ENFORCED o **NOT ENFORCED**
Especifica si el gestor de bases de datos obliga a aplicar la restricción durante operaciones normales como, por ejemplo, insertar, actualizar o suprimir.

ENFORCED
Cambie la restricción por ENFORCED. No se puede especificar ENFORCED para una dependencia funcional (SQLSTATE 42621).

NOT ENFORCED
Cambie la restricción por NOT ENFORCED. Sólo debe especificarse si, independientemente, se sabe que los datos de tabla se ajustan a la restricción.

ENABLE QUERY OPTIMIZATION o **DISABLE QUERY OPTIMIZATION**
Especifica si se puede utilizar la restricción o la dependencia funcional para la optimización de la consulta bajo circunstancias adecuadas.

ENABLE QUERY OPTIMIZATION
Se supone que la restricción es verdadera y se puede utilizar para la optimización de la consulta.

DISABLE QUERY OPTIMIZATION
La restricción no puede utilizarse para la optimización de la consulta.

ALTER *modificación-columna*
Modifica la definición de una columna. Sólo se modificarán los atributos especificados; los demás permanecerán sin modificar.

nombre-columna
Especifica el nombre de la columna que se debe modificar. El *nombre-columna* debe identificar una columna existente de la tabla (SQLSTATE 42703). El nombre no puede estar calificado. El nombre no debe identificar una columna que se añade, descarta o modifica en la misma sentencia ALTER TABLE (SQLSTATE 42711).

SET DATA TYPE
Define el tipo de datos de una columna. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH).

La modificación de la columna no debe dar lugar a que el número total de bytes de todas las columnas exceda el tamaño de registro máximo (SQLSTATE 54010). Si la columna se utiliza en una restricción de unicidad o en un índice, la nueva longitud no debe dar lugar a que la suma de las longitudes almacenadas de la restricción de unicidad o del índice excedan de 1024 (SQLSTATE 54008).

VARCHAR *entero*
Aumenta la longitud de una columna VARCHAR existente. CHARACTER VARYING o CHAR VARYING se pueden utilizar como sinónimos de la palabra clave VARCHAR. El tipo de datos de *nombre-columna* debe ser VARCHAR y la longitud máxima actual definida para la columna no debe ser mayor que el valor de *entero* (SQLSTATE 42837). El valor de *entero* es, como máximo, de 32 672.

VARGRAPHIC *entero*
Aumenta la longitud de una columna VARGRAPHIC existente. El tipo de datos de *nombre-columna* debe ser VARGRAPHIC y la longitud máxima actual definida para la columna no debe ser mayor que el valor de *entero* (SQLSTATE 42837). El valor de *entero* es, como máximo, de 16 336.

SET EXPRESSION AS (*expresión-generación***)**
Cambia la expresión de la columna por la *expresión-generación* especificada. SET EXPRESSION AS necesita que la tabla es establezca en estado pendiente de comprobación, utilizando la sentencia SET INTEGRITY. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY para actualizar y comparar todos los valores de la columna con la nueva expresión. La columna ya deberá haberse definido como columna generada basada en una expresión (SQLSTATE 42837) y no deberá aparecer en la cláusula DIMENSIONS de la tabla (SQLSTATE 42997). La expresión de generación debe seguir las mismas normas que se aplican al definir una columna generada. El tipo de datos resultante de la expresión de generación se debe poder asignar al tipo de datos de la columna (SQLSTATE 42821).

SET *espec-columna-generada*
Especifica la técnica utilizada para generar un valor para la columna. Puede adoptar la forma de un valor por omisión específico, una expresión o la definición de la columna como una columna de identidad. Si un valor por omisión de la columna es el resultado de una técnica de generación diferente, se debe descartar ese valor por omisión, lo que puede efectuarse en la misma *modificación-columna* utilizando una de las cláusulas DROP.

cláusula-predefinida
Especifica un nuevo valor por omisión para la columna que se va a modificar. La columna no debe estar definida ya como columna de identidad ni tener definida una expresión de generación (SQLSTATE 42837). El valor por omisión especificado debe representar un valor que pueda asignarse a la columna según las normas de asignación descritas en "Asignaciones y comparaciones". La modificación del valor por omisión no cambia el valor asociado a esta columna para las filas existentes.

GENERATED ALWAYS o **GENERATED BY DEFAULT**
Especifica cuándo el gestor de bases de datos debe generar valores para la columna. GENERATED BY DEFAULT especifica que sólo se debe generar un valor cuando no se facilita uno o cuando se utiliza la palabra clave DEFAULT en una asignación para la columna. GENERATED ALWAYS especifica que el gestor de bases de datos debe generar siempre un valor para la columna. No se puede especificar GENERATED BY DEFAULT con una *expresión-generación*.

opciones-identidad
Especifica que se trata de la columna de identidad para la tabla. La columna no debe estar definida ya como columna de identidad, no puede contener una expresión de generación ni un valor por omisión explícito (SQLSTATE 42837). Una tabla sólo puede contener una única columna de identidad (SQLSTATE 428C1). La columna se debe especificar como no anulable (SQLSTATE 42997) y el tipo de datos asociado a la columna debe ser numérico exacto con una escala de cero (SQLSTATE 42815). El tipo de datos numérico exacto puede ser: SMALLINT, INTEGER, BIGINT, DECIMAL o NUMERIC con una escala de cero, o un tipo diferenciado basado en uno de estos tipos. Para ver detalles sobre las opciones de identidad, consulte "CREATE TABLE".

AS (*expresión-generación***)**
Especifica que la definición de la columna se basa en una expresión. La columna no debe estar definida ya con una expresión

de generación, no puede ser la columna de identidad ni tampoco contener un valor por omisión explícito (SQLSTATE 42837). La *expresión-generación* debe cumplir las mismas normas que se aplican al definir una columna generada. Debe ser posible asignar el tipo de datos resultante de la *expresión-generación* al tipo de datos de la columna (SQLSTATE 42821). No se debe hacer referencia a la columna en la columna de clave de particionamiento ni en la cláusula ORGANIZE BY (SQLSTATE 42997).

SET INLINE LENGTH *entero*
Cambia la longitud en línea de una columna de tipo estructurado existente. La longitud en línea indica el tamaño máximo en bytes de una instancia de un tipo estructurado que debe almacenarse en línea con el resto de los valores de la fila. Las instancias de tipos estructurados que no se pueden almacenar en serie se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB.

El tipo de datos de *nombre-columna* debe ser un tipo estructurado (SQLSTATE 42842).

La longitud en línea por omisión para una columna de tipo estructurado es la longitud en línea de su tipo (que se especifica explícitamente o por omisión en la sentencia CREATE TYPE). Si la longitud en línea de un tipo estructurado es menor que 292, el valor 292 se utiliza para la longitud en línea de la columna.

El valor explícito de la longitud en línea sólo puede incrementarse (SQLSTATE -1); el valor debe ser, como mínimo, 292; y no puede exceder de 32672 (SQLSTATE 54010).

La modificación de la columna no debe dar lugar a que el número total de bytes de todas las columnas exceda el tamaño de registro máximo (SQLSTATE 54010).

Esta sentencia no establecerá en línea los datos que ya se habían almacenado por separado del resto de la fila. Para beneficiarse de la longitud en línea alterada de una columna de tipo estructurado, invoque el mandato REORG y ejecútelo para la tabla especificada tras haber alterado la longitud en línea de su columna.

SET GENERATED ALWAYS o **GENERATED BY DEFAULT**
Especifica cuándo el gestor de bases de datos debe generar valores para la columna. GENERATED BY DEFAULT especifica que sólo se debe generar un valor cuando no se facilita uno o cuando se utiliza la palabra clave DEFAULT en una asignación para la columna. GENERATED ALWAYS especifica que el gestor de bases de datos debe generar siempre un valor para la columna. La columna ya debe estar definida como columna generada basada en una columna de identidad; es decir, debe estar definida con la cláusula AS IDENTITY (SQLSTATE 42837).

DROP DEFAULT
Descarta el valor por omisión actual para la columna. La columna especificada debe tener un valor por omisión (SQLSTATE 42837).

DROP EXPRESSION
Descarta los atributos de expresión generada de la columna, lo que la convierte en una columna no generada. No se permite utilizar DROP EXPRESSION si no se trata de una columna de expresión generada (SQLSTATE 42837).

DROP IDENTITY
Descarta los atributos de identidad de la columna, lo que la convierte en una columna de tipo de datos numéricos simple. No se permite utilizar DROP IDENTITY si no se trata de una columna de identidad (SQLSTATE 42837).

ADD SCOPE
Añade un ámbito a una columna de tipo de referencia existente que todavía no tiene un ámbito definido (SQLSTATE 428DK). Si la tabla que se modifica es una tabla con tipo, la columna no debe ser herencia de una supertabla (SQLSTATE 428DJ).

nombre-tabla-tipo
El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en el *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en *nombre-vista-tipo*.

COMPRESS
Especifica si los valores por omisión de esta columna van a almacenarse o no de forma más eficaz.

SYSTEM DEFAULT
Especifica que los valores por omisión del sistema (es decir, los valores por omisión que se utilizan para los tipos de datos cuando no se ha especificado ningún valor específico) van a almacenarse utilizando el espacio mínimo. Si la tabla todavía no se ha establecido con el atributo VALUE COMPRESSION activado, se devuelve un mensaje de aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el espacio mínimo.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

Los datos que existen en la columna no cambian. Considere la reorganización de la tabla fuera de línea para que los datos existentes puedan beneficiarse del almacenamiento de los valores por omisión del sistema en el que se utiliza el espacio mínimo.

OFF
Especifica que los valores por omisión del sistema van a almacenarse en la columna como valores normales. Los datos que existen en la columna no cambian. Para cambiar los datos existentes, se recomienda la reorganización fuera de línea.

El tipo de datos base no debe ser DATE, TIME ni TIMESTAMP (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto. Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

Si la tabla que se modifica es una tabla con tipo, la columna no debe ser herencia de una supertabla (SQLSTATE 428DJ).

modificación-identidad
Modifica los atributos de identidad de la columna. Debe ser una columna de identidad.

SET INCREMENT BY *constante-numérica*
Especifica el intervalo existente entre valores consecutivos de la columna de identidad. El siguiente valor que se deberá generar para la columna de identidad se determinará a partir del último valor asignado con el incremento aplicado. La columna debe estar ya definida con el atributo IDENTITY (SQLSTATE 42837).

Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820) y sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente tras la sentencia ALTER. Si este valor es 0 o positivo, es una secuencia ascendente tras la sentencia ALTER.

SET NO MINVALUE o **MINVALUE** *constante-numérica*
Especifica el valor mínimo en el que una columna de identidad descendente pasa por un ciclo o deja de generar valores o bien el valor en el que una columna de identidad ascendente pasa por un ciclo después de alcanzar el valor máximo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

NO MINVALUE
Para una secuencia ascendente, el valor es el valor de inicio original. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos de la columna.

MINVALUE *constante-numérica*
Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor que o igual al valor máximo (SQLSTATE 42815).

SET NO MAXVALUE o **MAXVALUE** *constante-numérica*
Especifica el valor máximo en el que una columna de identidad ascendente pasa por un ciclo o deja de generar valores o bien el valor en el que una columna de identidad descendente pasa por un ciclo después de alcanzar el valor mínimo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

NO MAXVALUE
Para una secuencia ascendente, el valor es el valor máximo del tipo de datos de la columna. Para una secuencia descendente, el valor es el valor de inicio original.

MAXVALUE *constante-numérica*
Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la

derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser mayor que o igual al valor mínimo (SQLSTATE 42815).

SET NO CYCLE o **CYCLE**
Especifica si esta columna de identidad debe o no seguir generando valores tras la generación de su valor máximo o de su valor mínimo. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

NO CYCLE
Especifica que no se generarán valores para la columna de identidad una vez que se haya alcanzado el valor máximo o el valor mínimo.

CYCLE
Especifica que se continúen generando valores para esta columna después de haber alcanzado el valor máximo o el valor mínimo. Si se utiliza esta opción, cuando una columna de identidad ascendente ha alcanzado el valor máximo, genera el valor mínimo; o cuando una columna de secuencia descendente ha alcanzado el valor mínimo, genera el valor máximo. Los valores máximo y mínimo para la columna de identidad determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, pueden generarse valores duplicados para una columna de identidad. Aunque no es necesario, si se desean valores exclusivos, se puede definir un índice de unicidad de una sola columna utilizando la columna de identidad para asegurar la unicidad. Si existe un índice exclusivo en una columna de identidad de este tipo y se genera un valor que no es exclusivo, se produce un error (SQLSTATE 23505).

SET NO CACHE o **CACHE** *constante-entero*
Especifica si deben mantenerse en la memoria algunos valores preasignados, para conseguir un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste. La columna debe estar ya definida con el atributo IDENTITY (SQLSTATE 42837).

NO CACHE
Especifica que no se deben preasignar los valores de la columna de identidad. En un entorno de compartimiento de datos, si los valores de identidad *deben* generarse en el orden en que se solicitan, debe utilizarse la opción NO CACHE.

Si se especifica esta opción, los valores de la columna de identidad no se colocan en la memoria intermedia. En este caso, cada petición de un valor de identidad nuevo produce E/S síncrona en las anotaciones cronológicas.

CACHE *constante-entera*
Especifica cuántos valores de la secuencia de identidad deben asignarse previamente y conservarse en la memoria. Cuando se generan valores para la columna de identidad, la asignación previa y el almacenamiento de los valores en la antememoria reducen la E/S síncrona en el archivo de anotaciones cronológicas.

Si se necesita un nuevo valor para la columna de identidad y no existen valores no utilizados disponibles en la antememoria, la asignación del valor implica tener que esperar a que se produzca la

E/S en el archivo de anotaciones cronológicas. Sin embargo, cuando se necesita un valor nuevo para la columna de identidad y existe un valor no utilizado en la antememoria, la asignación de dicho valor de identidad puede suceder más rápidamente evitando la E/S en las anotaciones cronológicas.

En caso de que se produzca una desactivación de la base de datos, realizada con normalidad o como consecuencia de una anomalía en el sistema, todos los valores de secuencia que se han colocado en la antememoria y que no se han utilizado en las sentencias confirmadas se pierden (es decir, nunca se utilizarán). El valor especificado para la opción CACHE es el número máximo de valores para la columna de identidad que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42815).

SET NO ORDER o **ORDER**
Especifica si los valores de la columna de identidad deben generarse según el orden de petición. La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837).

NO ORDER
Especifica que los valores de la columna de identidad no necesitan generarse según el orden de petición.

ORDER
Especifica que los valores de la columna de identidad deben generarse en el orden en que se solicitan.

RESTART o **RESTART WITH** *constante-numérica*
Restablece el estado de la secuencia asociada con la columna de identidad. Si no se ha especificado WITH *constante-numérica*, la secuencia de la columna de identidad se reiniciará en el valor que se especificó, de forma implícita o explícita, como valor inicial para la columna de identidad durante su creación original.

La columna debe existir en la tabla especificada (SQLSTATE 42703) y ya tiene que estar definida con el atributo IDENTITY (SQLSTATE 42837). RESTART *no* cambia el valor START WITH original.

La *constante-numérica* es una constante numérica exacta que puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). La *constante-numérica* se utilizará como el siguiente valor para la columna.

DROP PRIMARY KEY
Elimina la definición de la clave primaria y todas las restricciones de referencia dependientes de esta clave primaria. La tabla debe tener una clave primaria.

DROP FOREIGN KEY *nombre-restricción*
Elimina la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia. Para obtener información sobre las implicaciones de eliminar una restricción de referencia, consulte el apartado “Notas” en la página 73.

DROP UNIQUE *nombre-restricción*
Elimina la definición de la restricción de unicidad *nombre-restricción* y todas las restricciones de referencia que dependen de esta restricción de unicidad. El

nombre-restricción debe identificar una restricción UNIQUE existente. Para obtener información sobre las consecuencias de eliminar una restricción de unicidad, vea "Notas" en la página 73.

DROP CHECK *nombre-restricción*
Elimina la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación que esté definida en la tabla.

DROP CONSTRAINT *nombre-restricción*
Elimina la restricción *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente, restricción de referencia, clave primaria o restricción de unicidad definida en la tabla. Para obtener información sobre las consecuencias de eliminar una restricción, vea "Notas" en la página 73.

DROP PARTITIONING KEY
Elimina la clave de particionamiento. La tabla debe tener una clave de particionamiento y debe estar en un espacio de tablas definido en un grupo de particiones de base de datos de una única partición.

DROP RESTRICT ON DROP
Elimina la restricción de eliminación de la tabla y el espacio de tablas que contiene la tabla.

DROP MATERIALIZED QUERY
Se utiliza para cambiar una tabla de consultas materializadas con el fin de que deje de considerarse una tabla de consultas materializadas. La tabla que *nombre-tabla* especifica debe definirse como una tabla de consultas materializadas que no está duplicada (SQLSTATE 428EW). La definición de las columnas de *nombre-tabla* no se modifica, pero la tabla ya no puede utilizarse para la optimización de consultas y ya no puede utilizarse la sentencia REFRESH TABLE.

DATA CAPTURE
Indica si se debe grabar en el registro cronológico información adicional para la duplicación de datos.

Si la tabla es una tabla con tipo, entonces esta opción no está soportadas (SQLSTATE 428DH para tablas raíz o 428DR para otras subtablas).

NONE
Indica que no se va a anotar ninguna información adicional.

CHANGES
Indica que en el archivo de anotaciones se escribirá información adicional referente a los cambios de SQL efectuados en esta tabla. Esta opción es necesaria para duplicar la tabla y se utiliza el programa Capture para capturar los cambios destinados a esta tabla y registrados en el archivo de anotaciones.

Si la tabla se ha definido para admitir datos en una partición distinta de la partición de catálogo (grupo de particiones de base de datos de varias particiones o grupo de particiones de base de datos con una partición distinta de la partición de catálogo), esta opción no recibe soporte (SQLSTATE 42997).

Si el nombre de esquema (implícito o explícito) de la tabla tiene más de 18 bytes, esta opción no recibe soporte (SQLSTATE 42997).

INCLUDE LONGVAR COLUMNS
Permite que los programas de utilidad de duplicación de datos capten los cambios realizados en las columnas LONG VARCHAR o LONG

VARGRAPHIC. La cláusula se puede especificar para las tablas que no tengan columnas LONG VARCHAR ni LONG VARGRAPHIC puesto que es posible MODIFICAR la tabla para que incluya estas columnas.

ACTIVATE NOT LOGGED INITIALLY
Activa el atributo NOT LOGGED INITIALLY de la tabla para esta unidad de trabajo actual.

Cualquier cambio realizado en la tabla mediante INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX o ALTER TABLE en la misma unidad de trabajo después de que esta sentencia haya alterado la tabla no se anota cronológicamente. Se anotan cronológicamente los cambios del catálogo del sistema realizados por una sentencia ALTER en la que esté activado el atributo NOT LOGGED INITIALLY. Se anotan cronológicamente los subsiguientes cambios realizados en la misma unidad de trabajo en la información del catálogo del sistema.

Cuando se completa la unidad de trabajo actual, se desactiva el atributo NOT LOGGED INITIALLY y se anotan cronológicamente todas las operaciones realizadas en la tabla en unidades de trabajo subsiguientes.

Si se utiliza esta opción para evitar bloqueos en las tablas del catálogo mientras se insertan datos, es importante que sólo se especifique esta cláusula en la sentencia ALTER TABLE. La utilización de cualquier otra cláusula en la sentencia ALTER TABLE dará como resultado bloqueos de catálogo. Si no se especifican otras cláusulas para la sentencia ALTER TABLE, sólo se conseguirá un bloqueo SHARE en las tablas del catálogo del sistema. Esto puede reducir en gran medida la posibilidad de conflictos de simultaneidad en el intervalo de tiempo entre cuando se ejecuta esta sentencia y cuando finaliza la unidad de trabajo en la que se ha ejecutado.

Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

Para obtener más información acerca del atributo NOT LOGGED INITIALLY, consulte la descripción de este atributo en “CREATE TABLE”.

Nota: Si se produce actividad que no es de anotaciones cronológicas para una tabla que tiene activado el atributo NOT LOGGED INITIALLY y si una sentencia no se ejecuta satisfactoriamente (dando lugar a una retrotracción) o si se ejecuta ROLLBACK TO SAVEPOINT, se retrotraerá la unidad de trabajo completa (SQL1476N). Además, la tabla para la que se había activado el atributo NOT LOGGED INITIALLY se marcará como no accesible tras producirse la retrotracción y sólo podrá eliminarse. Por lo tanto, debe minimizarse toda oportunidad de errores dentro de la unidad de trabajo en la que se haya activado el atributo NOT LOGGED INITIALLY.

WITH EMPTY TABLE
Causa la eliminación de todos los datos que se encuentran actualmente en una tabla. Una vez eliminados los datos, no pueden recuperarse a no ser que se utilice el recurso RESTORE. Si la unidad de trabajo en la que se ha emitido esta sentencia Alter se retrotrae, los datos de la tabla NO volverán a su estado original.

Cuando se solicite esta acción, no se activará ningún activador DELETE definido en la tabla afectada. También se vaciará cualquier índice que exista en la tabla.

PCTFREE *entero*
Especifica el porcentaje de cada página que se debe dejar como espacio libre durante una operación de carga o de reorganización de tabla. La primera fila de cada página se añade sin restricciones. Cuando se añaden filas adicionales a una página, al menos *enero* por ciento de la página se deja como espacio libre. El valor PCTFREE sólo se tienen en cuenta en los programas de utilidad de carga y de reorganización de tablas. El valor de *entero* entra en un rango que va de 0 a 99. El valor de PCTFREE -1 en el catálogo (SYSCAT.TABLES) se interpreta como el valor por omisión. El valor de PCTFREE por omisión para una página de tabla es 0. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

LOCKSIZE
Indica el tamaño (granularidad) de los bloqueos utilizados cuando se accede a la tabla. La utilización de esta opción en la definición de la tabla no evitará que se produzca un descenso escalonado de bloqueos normal. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ROW
Indica la utilización de bloqueos de fila. Este es el tamaño de bloqueo por omisión cuando se crea una tabla.

TABLE
Indica la utilización de bloqueos de tabla. Esto significa que se consigue el compartimiento adecuado o bloqueo exclusivo en la tabla y no se utilizan intentos de bloqueo (excepto el valor de ningún intento). La utilización de este valor puede mejorar el rendimiento de las consultas al limitarse el número de bloqueos que deben conseguirse. No obstante, también se reduce la simultaneidad porque todos los bloqueos están mantenidos sobre la tabla completa.

APPEND
Indica si los datos se añaden al final de los datos de la tabla o si se colocan donde haya espacio libre disponible en las páginas de datos. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ON
Indica que los datos de la tabla se añadirán y no se conservará la información sobre el espacio libre de las páginas. La tabla no debe tener un índice agrupado (SQLSTATE 428CA).

OFF
Indica que los datos de la tabla se colocarán donde haya espacio disponible. Este es el valor por omisión cuando se crea una tabla.

Se debe reorganizar la tabla después de establecer APPEND OFF porque la información sobre el espacio libre disponible no es exacta y puede dar como resultado un rendimiento bajo durante la inserción.

VOLATILE CARDINALITY o **NOT VOLATILE CARDINALITY**
Indica al optimizador si la cardinalidad de la tabla *nombre-tabla* puede variar significativamente en tiempo de ejecución o no. La volatilidad se aplica al número de filas de la tabla, no a la propia tabla. CARDINALITY es una palabra clave opcional. El valor por omisión es NOT VOLATILE.

VOLATILE
Especifica que la cardinalidad de la tabla *nombre-tabla* puede variar

significativamente en tiempo de ejecución, de vacía a grande. Para acceder a la tabla, el optimizador utilizará una exploración de índice (en vez de una exploración de tabla, sin tener en cuenta las estadísticas) si el índice es sólo índice (todas las columnas de referencia se encuentran en el índice) o si puede aplicar un predicado en la exploración de índice. No se utilizará el método de acceso de captación previa de lista para acceder a la tabla. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

NOT VOLATILE
Especifica que la cardinalidad de *nombre-tabla* no es volátil. Los planes de acceso a esta tabla seguirán basándose en las estadísticas existentes y en el nivel de optimización actual.

VALUE COMPRESSION
Especifica si los valores de datos NULL y de longitud 0 van a almacenarse o no de forma más eficaz para la mayoría de tipos de datos. También determina el formato de fila que va a utilizarse. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ACTIVATE
Especifica que los valores de datos de longitud 0 para las columnas cuyo tipo de datos es BLOB, CLOB, DBCLOB, LONG VARCHAR o LONG VARGRAPHIC van a almacenarse utilizando el espacio mínimo. Cada valor NULL se almacena sin utilizar un byte adicional. El formato de fila que se utiliza para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos durante las actualizaciones. El nuevo formato de fila (que, para una columna, se especifica mediante la opción COMPRESS SYSTEM DEFAULT) también permite que los valores por omisión del sistema correspondientes a la columna puedan almacenarse de forma más eficaz.

DEACTIVATE
Especifica que los valores NULL van a almacenarse con el espacio establecido aparte para posibles actualizaciones futuras. Este espacio no se establece aparte para las columnas de longitud variable. El formato de fila utilizado determina el número total de bytes de cada tipo de datos. Además, no da soporte al almacenamiento eficaz de los valores por omisión del sistema correspondientes a una columna. Si la columna ya existe y tiene el atributo COMPRESS SYSTEM DEFAULT, se devolverá un mensaje de aviso (SQLSTATE 01648).

Una operación de actualización dará lugar a que una fila existente cambie, adquiriendo el nuevo formato de fila. Para mejorar el rendimiento de las operaciones de actualización en las filas existentes, se recomienda la reorganización de tablas fuera de línea.

LOG INDEX BUILD
Especifica el nivel de anotación cronológica que se debe realizar durante las operaciones de crear, volver a crear o reorganizar el índice de esta tabla.

NULL
Especifica que se utilizará el valor del parámetro de configuración de base de datos *logindexbuild* para determinar si se debe realizar una anotación cronológica completa de las operaciones de creación de índice o no. Es el valor por omisión cuando se crea la tabla.

OFF
Especifica que se realizará una anotación cronológica mínima de cualquier

operación de creación de índice de esta tabla. Este valor prevalece sobre el valor del parámetro de configuración de base de datos *logindexbuild*.

ON
Especifica que se realizará una anotación cronológica completa de las operaciones de creación de índice de esta tabla. Este valor prevalece sobre el valor del parámetro de configuración de base de datos *logindexbuild*.

Normas:

- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de particionamiento, si hay una (SQLSTATE 42997).
- Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).
- Únicamente se puede hacer referencia a una columna en una cláusula ADD o ALTER COLUMN de una sola sentencia ALTER TABLE (SQLSTATE 42711).
- No se puede modificar una longitud de columna si la tabla tiene alguna tabla de consultas materializadas que dependa de la tabla (SQLSTATE 42997).
- Se debe establecer la tabla en estado pendiente de comprobación, utilizando la sentencia SET INTEGRITY (SQLSTATE 55019), antes de:
  - Añadir una columna con una expresión de generación
  - Modificar la expresión generada de una columna
  - Cambiar una columna para que contenga una expresión generada

Notas:

- La modificación de una tabla para convertirla en una tabla de consultas materializadas establecerá la tabla en estado pendiente de comprobación. Si la tabla está definida como REFRESH IMMEDIATE, se debe sacar la tabla del estado de comprobación pendiente para poder invocar mandatos INSERT, DELETE o UPDATE para la tabla referenciada por la selección completa. La tabla se puede sacar del estado de comprobación pendiente utilizando REFRESH TABLE o SET INTEGRITY, con la opción IMMEDIATE CHECKED, para renovar completamente los datos de la tabla de acuerdo con la selección completa. Si los datos de la tabla reflejan fielmente el resultado de la selección completa, se puede utilizar la opción IMMEDIATE UNCHECKED de SET INTEGRITY para sacar la tabla del estado de comprobación pendiente.
- La modificación de una tabla con el fin de cambiarla por una tabla de consultas materializadas REFRESH IMMEDIATE dará lugar a que se invalide cualquier paquete sujeto a la utilización de INSERT, DELETE o UPDATE en la tabla a la que hace referencia la selección completa.
- La modificación de una tabla con el fin de cambiar una tabla de consultas materializadas por una tabla normal dará lugar a que se invalide cualquier paquete que dependa de la tabla.
- La modificación de una tabla de consultas materializadas MAINTAINED BY FEDERATED_TOOL para que sea una tabla normal no efectuará ningún cambio en la configuración de suscripción de la herramienta de duplicación. Puesto que un cambio posterior en una tabla de consultas materializadas MAINTAINED BY SYSTEM hará que falle la herramienta de duplicación, deberá cambiar el valor de suscripción al cambiar la tabla de consultas materializadas MAINTAINED BY FEDERATED_TOOL.
- Si una tabla de consultas materializadas diferida se ha asociado a una tabla de etapas, la tabla de etapas se eliminará si la tabla de consultas materializadas se altera con el fin de convertirla en una tabla normal.

- Las cláusulas ADD se procesan antes que todas las demás cláusulas. Las demás cláusulas se procesan en el orden en que se especifican.
- Ninguna columna añadida mediante ALTER TABLE se añadirá automáticamente a ninguna vista existente de la tabla.
- Cuando se crea un índice automáticamente para una restricción de clave primaria o de unicidad, el gestor de bases de datos intentará utilizar el nombre de la restricción especificada como el nombre de índice con un nombre de esquema que coincida con el nombre de la tabla. Si esto coincide con un nombre de índice existente o no se ha especificado ningún nombre para la restricción, el índice se crea en el esquema SYSIBM con un nombre generado por el sistema y formado por ″SQL″ seguido de una secuencia de 15 caracteres numéricos, generados por una función basada en la indicación de la hora.
- Se dice que cualquier tabla que participe en una operación DELETE de la tabla T está *conectada-con-supresión* a T. Así, una tabla está conectada con supresión a T si es dependiente de T o es dependiente de una tabla en la que se realizan supresiones de T en cascada.
- Un paquete tiene una utilización de inserción (actualización/supresión) en la tabla T si los registros se insertan en (actualizan en/suprimen de) T, directamente por una sentencia en el paquete, o indirectamente por las restricciones o los activadores ejecutados por el paquete en nombre de una de sus sentencias. De manera similar, un paquete tiene una utilización de actualización sobre una columna si la columna se modifica directamente por una sentencia del paquete, o indirectamente por las restricciones o los activadores ejecutados por el paquete en nombre de una de sus sentencias.
- En un sistema federado, se puede modificar una tabla base remota que se ha creado utilizando un DDL transparente. Sin embargo, el DDL transparente impone algunos límites en las modificaciones que pueden efectuarse:
  - Una tabla base remota sólo se puede modificar añadiendo nuevas columnas o especificando una clave primaria.
  - No se puede especificar un comentario en una columna existente de una tabla base remota.
  - No se puede modificar ni descartar una clave primaria existente en una tabla base remota.
  - La modificación de una tabla base remota invalida cualquier paquete que dependa del apodo asociado a esa tabla base remota.
  - La fuente de datos remota debe dar soporte a los cambios que se piden con la sentencia ALTER TABLE. Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.
  - El intento de modificar una tabla base remota que no se ha creado utilizando un DDL transparente devuelve un error.
- Los cambios en la clave primaria, claves de unicidad o claves foráneas pueden tener el siguiente efecto en los paquetes, índices y otras claves foráneas.
  - Si se añade una clave primaria o una clave de unicidad:
    - Ello no afecta a los paquetes, a las claves foráneas ni a las claves exclusivas existentes. (Si la clave primaria o exclusiva utiliza un índice exclusivo existente que se ha creado en una versión anterior y que no se ha convertido para dar soporte a la unicidad diferida, el índice se convierte y se invalidan los paquetes sujetos a la utilización de UPDATE en la tabla asociada.)
  - Si se elimina una clave primaria o una clave de unicidad:

- El índice se elimina si se ha creado automáticamente para la restricción. Cualquier paquete dependiente del índice se invalida.
- El índice se vuelve a establecer como de no unicidad si se había convertido en de unicidad para la restricción y el sistema ya no lo sigue necesitando. Cualquier paquete dependiente del índice se invalida.
- El índice se establece como ya no necesario para el sistema si era un índice de unicidad existente utilizado para la restricción. No tiene ningún efecto en los paquetes.
- Se eliminan todas las claves foráneas dependientes. Se realiza una acción adicional para cada clave foránea dependiente, como se especifica en el punto siguiente.

– Si se añade, elimina o altera una clave primaria de NOT ENFORCED a ENFORCED (o de ENFORCED a NOT ENFORCED):
  - Se invalidan todos los paquetes que estén sujetos a inserción en la tabla de objetos.
  - Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave foránea.
  - Se invalidan todos los paquetes que estén sujetos a supresión en la tabla padre.
  - Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.

– Si se modifica una clave foránea o una dependencia funcional ENABLE QUERY OPTIMIZATION por DISABLE QUERY OPTIMIZATION:
  - Se invalidan todos los paquetes que dependan de la restricción para la realización de la optimización.

– La adición de una columna a una tabla dará como resultado la invalidación de todos los paquetes sujetos a inserción en la tabla alterada. Si la columna añadida es la primera columna de tipo estructurado definido por el usuario de la tabla, también se invalidarán los paquetes sujetos a DELETE en la tabla modificada.

– La adición de una restricción de comprobación o de referencia a una tabla que ya existe y que no está en estado pendiente de comprobación o la modificación de la restricción de comprobación o de referencia existente de NOT ENFORCED a ENFORCED en una tabla existente que no está en estado pendiente de comprobación dará lugar a que las filas existentes de la tabla se evalúen inmediatamente para la restricción. Si la verificación resulta anómala, se emitirá un error (SQLSTATE 23512). Si una tabla está en estado pendiente de comprobación, la adición de una restricción de comprobación o de referencia o la modificación de una restricción de NOT ENFORCED a ENFORCED no dará lugar a la aplicación inmediata de la restricción. En cambio, se actualizarán los distintivos de tipo de restricción utilizados en la operación pendiente de comprobación. Emita la sentencia SET INTEGRITY para iniciar la aplicación de la restricción.

– La adición, modificación o eliminación de una restricción de comprobación dará lugar a la invalidación de todos los paquetes sujetos a una utilización de INSERT en la tabla de objetos, a una utilización de UPDATE en, como mínimo, una de las columnas implicadas en la restricción o a una utilización de SELECT que se beneficie de la restricción para mejorar el rendimiento.

– La adición de una clave de particionamiento producirá la invalidación de todos los paquetes sujetos a actualización en al menos una columna de la clave de particionamiento.

- Una clave de particionamiento que esté predefinida como primera columna de la clave primaria no se ve afectada por la eliminación de la clave primaria y la adición de una clave primaria diferente.
- La modificación de una columna para aumentar la longitud invalidará todos los paquetes que hagan referencia a la tabla (directa o indirectamente a través de un activador o una restricción de referencia) con la columna alterada.
- La modificación de una columna para aumentar la longitud regenerará las vistas (excepto las vistas con tipo) que dependan de la tabla. Si se produce algún error durante la regeneración de una vista, se devuelve un error (SQLSTATE 56098). Las vistas con tipo dependientes de la tabla se marcan como no operativas.
- La modificación de una columna para aumentar la longitud puede causar errores (SQLSTATE 54010) al procesar activadores cuando se prepare o enlace una sentencia que involucre al activador. Esto puede ocurrir cuando la longitud de fila basada en la suma de las longitudes de las variables de transición y las columnas de tabla de transición es demasiado larga. Si se elimina este activador, un intento subsiguiente para crearlo producirá un error (SQLSTATE 54040).
- Las columnas de tipo VARCHAR y VARGRAPHIC que se han modificado para ser mayores que 4000 y 2000 respectivamente, no se deben utilizar como parámetros de entrada en las funciones del esquema SYSFUN (SQLSTATE 22001).
- La modificación de una columna de tipo estructurado con el fin de incrementar la longitud en línea invalidará todos los paquetes que hacen referencia a la tabla, ya sea directa o indirectamente por medio de un activador o de una restricción de referencia.
- La modificación de una columna de tipo estructurado con el fin de incrementar la longitud en línea volverá a generar las vistas que dependen de la tabla.
- Cambiar LOCKSIZE para una tabla dará como resultado una invalidación de todos los paquetes que dependan de la tabla alterada.
- La cláusula ACTIVATE NOT LOGGED INITIALLY no puede utilizarse cuando se están añadiendo columnas DATALINK con el atributo FILE LINK CONTROL a la tabla (SQLSTATE 42613).
- Al cambiar VOLATILE o NOT VOLATILE CARDINALITY se obtendrá como resultado una invalidación de todos los paquetes que tienen una dependencia en la tabla alterada.
- Los clientes de duplicación deben actuar con precaución cuando aumenten la longitud de las columnas VARCHAR. La tabla de datos de cambio asociada con una tabla de aplicaciones podría estar en el límite de tamaño de fila de DB2 o bien podría acercarse al mismo. La tabla de datos de cambio se debe modificar antes de modificar la tabla de aplicaciones o bien se deben modificar las dos dentro de la misma unidad de trabajo, para asegurarse de que la modificación puede realizarse para ambas tablas. Hay que tener en cuenta las copias, que también pueden estar en el límite de tamaño de fila o bien cerca del mismo o bien residir en plataformas que carezcan de la característica para aumentar la longitud de una columna existente.

  Si la tabla de datos de cambio no se altera antes de que el programa Capture procese los registros de anotación cronológica con la longitud de columna VARCHAR aumentada, posiblemente el programa Capture falle. Si una copia que contiene la columna VARCHAR no se modifica antes de que se ejecute la suscripción que mantiene la copia, posiblemente la suscripción fallará.
- ***Compatibilidades***

- Para mantener la compatibilidad con las versiones anteriores de DB2:
  - La palabra clave ADD es opcional para lo siguiente:
    - Restricciones PRIMARY KEY sin nombre
    - Restricciones de referencia sin nombre
    - Restricciones de referencia cuyo nombre aparezca a continuación de la frase FOREIGN KEY
  - La palabra clave CONSTRAINT puede omitirse en una *definición-columna* que defina a una cláusula de referencias
  - El *nombre-restricción* puede especificarse a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT)
  - SET SUMMARY AS puede especificarse en lugar de SET MATERIALIZED QUERY AS
  - SET MATERIALIZED QUERY AS DEFINITION ONLY puede especificarse en lugar de DROP MATERIALIZED QUERY
  - SET MATERIALIZED QUERY AS (selección completa) puede especificarse en lugar de ADD MATERIALIZED QUERY (selección completa)
- Para mantener la compatibilidad con las versiones anteriores de DB2 y por coherencia:
  - Puede utilizarse una coma para separar varias opciones en la cláusula de *modificación-identidad*.
- También recibe soporte la sintaxis siguiente:
  - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER

Ejemplos:

Ejemplo 1: Añada una nueva columna llamada RATING, que tiene un carácter de longitud, a la tabla DEPARTMENT.

```
ALTER TABLE DEPARTMENT
   ADD RATING CHAR(1)
```

Ejemplo 2: Añada una nueva columna llamada SITE_NOTES a la tabla PROJECT. Cree SITE_NOTES como una columna de longitud variable con una longitud máxima de 1000 caracteres. Los valores de la columna no tienen un juego de caracteres asociado y, por lo tanto, no deben convertirse.

```
ALTER TABLE PROJECT
   ADD SITE_NOTES  VARCHAR(1000) FOR BIT DATA
```

Ejemplo 3: Supongamos que existe una tabla llamada EQUIPMENT definida con las siguientes columnas:

```
Nombre columna     Tipo datos
   EQUIP_NO           INT
   EQUIP_DESC         VARCHAR(50)
   LOCATION           VARCHAR(50)
   EQUIP_OWNER        CHAR(3)
```

Añada una restricción de referencia a la tabla EQUIPMENT, de manera que el propietario (EQUIP_OWNER) sea un número de departamento (DEPTNO) que esté presente en la tabla DEPARTMENT. DEPTNO es la clave primaria de la tabla DEPARTMENT. Si se elimina un departamento de la tabla DEPARTMENT, los valores del propietario (EQUIP_OWNER) referentes a todo el equipo propiedad de dicho departamento deben desasignarse (o establecerse en nulo). Dé a la restricción el nombre de DEPTQUIP.

```
ALTER TABLE EQUIPMENT
     ADD CONSTRAINT DEPTQUIP
      FOREIGN KEY (EQUIP_OWNER)
        REFERENCES DEPARTMENT
           ON DELETE SET NULL
```

Además, se necesita una columna adicional para permitir el registro de la cantidad asociada con este registro de equipo. A menos que se especifique lo contrario, la columna EQUIP_QTY debe tener un valor de 1 y nunca debe ser nulo.

```
ALTER TABLE EQUIPMENT
  ADD COLUMN EQUIP_QTY
  SMALLINT NOT NULL DEFAULT 1
```

Ejemplo 4: Modifique la tabla EMPLOYEE. Añada la restricción de comprobación denominada REVENUE, definida de tal manera que cada empleado debe recibir una cantidad total, entre el salario y la comisión, superior a 3.000.000 de pesetas.

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE
  CHECK (SALARY + COMM > 30000)
```

Ejemplo 5: Modifique la tabla EMPLOYEE. Elimine la restricción REVENUE que se ha definido anteriormente.

```
ALTER TABLE EMPLOYEE
  DROP CONSTRAINT REVENUE
```

Ejemplo 6: Modifique una tabla para anotar cronológicamente los cambios SQL en el formato por omisión.

```
ALTER TABLE SALARY1
  DATA CAPTURE NONE
```

Ejemplo 7: Modifique una tabla para anotar cronológicamente los cambios SQL en un formato expandido.

```
ALTER TABLE SALARY2
  DATA CAPTURE CHANGES
```

Ejemplo 8: Modifique la tabla EMPLOYEE para añadir 4 nuevas columnas con los valores por omisión.

```
ALTER TABLE EMPLOYEE
ADD COLUMN HEIGHT MEASURE   DEFAULT MEASURE(1)
ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
ADD COLUMN FLAGS BLOB(1M)  DEFAULT BLOB(X'01')
ADD COLUMN PHOTO PICTURE   DEFAULT BLOB(X'00')
```

Los valores por omisión utilizan varios nombres de función al especificar el valor por omisión. Debido a que MEASURE es un tipo diferenciado basado en INTEGER, se utiliza la función MEASURE. El valor por omisión de la columna HEIGHT se podría haber especificado sin la función debido a que el tipo de fuente de MEASURE no es BLOB ni un tipo de datos de indicación de fecha y hora. Debido a que BIRTHDATE es un tipo diferenciado basado en DATE, se utiliza la función DATE (BIRTHDATE no puede utilizarse aquí). Para las columnas FLAGS y PHOTO el valor por omisión se especifica utilizando la función BLOB aunque PHOTO es un tipo diferenciado. Para especificar un valor por omisión para las columnas BIRTHDAY, FLAGS y PHOTO, se debe utilizar una función porque el tipo es BLOB o bien un tipo diferenciado con origen en un tipo de datos BLOB o de indicación de fecha y hora.

Ejemplo 9: Se ha definido una tabla denominada CUSTOMERS con las siguientes columnas:

```
Nombre columna     Tipo datos
    BRANCH_NO          SMALLINT
    CUSTOMER_NO        DECIMAL(7)
    CUSTOMER_NAME      VARCHAR(50)
```

En esta tabla, la clave primaria está formada por las columnas BRANCH_NO y CUSTOMER_NO. Para particionar la tabla, deberá crear una clave de particionamiento para la tabla. La tabla deberá definirse en un espacio de tablas de un grupo de particiones de base de datos de un único nodo. La clave primaria debe ser un superconjunto de las columnas de particionamiento: como mínimo, una de las columnas de la clave primaria debe utilizarse como clave de particionamiento. Establezca BRANCH_NO como clave de particionamiento, tal como se indica a continuación:

```
ALTER TABLE CUSTOMERS
      ADD PARTITIONING KEY (BRANCH_NO)
```

Ejemplo 10: Se ha creado la tabla remota EMPLOYEE en un sistema federado utilizando un DDL transparente. Altere la tabla remota EMPLOYEE para añadir las columnas PHONE_NO y WORK_DEPT; añada, también, una clave primaria en la columna existente EMP_NO y la nueva columna WORK_DEPT.

```
ALTER TABLE EMPLOYEE
  ADD COLUMN PHONE_NO CHAR(4) NOT NULL
  ADD COLUMN WORK_DEPT CHAR(3)
  ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

Ejemplo 11: Modifique la tabla DEPARTMENT para añadir la dependencia funcional FD1, después descarte la dependencia funcional FD1 de la tabla DEPARTMENT.

```
ALTER TABLE DEPARTMENT
  ADD COSTRAINT FD1
    CHECK ( DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED

ALTER TABLE DEPARTMENT
  DROP CHECK FD1
```

Ejemplo 12: Cambie el valor por omisión de la columna WORKDEPT de la tabla EMPLOYEE por 123.

```
ALTER TABLE EMPLOYEE
  ALTER COLUMN WORKDEPT
    SET DEFAULT '123'
```

Conceptos relacionados:

- "What is transparent DDL?" en la publicación *Federated Systems Guide*

Tareas relacionadas:

- "Altering remote tables using transparent DDL" en la publicación *Federated Systems Guide*

Información relacionada:

- "ALTER TYPE (Estructurado)" en la página 89
- "CREATE TABLE" en la página 341
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*
- "Procedimiento ALTOBJ" en la publicación *SQL Administrative Routines*

Ejemplos relacionados:

- "dbrecov.sqc -- How to recover a database (C)"

- “tbconstr.sqc -- How to create, use, and drop constraints (C)”
- “dbrecov.sqC -- How to recover a database (C++)”
- “dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)”
- “tbconstr.sqC -- How to create, use, and drop constraints (C++)”
- “TbGenCol.java -- How to use generated columns (JDBC)”

ALTER TABLESPACE

La sentencia ALTER TABLESPACE se utiliza para modificar un espacio de tablas existente para:

- Añadir un contenedor a un espacio de tablas DMS o descartarlo del mismo; es decir, un espacio creado con la opción MANAGED BY DATABASE.
- Modificar el tamaño de un contenedor en un espacio de tablas DMS.
- Añadir un contenedor a un espacio de tablas SMS en una partición que actualmente no tiene ningún contenedor.
- Modificar el valor PREFETCHSIZE para un espacio de tablas.
- Modificar el valor BUFFERPOOL utilizado para las tablas del espacio de tablas.
- Modificar el valor OVERHEAD para un espacio de tablas.
- Modificar el valor TRANSFERRATE para un espacio de tablas.
- Modificar la política de almacenamiento en antememoria del sistema de archivos para un espacio de tablas.

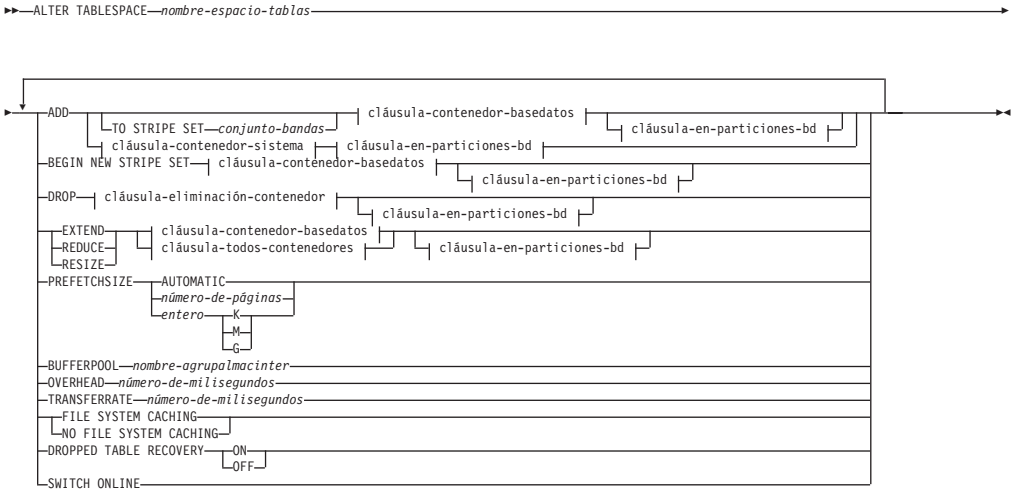
Invocación:

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

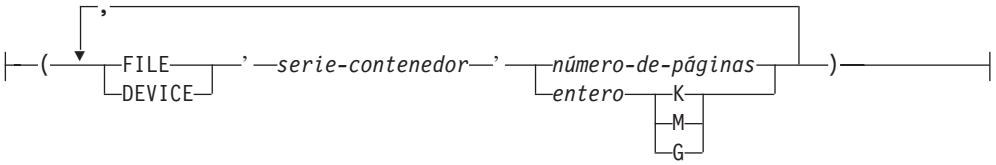
Autorización:

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

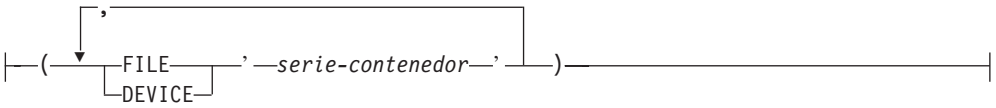
Sintaxis:



cláusula-contenedor-basedatos:

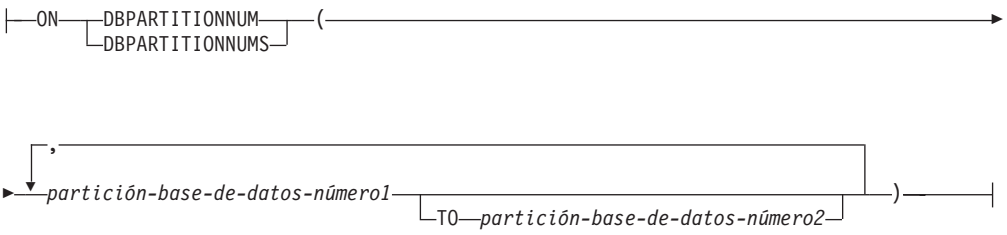


cláusula-eliminación-contenedor:

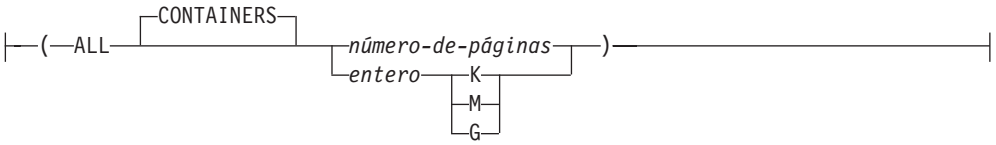


cláusula-contenedor-sistema:

cláusula-en-particiones-bd:



cláusula-todos-contenedores:



Descripción:

nombre-espacio-tablas
Nombra el espacio de tablas. Es un nombre que consta de una sola parte. Es un identificador SQL largo (ordinario o delimitado).

ADD
Especifica que se deben añadir uno o varios contenedores nuevos al espacio de tablas.

TO STRIPE SET *conjunto-bandas*
Especifica que se deben añadir uno o varios contenedores nuevos al espacio de tablas, y que se colocarán en el conjunto de bandas especificado.

BEGIN NEW STRIPE SET
Especifica que se debe crear un conjunto de bandas nuevo en el espacio de tablas, y que se deben añadir uno o varios contenedores a este conjunto de

bandas nuevo. Los contenedores que se añadan posteriormente utilizando la opción ADD se añadirán a este nuevo conjunto de bandas a menos que se especifique TO STRIPE SET.

DROP
Especifica que se deben descartar uno o varios contenedores del espacio de tablas.

EXTEND
Especifica que va a aumentarse el tamaño de los contenedores existentes. El tamaño especificado es el tamaño en el que se aumenta el contenedor existente. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas experimentarán un aumento de este tamaño.

REDUCE
Especifica que va a reducirse el tamaño de los contenedores existentes. El tamaño que se especifica es el tamaño que se reducirá el contenedor existente. Si se especifica *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas experimentarán una reducción de este tamaño.

RESIZE
Especifica que va a cambiarse el tamaño de los contenedores existentes. El tamaño especificado es el nuevo tamaño del contenedor. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas se cambiarán a este tamaño. Si la operación afecta a más de un contenedor, deberá aumentarse o disminuirse el tamaño de todos esos contenedores. No es posible aumentar el tamaño de algunos de ellos y, al mismo tiempo, reducir el de otros (SQLSTATE 429BC).

cláusula-contenedor-basedatos
Añade uno o varios contenedores a un espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

cláusula-eliminación-contenedor
Descarta uno o varios contenedores del espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

cláusula-contenedor-sistema
Añade uno o varios contenedores a un espacio de tablas SMS de las particiones especificadas. El espacio de tablas debe identificar un espacio de tablas SMS que ya exista en el servidor de aplicaciones. No debe haber ningún contenedor en las particiones especificadas para el espacio de tablas. (SQLSTATE 42921).

cláusula-en-particiones-bd
Especifica una o más particiones para las correspondientes operaciones de contenedor.

cláusula-todos-contenedores
Amplía, reduce o cambia el tamaño de todos los contenedores de un espacio de tablas DMS. El espacio de tablas debe identificar un espacio de tablas DMS que ya exista en el servidor de aplicaciones.

PREFETCHSIZE
La captación previa lee los datos que una consulta necesita antes que la consulta haga referencia a ellos, por lo que la consulta no necesita esperar a que se efectúen operaciones de E/S.

AUTOMATIC
Especifica que el tamaño de captación previa de un espacio de tablas se

debe actualizar automáticamente; es decir, DB2 gestionará el tamaño de captación previa, utilizando la fórmula siguiente:

```
Tamaño de captación previa =
 (número de contenedores) *
 (número de discos físicos por contenedor) *
 (tamaño extensión)
```

El número de discos físicos por contenedor toma por omisión el valor 1, a menos que se especifique un valor en la variable de registro DB2_PARALLEL_IO.

DB2 actualizará automáticamente el tamaño de captación previa cuando cambie el número de contenedores en un espacio de tablas (después de la ejecución satisfactoria de una sentencia ALTER TABLESPACE que añada o descarte uno o varios contenedores). El tamaño de captación previa se actualiza al arrancar la base de datos.

La actualización automática del tamaño de captación previa puede desactivarse especificando un valor numérico en la cláusula PREFETCHSIZE.

número-de-páginas
Especifica el número de páginas PAGESIZE del espacio de tablas que se leerán cuando se realice la captación previa de datos. El valor del tamaño de captación previa también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica así, se utilizará el nivel mínimo del número de bytes dividido por el tamaño de página para determinar el valor de número de páginas para el tamaño de captación previa.

BUFFERPOOL *nombre-agrupalmacinter*
El nombre de la agrupación de almacenamientos intermedios utilizado para las tablas de este espacio de tablas. La agrupación de almacenamientos intermedios debe existir actualmente en la base de datos (SQLSTATE 42704). Debe haberse definido el grupo de particiones de base de datos del espacio de tablas para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos*
Cualquier literal numérico (entero, decimal o de coma flotante) que especifique la actividad general del controlador de E/S y el tiempo de búsqueda y latencia del disco, en milisegundos. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

TRANSFERRATE *número-de-milisegundos*
Cualquier literal numérico (entero, decimal o de coma flotante) que especifique el tiempo para leer una página (de 4K u 8K)en la memoria en milisegundos. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

FILE SYSTEM CACHING o **NO FILE SYSTEM CACHING**
Especifica si las operaciones de E/S se almacenarán en antememoria a nivel del sistema de archivos o no. Las conexiones con la base de datos deben terminar para que surta efecto una nueva política de almacenamiento en antememoria.

FILE SYSTEM CACHING
Todas las operaciones de E/S del espacio de tablas de destino se almacenarán en antememoria a nivel del sistema de archivos.

NO FILE SYSTEM CACHING
Todas las operaciones de E/S evitarán el almacenamiento en antememoria a nivel del sistema de archivos.

DROPPED TABLE RECOVERY
Las tablas descartadas del espacio de tablas especificado se pueden recuperar utilizando la opción RECOVER DROPPED TABLE ON del mandato ROLLFORWARD.

SWITCH ONLINE
Los espacios de tablas en estado OFFLINE se ponen en línea si los contenedores han pasado a ser accesibles. Si los contenedores no son accesibles, se devuelve un error (SQLSTATE 57048).

Notas:

- *Compatibilidades*
  - Para mantener la compatibilidad con las versiones anteriores a la Versión 8, la palabra clave:
    - NODE puede sustituirse por DBPARTITIONNUM
    - NODES puede sustituirse por DBPARTITIONNUMS
- Cada definición de contenedor requiere 53 bytes más el número de bytes necesario para almacenar el nombre de contenedor. La longitud combinada de todos los nombres de contenedores para el espacio de tablas no puede exceder de 20 480 bytes (SQLSTATE 54034).
- Las operaciones de contenedor por omisión son operaciones de contenedor que se han especificado en la sentencia ALTER TABLESPACE, pero que no se han dirigido de forma explícita a una partición de base de datos específica. Estas operaciones de contenedor se envían a cualquier partición de base de datos que no aparezca en la lista de la sentencia. Si estas operaciones de contenedor por omisión no se envían a ninguna partición de base de datos porque todas las particiones de base de datos se han mencionado explícitamente para una operación de contenedor, se devuelve un mensaje de aviso (SQLSTATE 1758W).
- Tras añadir o eliminar espacio de un espacio de tablas y confirmar la transacción, se puede reequilibrar el contenido del espacio de tablas entre los contenedores. El acceso al espacio de tablas no se restringe durante el reequilibrio.
- Si el espacio de tablas se encuentra en estado OFFLINE y los contenedores han pasado a ser accesibles, el usuario puede desconectar todas las aplicaciones y volverlas a conectar a la base de datos para que el espacio de tablas salga del estado OFFLINE. De forma alternativa, la opción SWITCH ONLINE puede activar el espacio de tablas (cambiar su estado OFFLINE) mientras el resto de la base de datos sigue activa y se utiliza.
- Si se añade más de un contenedor a un espacio de tablas, se recomienda añadirlos en la misma sentencia para que el reequilibrio sólo deba efectuarse una vez. Si se intenta añadir contenedores al mismo espacio de tablas en sentencias ALTER TABLESPACE diferentes dentro de una sola transacción, se producirá un error (SQLSTATE 55041).
- Si se intenta ampliar, reducir, cambiar el tamaño o eliminar contenedores que no existen, se generará un error (SQLSTATE 428B2).

- Cuando se amplía, se reduce o se cambia el tamaño de un contenedor, el tipo de contenedor debe coincidir con el tipo que se ha utilizando durante la creación del contenedor (SQLSTATE 428B2).
- Si se intenta cambiar los tamaños de los contenedores de un mismo espacio de tablas, utilizando sentencias ALTER TABLESPACE diferentes pero en una sola transacción, se producirá un error (SQLSTATE 55041).
- En una base de datos particionada, si más de una partición de base de datos reside en el mismo nodo físico, no podrá especificarse el mismo dispositivo o vía de acceso específica para tales particiones de base de datos (SQLSTATE 42730). Para este entorno, especifique una *serie-contenedor* específica para cada partición o bien utilice un nombre de vía de acceso relativa.
- Aunque la definición del espacio de tablas sea transaccional y los cambios en la definición se reflejen en las tablas del catálogo tras su confirmación, la agrupación de almacenamientos intermedios con la nueva definición no se podrá utilizar hasta la próxima vez que se inicie la base de datos. La agrupación de almacenamientos intermedios en uso, cuando se ha emitido la sentencia ALTER TABLESPACE, continuará utilizándose mientras tanto.

Normas:

- La cláusula BEGIN NEW STRIPE SET no puede especificarse en la misma sentencia que ADD, DROP, EXTEND, REDUCE y RESIZE, a menos que dichas cláusulas se dirijan a particiones distintas (SQLSTATE 429BC).
- El valor de conjunto de bandas especificado con la cláusula TO STRIPE SET se debe encontrar dentro del rango válido para el espacio de tablas que se está modificando (SQLSTATE 42615).
- Al añadir o eliminar espacio del espacio de tablas, se deben seguir las normas siguientes:
  - EXTEND y RESIZE pueden utilizarse en la misma sentencia, siempre que se aumente el tamaño de cada contenedor (SQLSTATE 429BC).
  - REDUCE y RESIZE pueden utilizarse en la misma sentencia, siempre que se reduzca el tamaño de cada contenedor (SQLSTATE 429BC).
  - EXTEND y REDUCE no pueden utilizarse en la misma sentencia, a menos que se dirijan a particiones distintas (SQLSTATE 429BC).
  - ADD no puede utilizarse con REDUCE o DROP en la misma sentencia, a menos que se dirijan a particiones distintas (SQLSTATE 429BC).
  - DROP no puede utilizarse con EXTEND o ADD en la misma sentencia, a menos que se dirijan a particiones distintas (SQLSTATE 429BC).

Ejemplos:

Ejemplo 1: Añada un dispositivo al espacio de tablas PAYROLL.

```
ALTER TABLESPACE PAYROLL
   ADD (DEVICE '/dev/rhdisk9' 10000)
```

Ejemplo 2: Cambie el tamaño de captación previa y la actividad general de E/S para el espacio de tablas ACCOUNTING.

```
ALTER TABLESPACE ACCOUNTING
  PREFETCHSIZE 64
  OVERHEAD 19.3
```

Ejemplo 3: Cree el espacio de tablas TS1, luego cambie el tamaño de todos los contenedores a 2000 páginas. (Se muestran tres sentencias ALTER TABLESPACE diferentes que efectuarán este cambio de tamaño.)

```
CREATE TABLESPACE TS1
  MANAGED BY DATABASE
  USING (FILE '/conts/cont0' 1000,
         DEVICE '/dev/rcont1' 500,
         FILE 'cont2' 700)
ALTER TABLESPACE TS1
  RESIZE (FILE '/conts/cont0' 2000,
          DEVICE '/dev/rcont1' 2000,
          FILE 'cont2' 2000)
```

O

```
ALTER TABLESPACE TS1
  RESIZE (ALL 2000)
```

O

```
ALTER TABLESPACE TS1
  EXTEND (FILE '/conts/cont0' 1000,
          DEVICE '/dev/rcont1' 1500,
          FILE 'cont2' 1300)
```

Ejemplo 4: Amplíe todos los contenedores del espacio de tablas DATA_TS en 1000 páginas.

```
ALTER TABLESPACE DATA_TS
  EXTEND (ALL 1000)
```

Ejemplo 5: Cambie el tamaño de todos los contenedores del espacio de tablas INDEX_TS a 100 megabytes (MB).

```
ALTER TABLESPACE INDEX_TS
  RESIZE (ALL 100 M)
```

Ejemplo 6: Añada tres nuevos contenedores. Amplíe el primer contenedor y cambie el tamaño del segundo.

```
ALTER TABLESPACE TS0
  ADD (FILE 'cont2' 2000, FILE 'cont3' 2000)
  ADD (FILE 'cont4' 2000)
  EXTEND (FILE 'cont0' 100)
  RESIZE (FILE 'cont1' 3000)
```

Ejemplo 7: El espacio de tablas TSO existe en las particiones 0, 1 y 2. Añada un nuevo contenedor a la partición de base de datos 0. Amplíe todos los contenedores de la partición de base de datos 1. Cambie el tamaño de un contenedor en todas las particiones de base de datos distintas de las particiones que se han especificado explícitamente (es decir, las particiones de base de datos 0 y 1).

```
ALTER TABLESPACE TS0
  ADD (FILE 'A' 200) ON DBPARTITIONNUM (0)
  EXTEND (ALL 200) ON DBPARTITIONNUM (1)
  RESIZE (FILE 'B' 500)
```

La cláusula RESIZE es la cláusula de contenedor por omisión de este ejemplo y se ejecutará en la partición de base de datos 2, ya que, explícitamente, están enviándose otras operaciones a las particiones de base de datos 0 y 1. Sin embargo, si sólo hubieran existido estas dos particiones de base de datos, la sentencia se habría ejecutado satisfactoriamente, pero se habría devuelto un mensaje de aviso (SQL1758W) indicando que se han especificado contenedores por omisión pero que no se han utilizado.

Información relacionada:

• “CREATE TABLESPACE” en la página 405

- "System environment variables" en la publicación *Administration Guide: Performance*

ALTER TYPE (Estructurado)

La sentencia ALTER TYPE se utiliza para añadir o eliminar especificaciones de atributo o de método de un tipo estructurado definido por el usuario. También pueden modificarse las propiedades de los métodos existentes.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema del tipo
- Autorización del usuario que define el tipo, tal como se ha registrado en la columna DEFINER de SYSCAT.DATATYPES

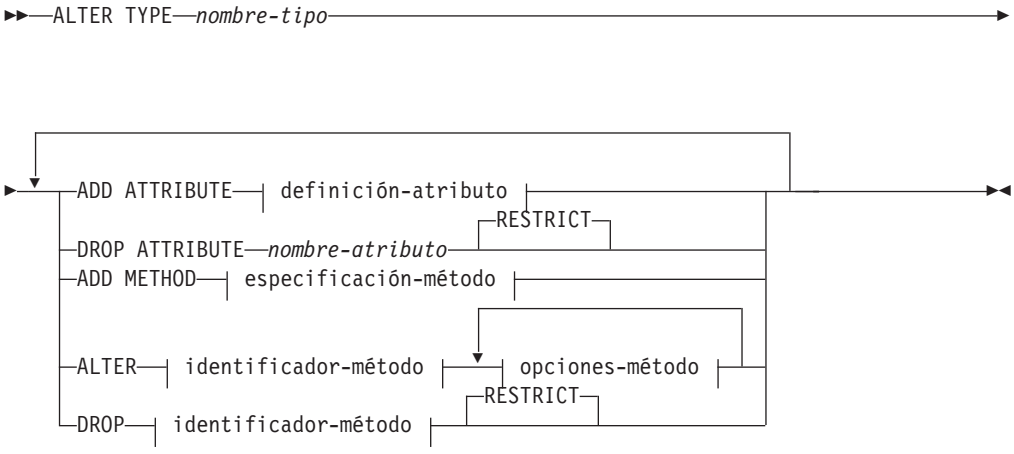
Para modificar un método con el fin de que no esté delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos

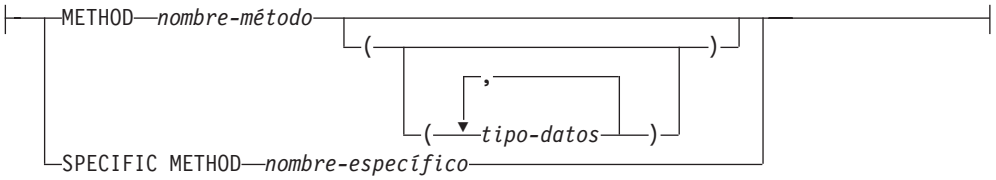
Para modificar un método con el fin de que esté delimitado, no se necesita ninguna autorización ni privilegio adicional.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:



identificador-método:



opciones-método:

Descripción:

nombre-tipo
Identifica el tipo estructurado a cambiar. Debe ser un tipo existente definido en el catálogo (SQLSTATE 42704) y el tipo debe ser un tipo estructurado (SQLSTATE 428DP). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

ADD ATTRIBUTE
Añade un atributo después del último atributo del tipo estructurado existente.

definición-atributo
Define los atributos del tipo estructurado.

nombre-atributo
Especifica un nombre para el atributo. El nombre no puede ser el mismo que el de otro atributo de este tipo estructurado (incluidos los atributos heredados) ni de ningún subtipo de este tipo estructurado (SQLSTATE 42711).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-atributo* (SQLSTATE 42939). Estos nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

tipo-datos 1
Especifica el tipo de datos del atributo. Es uno de los tipos de datos listados en la descripción de CREATE TABLE, excepto LONG VARCHAR, LONG VARGRAPHIC, o un tipo diferenciado basado en LONG VARCHAR o LONG VARGRAPHIC (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". Si el tipo de datos del atributo es un tipo de referencia, el tipo destino de la referencia debe ser un tipo estructurado existente (SQLSTATE 42704).

Un tipo de datos estructurado definido con un atributo de tipo DATALINK sólo puede utilizarse eficazmente como tipo de datos para una tabla con tipo o para una vista con tipo (SQLSTATE 01641).

Para evitar las definiciones de tipo que, durante la ejecución, pueden permitir que una instancia del tipo contengan, directa o indirectamente, otra instancia del mismo tipo o uno de sus subtipos, existe una restricción que establece que un tipo no puede definirse de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP).

opciones-lob
Especifica las opciones asociadas a tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de las opciones-lob, consulte "CREATE TABLE".

opciones-datalink
Especifica las opciones asociadas a tipos DATALINK (o tipos diferenciados basados en tipos DATALINK). Para obtener una descripción detallada de las opciones-datalink, consulte "CREATE TABLE".

Tenga en cuenta que si no se especifica ninguna opción para un tipo de datos DATALINK o un tipo diferenciado que tenga su origen en el tipo DATALINK, LINKTYPE URL y NO LINK CONTROL son las opciones por omisión.

DROP ATTRIBUTE
Elimina un atributo del tipo estructurado existente.

nombre-atributo
El nombre del atributo. El atributo debe existir como un atributo del tipo (SQLSTATE 42703).

RESTRICT
Aplica la norma que establece que ningún atributo puede eliminarse si *nombre-tipo* se utiliza como tipo de una tabla, vista, columna o atributo existente anidado dentro del tipo de una columna o una extensión de índice.

ADD METHOD *especificación-método*
Añade una especificación de método al tipo que *nombre-tipo* identifica. Para poder utilizar el método es necesario darle un cuerpo mediante una sentencia CREATE METHOD separada. Para obtener más información acerca de la *especificación-método*, consulte "CREATE TYPE (Estructurado)".

ALTER *identificador-método*
Identifica de forma exclusiva una instancia de un método que va a modificarse. El método especificado puede o no tener un cuerpo de método existente. Los métodos declarados como LANGUAGE SQL no pueden modificarse (SQLSTATE 42917).

identificador-método

METHOD *nombre-método*
Identifica un método en particular y sólo es válido si existe exactamente una instancia de método con el nombre *nombre-método* para el tipo *nombre-tipo*. Para el método identificado puede definirse cualquier número de parámetros. Si no existe ningún método con este

nombre para el tipo, se generará un error (SQLSTATE 42704). Si existe más de una instancia del método para el tipo, se generará un error (SQLSTATE 42725).

METHOD *nombre-método (tipo-datos,...)*
Proporciona la signatura del método, que identifica de forma exclusiva al método. No se utiliza el algoritmo de resolución de método.

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

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

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

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

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

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

No es necesario que un tipo de FLOAT(*n*) coincida con el valor que se ha definido para *n*, pues $0 < n < 25$ significa REAL y $24 < n < 54$ significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún método con la signatura especificada para el tipo en el esquema indicado o implícito, se generará un error (SQLSTATE 42883).

SPECIFIC METHOD *nombre-específico*
Identifica un método en particular, utilizándose el nombre que se ha especificado o que se ha tomado por omisión durante la creación del método. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de método específica en el esquema indicado o implícito; de lo contrario, se generará un error (SQLSTATE 42704).

opciones-método
Especifica las opciones que van a modificarse para el método.

FENCED o NOT FENCED
Especifica si se considera que el método es seguro para ejecutarse en el

espacio de dirección o en el proceso del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED). La mayoría de métodos tienen la opción de ejecutarse como FENCED o NOT FENCED.

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

PRECAUCIÓN:
La utilización de NOT FENCED para los métodos que no se han codificado, revisado y probado de forma adecuada puede comprometer la integridad de DB2. DB2 dispone de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no puede garantizar la integridad completa cuando se utilizan métodos NOT FENCED.

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

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

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

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

Si el método se ha definido con un LANGUAGE distinto de OLE:

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

DROP *identificador-método*
Identifica de forma exclusiva una instancia de un método que va a eliminarse. El método especificado no debe tener un cuerpo de método existente (SQLSTATE 428ER). Utilice la sentencia DROP METHOD para eliminar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD. Los métodos generados implícitamente por la sentencia CREATE TYPE (tales como mutadores y observadores) no se pueden eliminar (SQLSTATE 42917).

RESTRICT
Indica que el método especificado no puede tener un cuerpo de método existente. Utilice la sentencia DROP METHOD para eliminar el cuerpo del método antes de utilizar ALTER TYPE DROP METHOD.

Normas:

- La adición o eliminación de un atributo no está permitida para el tipo *nombre-tipo* (SQLSTATE 55043) si:
  - El tipo o uno de sus subtipos es el tipo de una tabla o vista existente.
  - Existe una columna de una tabla cuyo tipo utiliza directa o indirectamente el *nombre-tipo*. Los términos *utiliza directamente* y *utiliza indirectamente* se definen en "Tipos estructurados".
  - El tipo o uno de sus subtipos se utiliza en una extensión de índice.
- No se puede modificar un tipo mediante la adición de atributos si el número total de atributos para el tipo o cualquiera de sus subtipos es mayor que 4082 (SQLSTATE 54050).
- Opción ADD ATTRIBUTE:
  - ADD ATTRIBUTE genera métodos de observador y mutador para el nuevo atributo. Estos métodos son similares a los que se generan durante la creación de un tipo estructurado (véase "CREATE TYPE (Estructurado)"). Si estos métodos alteran temporalmente métodos o funciones existentes o entran en conflicto con ellos, la sentencia ALTER TYPE fallará (SQLSTATE 42745).
  - Si el usuario especificó explícitamente un valor menor que 292 para INLINE LENGTH del tipo (o de cualquiera de sus subtipos), y los atributos añadidos hacen que esa longitud especificada sea menor que el tamaño del resultado de la función constructora para el tipo modificado (32 bytes más 10 bytes por cada atributo), se produce un error (SQLSTATE 42611).
- Opción DROP ATTRIBUTE:
  - Un atributo que se ha heredado de un supertipo existente no se puede eliminar (SQLSTATE 428DJ).
  - DROP ATTRIBUTE elimina los métodos de mutador y observador de los atributos eliminados y comprueba si hay dependencias respecto a esos métodos eliminados.
- Opción DROP METHOD:
  - No puede eliminarse un método original que otros métodos alteran temporalmente (SQLSTATE -2).

Notas:

- No es posible modificar un método que esté en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).
- Cuando se altera un tipo, mediante la adición o eliminación de un atributo, se invalidan todos los paquetes que dependen de funciones o métodos que utilizan este tipo o un subtipo de él como parámetro o resultado.
- Cuando un atributo se añade a un tipo estructurado o se elimina de él:
  - Si el valor INLINE LENGTH del tipo fue calculado por el sistema cuando se creó el tipo, los valores INLINE LENGTH se modifican automáticamente para el tipo alterado, y para todos sus subtipos, para reflejar el cambio. Los valores INLINE LENGTH también se modifican automáticamente (de manera recursiva) para todos los tipos estructurados cuando INLINE LENGTH fue calculado por el sistema y el tipo incluye un atributo de cualquier tipo con un valor INLINE LENGTH cambiado.
  - Si el usuario especificó explícitamente el valor INLINE LENGTH de un tipo cualquiera que se alteró mediante la adición o eliminación de atributos, entonces el valor INLINE LENGTH de ese tipo determinado no se modifica. Debe tenerse especial cuidado en el caso de valores INLINE LENGTH especificados explícitamente. Si es probable que más tarde se añadan atributos

a un tipo, el valor de INLINE LENGTH, para cualquier uso de ese tipo o de uno de sus subtipos en una definición de columna, debe ser lo suficiente grande para tener en cuenta el posible aumento de longitud del objeto del que se creado una instancia.

– Si deben habilitarse nuevos atributos para ser utilizados por programas de aplicación, se deben modificar las funciones de transformación existentes para que coincidan con la nueva estructura del tipo de datos.

- En un entorno de bases de datos particionadas, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- ***Privilegios***

  El privilegio EXECUTE no se otorgará para ninguno de los métodos explícitamente especificados en la sentencia ALTER TYPE hasta que se haya definido un cuerpo de método mediante la utilización de la sentencia CREATE METHOD. El usuario que define el tipo definido por el usuario puede eliminar la especificación de método mediante la utilización de la sentencia ALTER TYPE.

Ejemplos:

Ejemplo 1: La sentencia ALTER TYPE puede utilizarse para permitir un ciclo de tablas y tipos que se hacen referencia mutuamente. Piense en las tablas denominadas EMPLOYEE y DEPARTMENT que se hacen referencia mutuamente.

La secuencia siguiente permitiría crear los tipos y las tablas.

```
CREATE TYPE DEPT ...
CREATE TYPE EMP ... (incluido el atributo denominado DEPTREF del tipo REF(DEPT))
ALTER TYPE DEPT ADD ATTRIBUTE MANAGER REF(EMP)
CREATE TABLE DEPARTMENT OF DEPT ...
CREATE TABLE EMPLOYEE OF EMP (DEPTREF WITH OPTIONS SCOPE DEPARTMENT)
ALTER TABLE DEPARTMENT ALTER COLUMN MANAGER ADD SCOPE EMPLOYEE
```

La secuencia siguiente permitiría eliminar estas tablas y tipos.

```
DROP TABLE EMPLOYEE (la columna MANAGER de DEPARTMENT se queda sin ámbito)
DROP TABLE DEPARTMENT
ALTER TYPE DEPT DROP ATTRIBUTE MANAGER
DROP TYPE EMP
DROP TYPE DEPT
```

Ejemplo 2: La sentencia ALTER TYPE puede utilizarse para crear un tipo con un atributo que haga referencia a un subtipo.

```
CREATE TYPE EMP ...
CREATE TYPE MGR UNDER EMP ...
ALTER TYPE EMP ADD ATTRIBUTE MANAGER REF(MGR)
```

Ejemplo 3: La sentencia ALTER TYPE puede utilizarse para añadir un atributo. La sentencia siguiente añade el atributo SPECIAL al tipo EMP. Debido a que la sentencia CREATE TYPE original no especificaba el valor INLINE LENGTH, DB2 lo recalcula añadiendo 13 bytes (10 bytes para el nuevo atributo + longitud del atributo + 2 bytes para un atributo que no es LOB).

```
ALTER TYPE EMP ...
   ADD ATTRIBUTE SPECIAL CHAR(1)
```

Ejemplo 4: La sentencia ALTER TYPE puede utilizarse para añadir un método asociado a un tipo. La sentencia siguiente añade un método llamado BONUS.

```
ALTER TYPE EMP ...
  ADD METHOD BONUS (RATE DOUBLE)
    RETURNS INTEGER
```

```
LANGUAGE SQL
CONTAINS SQL
NO EXTERNAL ACTION
DETERMINISTIC
```

Observe que para utilizar el método BONUS es necesario emitir una sentencia CREATE METHOD para crear el cuerpo del método. Si se supone que el tipo EMP incluye un atributo llamado SALARY, el ejemplo siguiente define un cuerpo de método.

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP
  RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

Información relacionada:

- "CREATE TABLE" en la página 341
- "CREATE TYPE (Estructurado)" en la página 433
- "CREATE METHOD" en la página 289
- "Tipos definidos por el usuario" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)"

ALTER USER MAPPING

La sentencia ALTER USER MAPPING se utiliza para cambiar el ID de autorización o la contraseña que se utilizan en una fuente de datos para el ID de autorización de un servidor federado especificado.

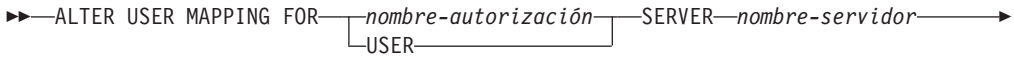
Invocación:

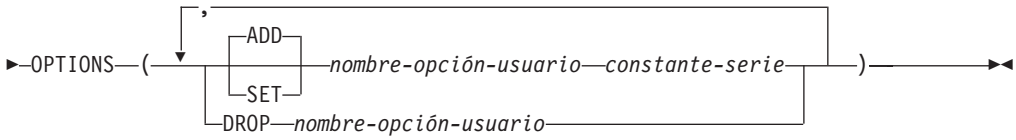
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

Si el ID de autorización de la sentencia es diferente del nombre de autorización que se correlaciona con la fuente de datos, el ID de autorización de la sentencia debe incluir la autorización SYSADM o DBADM. De lo contrario, si el ID de autorización y el nombre de autorización coinciden, no es necesario ningún privilegio ni ninguna autorización.

Sintaxis:





Descripción:

nombre-autorización
Especifica el nombre de autorización bajo el cual un usuario o una aplicación se conecta a una base de datos federada.

USER
El valor del registro especial USER. Cuando se especifica USER, el ID de autorización de la sentencia ALTER USER MAPPING se correlacionará con el ID de autorización de la fuente de datos que se ha especificado en la opción de usuario REMOTE_AUTHID.

SERVER *nombre-servidor*
Identifica la fuente de datos que se puede acceder bajo el ID de autorización remoto que se correlaciona con el ID de autorización local que indica *nombre-autorización* o al que USER hace referencia.

OPTIONS
Indica las opciones de usuario que se deben habilitar, restaurar o desactivar para la correlación que se está modificando.

ADD
Habilita una opción de usuario.

SET
Cambia el valor de una opción de usuario.

nombre-opción-usuario
Nombra una opción de usuario que se debe habilitar o restaurar.

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

DROP *nombre-opción-usuario*
Desactiva una opción de usuario.

Notas:

- Una opción de usuario no se puede especificar más de una vez en la misma sentencia ALTER USER MAPPING (SQLSTATE 42853). Cuando se habilita, restaura o desactiva una opción de usuario, no afecta a ninguna otra opción de usuario que se esté utilizando.
- Una sentencia ALTER USER MAPPING de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) si la UOW ya incluye uno de los elementos siguientes:
  - Una sentencia SELECT que hace referencia a un apodo para una tabla o una vista de la fuente de datos que se debe incluir en la correlación
  - Un cursor abierto en un apodo para una tabla o una vista de la fuente de datos que se debe incluir en la correlación
  - Una sentencia INSERT, DELETE o UPDATE emitida para un apodo de una tabla o una vista de la fuente de datos que se debe incluir en la correlación.

Ejemplos:

Ejemplo 1: Jim utiliza una base de datos local para conectarse a una fuente de datos Oracle llamada ORACLE1. Accede a la fuente de datos local bajo el ID de autorización KLEEWEIN; KLEEWEIN se correlaciona con CORONA, el ID de autorización bajo el cual accede a ORACLE1. Jim va a iniciar el acceso a ORACLE1 bajo un nuevo ID, JIMK. De modo que ahora es necesario correlacionar KLEEWEIN con JIMK.

```
ALTER USER MAPPING FOR KLEEWEIN
  SERVER ORACLE1
  OPTIONS ( SET REMOTE_AUTHID 'JIMK'  )
```

Ejemplo 2: Mary utiliza una base de datos federada para conectarse a una fuente de datos DB2 Universal Database para z/OS y OS/390 llamada DORADO. Utiliza un ID de autorización para acceder a DB2 y otro para acceder a DORADO y ha creado una correlación entre los dos ID. Ha utilizado la misma contraseña con ambos ID, pero ahora decide utilizar una contraseña diferente, ZNYQ, con el ID para DORADO. De acuerdo a ello, necesita correlacionar la contraseña de base de datos federada con ZNYQ.

```
ALTER USER MAPPING FOR MARY
  SERVER DORADO
  OPTIONS ( ADD REMOTE_PASSWORD 'ZNYQ'  )
```

Información relacionada:

- "User mapping options for federated systems" en la publicación *Federated Systems Guide*

ALTER VIEW

La sentencia ALTER VIEW modifica una vista existente alterando una columna de tipo de referencia para añadir un ámbito.

Invocación:

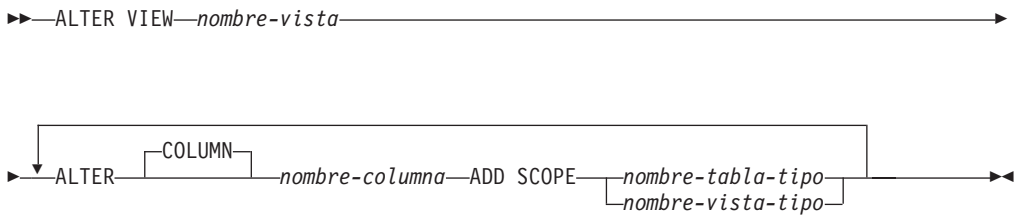
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTERIN para el esquema de la vista
- Definidor de la vista que se va a modificar
- Privilegio CONTROL para la vista que se va a modificar.

Sintaxis:



Descripción:

nombre-vista
Identifica la vista que se va a cambiar. Debe ser una vista descrita en el catálogo.

ALTER COLUMN *nombre-columna*
Es el nombre de la columna que se va a modificar en la vista. El *nombre-columna* debe identificar una columna existente de la vista (SQLSTATE 42703). El nombre no puede estar calificado.

ADD SCOPE
Añade un ámbito a una columna de tipo de referencia existente que todavía no tiene un ámbito definido (SQLSTATE 428DK). La columna no debe ser herencia de una supervista (SQLSTATE 428DJ).

nombre-tabla-tipo
El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No

se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

ALTER WRAPPER

La sentencia ALTER WRAPPER se utiliza para actualizar las propiedades de un reiniciador.

Invocación:

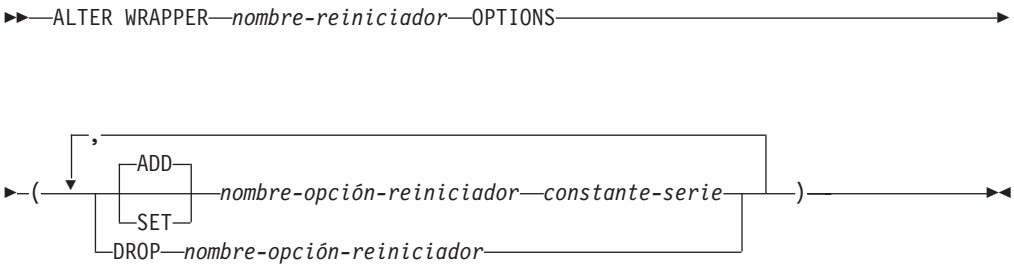
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM

Sintaxis:



Descripción:

nombre-reiniciador
Especifica el nombre del reiniciador.

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

ADD
Habilita una opción de servidor.

SET
Cambia el valor de una opción de reiniciador.

nombre-opción-reiniciador
Nombra una opción de reiniciador que se debe habilitar o restaurar. Actualmente el único nombre de opción de reiniciador soportado es DB2_FENCED.

constante-serie
Especifica el valor para *nombre-opción-reiniciador* como una constante de serie de caracteres. Los valores válidos son 'Y' o 'N'. El valor por omisión para los reiniciadores relacionales es 'N', y el valor por omisión para los reiniciadores no relacionales es 'Y'.

DROP *nombre-opción-reiniciador*
Descarta una opción de reiniciador.

Notas:

- La ejecución de la sentencia ALTER WRAPPER no incluye la comprobación de validez de las opciones específicas de reiniciador.

Ejemplos:

Ejemplo 1: Active la opción DB2_FENCED para el reiniciador SQLNET.

```
ALTER WRAPPER SQLNET OPTIONS (SET DB2_FENCED 'Y')
```

Información relacionada:

- "CREATE WRAPPER" en la página 481

ASSOCIATE LOCATORS

La sentencia ASSOCIATE LOCATORS obtiene el valor localizador de cada conjunto de resultados devuelto por un procedimiento almacenado.

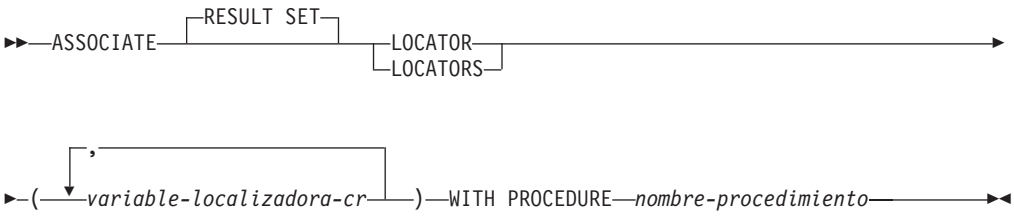
Invocación:

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

Autorización:

No se necesita.

Sintaxis:



Descripción:

variable-localizadora-cr
Especifica una variable localizadora de conjuntos de resultados que se ha declarado en una sentencia compuesta.

WITH PROCEDURE
Identifica el procedimiento almacenado que devuelve localizadores de conjuntos de resultados de acuerdo con el nombre de procedimiento especificado.

nombre-procedimiento
Un nombre de procedimiento es un nombre calificado o no calificado. Cada parte del nombre debe estar formada por caracteres SBCS.

Un nombre de procedimiento totalmente calificado consta de dos partes. La primera parte es un identificador que contiene el nombre de esquema del procedimiento almacenado. La última parte es un identificador que contiene el nombre del procedimiento almacenado. Las dos partes deben estar separadas por un punto. Cualquiera de las partes o ambas puede ser un identificador delimitado.

Si el nombre de procedimiento no está calificado, consta de un solo nombre, pues el nombre de esquema implícito no se añade como calificador al nombre del procedimiento. Para que la sentencia ASSOCIATE LOCATOR se ejecute satisfactoriamente sólo es necesario que el nombre de procedimiento no calificado contenido en la sentencia sea el mismo que el nombre de procedimiento contenido en la sentencia CALL ejecutada más recientemente y que se especificó con un nombre de procedimiento no calificado. Cuando se comparan los nombres, no se tiene en cuenta el nombre de esquema implícito del nombre no calificado contenido en la sentencia CALL. A continuación se describen las normas para especificar un nombre de procedimiento.

Cuando se ejecuta la sentencia ASSOCIATE LOCATORS, el nombre o especificación del procedimiento debe identificar un procedimiento almacenado que el peticionario ya ha invocado utilizando la sentencia CALL. El nombre de procedimiento ASSOCIATE LOCATORS se debe especificar de la misma manera que se especificó en la sentencia CALL. Por ejemplo, si en la sentencia CALL se especificó un nombre de dos partes, se debe utilizar un nombre de dos partes en la sentencia ASSOCIATE LOCATORS.

Normas:

- Se puede asignar más de un localizador a un conjunto de resultados. Se puede emitir una misma sentencia ASSOCIATE LOCATORS más de una vez con diferentes variables localizadoras de conjuntos de resultados.
- Si el número de variables localizadoras de conjuntos de resultados que aparecen en la sentencia ASSOCIATE LOCATORS es menor que el número de localizadores devueltos por el procedimiento almacenado, todas las variables de la sentencia se asignan a un valor, y se emite un aviso.
- Si el número de variables localizadoras de conjuntos de resultados que aparecen en la sentencia ASSOCIATE LOCATORS es mayor que el número de localizadores devueltos por el procedimiento almacenado, se asigna el valor 0 a las variables sobrantes.
- Si un mismo llamador invoca un procedimiento almacenado más de una vez, sólo son accesibles los conjuntos de resultados más recientes.

Ejemplos:

En los ejemplos siguientes se da por supuesto que las sentencias utilizadas están intercaladas en procedimientos SQL.

Ejemplo 1: Utilice las variables localizadoras de conjuntos de resultados LOC1 y LOC2 para obtener los valores de los dos conjuntos de resultados devueltos por el procedimiento almacenado P1. Se supone que el procedimiento almacenado se invoca utilizando un nombre que consta de dos partes.

```
CALL P1;
  ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
     WITH PROCEDURE P1;
```

Ejemplo 2: Repita el supuesto del Ejemplo 1, pero utilice un nombre de dos partes para especificar un nombre de esquema explícito y asegurar que se utilice el procedimiento almacenado P1 del esquema MYSCHEMA.

```
CALL MYSCHEMA.P1;
  ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
     WITH PROCEDURE MYSCHEMA.P1;
```

BEGIN DECLARE SECTION

La sentencia BEGIN DECLARE SECTION marca el principio de una sección de declaración de variables del lenguaje principal.

Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. No debe especificarse en REXX.

Autorización:

No se necesita.

Sintaxis:

Descripción:
La sentencia BEGIN DECLARE SECTION puede codificarse en el programa de aplicación siempre que puedan aparecer declaraciones de variables de acuerdo a las normas del lenguaje principal. Se utiliza para indicar el inicio de una sección de declaración de variables del lenguaje principal. Una sección de variables del lenguaje principal finaliza con una sentencia END DECLARE SECTION.
Normas:

- Las sentencias BEGIN DECLARE SECTION y END DECLARE SECTION deben especificarse por pares y no pueden anidarse.
- No pueden incorporarse sentencias de SQL dentro de la sección de declaración.
- Las variables a las que sentencias de SQL hagan referencia deben declararse en una sección de declaración en todos los lenguajes principales que no sean REXX. Además, la sección debe aparecer antes que la primera referencia a la variable. Por lo general, las variables del lenguaje principal no se declaran en REXX, excepto los localizadores de LOB y las variables de referencia a archivos. En este caso, no se declaran dentro de una BEGIN DECLARE SECTION.
- Las variables declaradas fuera de una sección de declaración no deben tener el mismo nombre que las variables declaradas dentro de una sección de declaración.
- El tipo de datos y la longitud de los tipos de datos LOB deben ir precedidos por las palabras clave SQL TYPE IS.

Ejemplos:

Ejemplo 1: Defina las variables del lenguaje principal hv_smint (smallint), hv_vchar24 (varchar(24)), hv_double (double), hv_blob_50k (blob(51200)), hv_struct (del tipo estructurado ″struct_type″ como blob(10240)) en un programa escrito en C.

```
EXEC SQ  BEGIN DECLARE SECTION;
  short  hv_smint;
  struct {
  short hv_vchar24_len;
  char  hv_vchar24_value[24];
 }                              hv_vchar24;
```

```
  double hv_double;
  SQL TYPE IS BLOB(50K) hv_blob_50k;
  SQL TYPE IS struct_type AS BLOB(10k) hv_struct;
EXEC SQL  END DECLARE SECTION;
```

Ejemplo 2: Defina las variables del lenguaje principal HV-SMINT (smallint), HV-VCHAR24 (varchar(24)), HV-DEC72 (dec(7,2)) y HV-BLOB-50k (blob(51200)) en un programa COBOL.

```
WORKING-STORAGE SECTION.
      EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
01  HV-SMINT              PIC S9(4)       COMP-4.
01  HV-VCHAR24.
    49 HV-VCHAR24-LENGTH  PIC S9(4)       COMP-4.
    49 HV-VCHAR24-VALUE   PIC X(24).
01  HV-DEC72              PIC S9(5)V9(2)  COMP-3.
01 HV-BLOB-50K            USAGE SQL TYPE IS BLOB(50K).
      EXEC SQL END DECLARE SECTION  END-EXEC.
```

Ejemplo 3: Defina las variables del lenguaje principal HVSMINT (smallint), HVVCHAR24 (char(24)), HVDOUBLE (double) y HVBLOB50k (blob(51200)) en un programa Fortran.

```
EXEC SQL  BEGIN DECLARE SECTION
  INTEGER*2      HVSMINT
  CHARACTER*24   HVVCHAR24
  REAL*8         HVDOUBLE
  SQL TYPE IS BLOB(50K)  HVBLOB50K
EXEC SQL  END DECLARE SECTION
```

Nota: En Fortran, si el valor esperado es mayor que 254 caracteres, debe utilizarse una variable del lenguaje principal CLOB.

Ejemplo 4: Defina las variables del lenguaje principal HVSMINT (smallint), HVBLOB50K (blob(51200)) y HVCLOBLOC (un localizador de CLOB) en un programa REXX.

```
   DECLARE :HVCLOBLOC LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'
```

Observe que las variables HVSMINT y HVBLOB50K se han definido de manera implícita al utilizarlas en la sentencia FETCH.

Información relacionada:

- "END DECLARE SECTION" en la página 539

Ejemplos relacionados:

- "advsql.sqb -- How to read table data using CASE (MF COBOL)"
- "dtlob.sqc -- How to use the LOB data type (C)"
- "spclient.sqc -- Call various stored procedures (C)"
- "tut_read.sqc -- How to read tables (C)"
- "udfemsrv.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "spclient.sqC -- Call various stored procedures (C++)"
- "tut_read.sqC -- How to read tables (C++)"
- "udfemsrv.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"

CALL

La sentencia CALL llama a un procedimiento.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

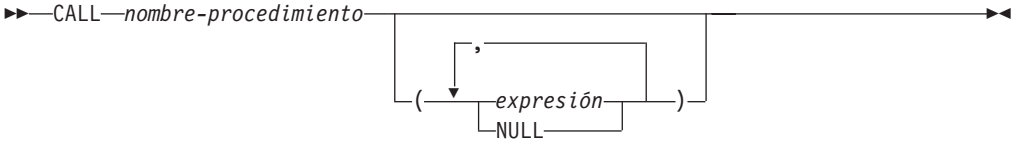
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio EXECUTE para el procedimiento
- Autorización SYSADM o DBADM

Si existe un procedimiento coincidente para el que el ID de autorización de la sentencia no tiene autorización de ejecución, se devolverá un error (SQLSTATE 42501).

Sintaxis:



Descripción:

nombre-procedimiento
Especifica el procedimiento que va a llamarse. Debe ser un procedimiento que esté descrito en el catálogo. El procedimiento específico que debe invocarse se elige mediante la utilización de la resolución de procedimiento. (Para obtener más detalles, consulte el apartado "Notas" de esta sentencia.)

expresión **o NULL**
Cada especificación de *expresión* o NULL es un argumento de CALL. El argumento número *n* de la sentencia CALL corresponde al parámetro número *n* que se ha definido en la sentencia CREATE PROCEDURE para el procedimiento.

Cada argumento de CALL debe ser compatible con el correspondiente parámetro de la definición del procedimiento, como se indica a continuación:

- Parámetro IN
  - El argumento debe poder asignarse al parámetro.
  - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de almacenamiento.
- Parámetro OUT
  - El argumento debe ser una única variable o marcador de parámetro (SQLSTATE 42886).
  - El argumento debe poder asignarse al parámetro.
  - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de recuperación.

- Parámetro INOUT
  - El argumento debe ser una única variable o marcador de parámetro (SQLSTATE 42886).
  - El argumento debe poder asignarse al parámetro.
  - La asignación de un argumento de serie de caracteres utiliza las normas de asignación de almacenamiento y, a su vez, las normas de asignación de recuperación.

Notas:

- ***Signaturas de procedimiento:***

  Un procedimiento se identifica por su esquema, un nombre de procedimiento y el número de parámetros. Esto se denomina signatura de procedimiento, que debe ser exclusiva dentro de la base de datos. Puede haber más de un procedimiento con el mismo nombre en un esquema, siempre que el número de parámetros sea distinto para cada procedimiento.
- ***Vía de acceso de SQL:***

  Un procedimiento puede invocarse haciendo referencia a un nombre calificado (nombre de esquema y de procedimiento), seguido de una lista opcional de argumentos especificados entre paréntesis. Un procedimiento también puede invocarse sin el nombre de esquema, lo que da como resultado una elección de posibles procedimientos de distintos esquemas que tienen el mismo número de parámetros. En este caso, la vía de acceso de SQL se utiliza como ayuda para la resolución del procedimiento. La vía de acceso de SQL es una lista de esquemas en la que se realizan búsquedas para identificar un procedimiento que tiene el mismo nombre y número de parámetros. Para las sentencias CALL estáticas, la vía de acceso de SQL se especifica utilizando la opción de enlace FUNCPATH. Para las sentencias CALL dinámicas, la vía de acceso de SQL es el valor del registro especial CURRENT PATH.
- ***Resolución de procedimiento:***

  En una invocación de procedimiento determinada, el gestor de bases de datos debe decidir a cuál de los posibles procedimientos que tienen el mismo nombre debe llamar. La resolución de procedimiento se realiza siguiendo los pasos que se indican a continuación.

  1. Buscar todos los procedimientos del catálogo (SYSCAT.ROUTINES) de modo que se cumplan todas las condiciones siguientes:
     - En las invocaciones en las que se ha especificado el nombre de esquema (es decir, las referencias calificadas), el nombre del esquema y el nombre del procedimiento coinciden con el nombre de la invocación.
     - En las invocaciones en las que no se ha especificado el nombre de esquema (es decir, las referencias no calificadas), el nombre del procedimiento coincide con el nombre de la invocación y tiene un nombre de esquema que coincide con uno de los esquemas de la vía de acceso de SQL.
     - El número de parámetros definidos debe coincidir con la invocación.
     - El usuario que realiza la invocación tiene el privilegio EXECUTE para el procedimiento.
  2. Elija el procedimiento cuyo esquema aparece en primer lugar en la vía de acceso de SQL.

  Si después de paso 2 no queda ningún procedimiento candidato, se devolverá un error (SQLSTATE 42884).
- ***Recuperación de RETURN_STATUS de un procedimiento de SQL:***

Si un procedimiento SQL emite satisfactoriamente una sentencia RETURN junto con un valor de estado, este valor se coloca en el primer campo SQLERRD de la SQLCA. Si la sentencia CALL se emite en un procedimiento SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de estado de retorno (RETURN_STATUS). El valor es –1 cuando SQLSTATE indica un error. El valor es 0 si no se ha devuelto ningún error y si la sentencia RETURN no se ha especificado en el procedimiento.

- ***Devolución de conjuntos de resultados de los procedimientos:***

  Si el programa que llama se ha escrito utilizando CLI, JDBC o SQLJ, o si el que llama es un procedimiento SQL, los conjuntos de resultados se devuelven directamente al procedimiento o programa que llama. 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 de resultados al proceso que llama o (para los cursores WITH RETURN TO CLIENT) directamente al cliente.
  – Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto de resultados de un cursor tiene 500 filas y el procedimiento ha leído 150 de éstas al término del procedimiento, las filas 151 a 500 se devolverá al proceso o a la aplicación que llama (según proceda).

  Si el procedimiento se ha invocado desde la CLI o desde JDBC y se ha dejado abierto más de un cursor, los conjuntos de resultados sólo pueden procesarse en el orden en que se han abierto los cursores.

- ***Mejora del rendimiento:***

  Los valores de todos los argumentos se pasan desde la aplicación al procedimiento. Para mejorar el rendimiento de esta operación, las variables del lenguaje principal que corresponden a los parámetros OUT y que tienen longitudes de más de simplemente algunos bytes deben establecerse en NULL antes de ejecutarse la sentencia CALL.

- ***Anidamiento de las sentencias CALL:***

  Los procedimientos pueden llamarse desde las rutinas y también desde los programas de aplicación. Cuando un procedimiento se llama desde una rutina, se considera que la llamada está anidada.

  Si un procedimiento devuelve algún conjunto de resultados de la consulta, los conjuntos de resultados se devuelven como se indica a continuación:

  – Los conjuntos de resultados de RETURN TO CALLER sólo están visibles para el programa que se encuentran en el nivel de anidamiento anterior.
  – Los conjuntos de resultados de RETURN TO CLIENT sólo están visibles si el procedimiento se ha invocado desde un conjunto de procedimientos anidados. Si una función o un método tiene lugar en cualquier punto de la cadena de la llamada, el conjunto de resultados no está visible. Si el conjunto de resultados está visible, sólo está visible para la aplicación cliente que ha realizado la llamada de procedimiento inicial.

  Consideremos el ejemplo siguiente:

```
Programa cliente:
EXEC SQL CALL PROCA;

  PROCA:
  EXEC SQL CALL PROCB;

    PROCB:
```

```
EXEC SQL DECLARE B1 CURSOR WITH RETURN TO CLIENT ...;
EXEC SQL DECLARE B2 CURSOR WITH RETURN TO CALLER ...;
EXEC SQL DECLARE B3 CURSOR FOR SELECT UDFA FROM T1;

  UDFA:
  EXEC SQL CALL PROCC;

    PROCC:
    EXEC SQL DECLARE C1 CURSOR WITH RETURN TO CLIENT ...;
    EXEC SQL DECLARE C2 CURSOR WITH RETURN TO CALLER ...;
```

En el procedimiento PROCB:

- El cursor B1 está visible en la aplicación cliente, pero no está visible en el procedimiento PROCA.
- El cursor B2 está visible en PROCA, pero no está visible para el cliente.

En el procedimiento PROCC:

- El cursor C1 no está visible para UDFA ni para la aplicación cliente. (Puesto que UDFA aparece en la cadena de la llamada entre el cliente y PROCC, el conjunto de resultados no se devuelve al cliente.)
- El cursor C2 está visible en UDFA, pero no está visible para ninguno de los procedimientos superiores.

- ***Anidamiento de procedimientos en activadores, sentencias compuestas dinámicas, funciones o métodos:***

  Cuando se llama a un procedimiento en un activador, una sentencia compuesta dinámica, una función o un método:

  - El procedimiento no debe emitir las sentencias COMMIT ni ROLLBACK.
  - No se puede acceder a los conjuntos de resultados que devuelve el procedimiento.
  - Si el procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del mismo podrá acceder a la tabla que la sentencia que ha invocado el procedimiento está modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del mismo podrá modificar la tabla que la sentencia que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

  Cuando se llama a un procedimiento desde una función o un método:

  - El procedimiento tiene las mismas restricciones de acceso a las tablas que la función o el método que realiza la invocación.
  - Los puntos de salvaguarda que se han definido antes de invocarse la función o el método no estarán visibles para el procedimiento, y los puntos de salvaguarda que se han definido dentro del procedimiento no estarán visibles fuera de la función o del método.
  - No se puede acceder desde el cliente a los conjuntos de resultados RETURN TO CLIENT que devuelve el procedimiento.

- ***Compatibilidades:***
  - Existe una forma antigua de la sentencia CALL que puede incorporarse a las aplicaciones por medio de la precompilación de la aplicación con la opción CALL_RESOLUTION DEFERRED. Esta opción no está disponible para los procedimientos de SQL.

Ejemplos:

Ejemplo 1:

Se define un procedimiento de Java 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 'parts!onhand'
    LANGUAGE JAVA
                PARAMETER STYLE DB2GENERAL;
```

Una aplicación de 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, hvPartnum);
 stpCall.setBigDecimal(2, hvCost);
 stpCall.setInt(3, hvQuantity);

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

 hvCost = stpCall.getBigDecimal(2);
 hvQuantity = stpCall.getInt(3);
 ...
```

Este fragmento de código de aplicación invocará al método de Java `onhand` de la clase `parts`, ya que el nombre de procedimiento que se ha especificado en la sentencia CALL se encuentra en la base de datos y tiene el nombre externo `parts!onhand`.

Ejemplo 2:

Existen seis procedimientos FOO, en cuatro esquemas distintos, registrados de la forma siguiente (observe que no aparecen todas las palabras clave necesarias):

```
CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...
```

La referencia de procedimiento es la siguiente (donde I1 y I2 son valores INTEGER):

```
CALL FOO(I1, I2)
```

Suponga que la aplicación que efectúa esta referencia tiene una vía de acceso de SQL establecida como:

```
 "JULIUS","AUGUSTUS","CAESAR"
```

De acuerdo con el algoritmo...

El procedimiento que tiene el nombre específico FOO_6 se elimina como candidato, ya que el esquema ″NERO″ no se ha incluido en la vía de acceso de SQL. FOO_1, FOO_3 y FOO_4 se eliminan como candidatos, pues tienen un número incorrecto

de parámetros. Se considera que los restantes candidatos son correctos, tal como determina la vía de acceso de SQL. Observe que los tipos de los argumentos y parámetros se pasan por alto. Los parámetros de FOO_5 coinciden exactamente con los argumentos de CALL, pero se ha elegido FOO_2 porque ″AUGUSTUS″ aparece antes que ″CAESAR″ en la vía de acceso de SQL.

Información relacionada:

- “GET DIAGNOSTICS” en la página 565
- “Registro especial CURRENT PATH” en la publicación *Consulta de SQL, Volumen 1*
- “CALL invocada desde una sentencia compilada” en la publicación *Consulta de SQL, Volumen 1*
- “Asignaciones y comparaciones” en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- “outcli.sqb -- Call stored procedures using the SQLDA structure (MF COBOL)”
- “spclient.c -- Call various stored procedures”
- “spclient.sqc -- Call various stored procedures (C)”
- “spclient.sqC -- Call various stored procedures (C++)”
- “SpClient.sqlj -- Call a variety of types of stored procedures from SpServer.sqlj (SQLj)”

CASE

La sentencia CASE selecciona una vía de ejecución de acuerdo con varias condiciones. Esta sentencia no se debe confundir con la expresión CASE, que permite seleccionar una expresión basándose en la evaluación de una o varias condiciones.

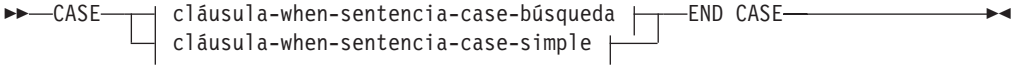
Invocación:

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

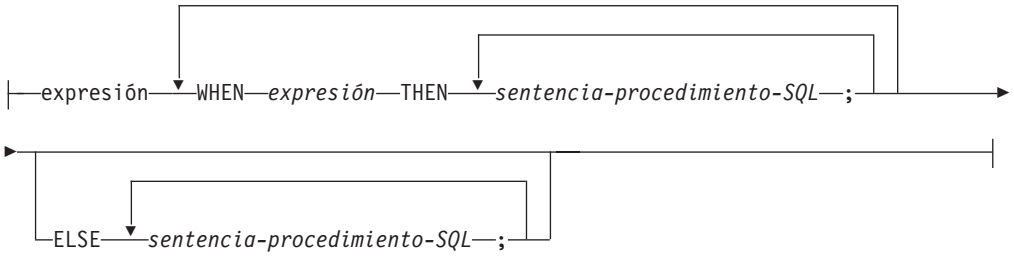
Autorización:

No se requieren privilegios para invocar una sentencia CASE. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL y las expresiones incorporadas en la sentencia CASE.

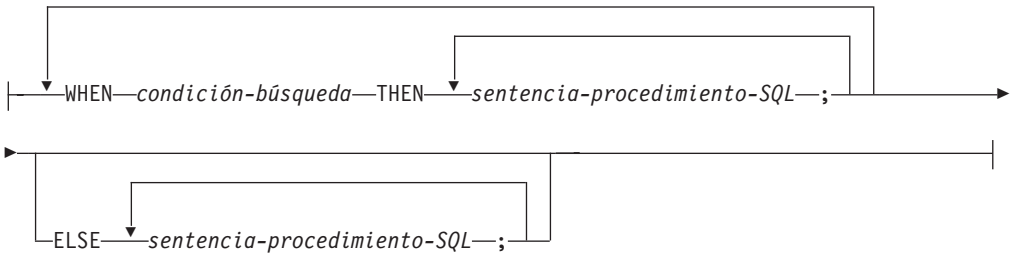
Sintaxis:



cláusula-when-sentencia-case-simple:



cláusula-when-sentencia-case-búsqueda:



Descripción:

CASE
Empieza una *sentencia-case*.

cláusula-when-sentencia-case-simple
El valor de la *expresión* anterior a la primera palabra clave WHEN se comprueba si es igual al valor de cada *expresión* que sigue a la palabra clave WHEN. Si se cumple la condición de búsqueda, se ejecuta la sentencia THEN. Si el resultado es desconocido o falso, el proceso continúa en la siguiente condición de búsqueda. Si el resultado no coincide con ninguna de las condiciones de búsqueda y existe una cláusula ELSE, se procesan las sentencias de la cláusula ELSE.

cláusula-when-sentencia-case-búsqueda
Se evalúa la *condición-búsqueda* que sigue a la palabra clave WHEN. Si su evaluación da un resultado verdadero, se procesan las sentencias de la cláusula THEN asociada. Si su evaluación da un resultado falso o desconocido, se evalúa la siguiente *condición-búsqueda*. Si ninguna *condición-búsqueda* devuelve un resultado verdadero y existe una cláusula ELSE, se procesan las sentencias de la cláusula ELSE.

sentencia-procedimiento-SQL
Especifica una sentencia que se debe invocar. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

END CASE
Finaliza una *sentencia-case*.

Notas:

- Si ninguna de las condiciones especificadas en la cláusula WHEN devuelve un resultado verdadero y no hay una cláusula ELSE especificada, se emite un error durante la ejecución y se interrumpe la ejecución de la sentencia CASE (SQLSTATE 20000).
- Asegúrese de que la sentencia CASE abarca todas las condiciones de ejecución posibles.

Ejemplos:

En función del valor de la variable de SQL `v_workdept`, actualice la columna DEPTNAME de la tabla DEPARTMENT con el nombre adecuado.

El ejemplo siguiente muestra cómo hacer esto utilizando la sintaxis de una *cláusula-when-sentencia-case-simple*:

```
CASE v_workdept
     WHEN'A00'
       THEN UPDATE department
       SET deptname = 'DATA ACCESS 1';
     WHEN 'B01'
       THEN UPDATE department
       SET deptname = 'DATA ACCESS 2';
     ELSE UPDATE department
       SET deptname = 'DATA ACCESS 3';
END CASE
```

El ejemplo siguiente muestra cómo hacer esto utilizando la sintaxis de una *cláusula-when-sentencia-case-búsqueda*:

```
         CASE
  WHEN v_workdept = 'A00'
       THEN UPDATE department
```

```
        SET deptname = 'DATA ACCESS 1';
    WHEN v_workdept = 'B01'
        THEN UPDATE department
        SET deptname = 'DATA ACCESS 2';
      ELSE UPDATE department
        SET deptname = 'DATA ACCESS 3';
END CASE
```

Información relacionada:

- "Expresiones" en la publicación *Consulta de SQL, Volumen 1*
- "SQL compuesto (procedimiento)" en la página 139

Ejemplos relacionados:

- "advsql.sqb -- How to read table data using CASE (MF COBOL)"
- "tbtrig.sqc -- How to use a trigger on a table (C)"
- "tbtrig.sqC -- How to use a trigger on a table (C++)"
- "TbTrig.java -- How to use triggers (JDBC)"
- "TbTrig.sqlj -- How to use triggers (SQLj)"

CLOSE

La sentencia CLOSE cierra un cursor. Si se ha creado una tabla de resultados cuando se ha abierto el cursor, se ha destruido esa tabla.

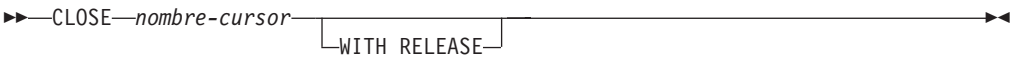
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita. Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

Sintaxis:



Descripción:

nombre-cursor
Identifica el cursor que se va a cerrar. El *nombre-cursor* debe identificar un cursor declarado tal como se explica en la sentencia DECLARE CURSOR. Cuando se ejecuta la sentencia CLOSE, el cursor debe estar en el estado abierto.

WITH RELEASE
Se intenta liberar todos los bloqueos que se han mantenido para el cursor. Tenga en cuenta que no se liberan necesariamente todos los bloqueos; se pueden conservar bloqueos para otras operaciones o actividades.

Notas:

- Al final de una unidad de trabajo, todos los cursores que pertenecen a un proceso de aplicación y que se han declarado sin la opción WITH HOLD se cierran de manera implícita.
- La cláusula WITH RELEASE no tiene ningún efecto al cerrar los cursores que se han definido en las funciones o en los métodos. La cláusula tampoco tiene ningún efecto al cerrarse los cursores que se han definido en los procedimientos almacenados que se han llamado desde las funciones o los métodos.
- La cláusula WITH RELEASE no tiene ningún efecto para los cursores que están funcionando bajo los niveles de aislamiento CS o UR. Cuando se especifica para cursores que están funcionando bajo los niveles de aislamiento RS o RR, WITH RELEASE termina algunas de las garantías de dichos niveles de aislamiento. De forma específica, si se vuelve a abrir el cursor, un cursor RS puede experimentar el fenómeno de 'lectura no repetible', y un cursor RR puede experimentar el fenómeno de 'lectura no repetible' y el de 'fantasma'.

  Si un cursor que originalmente era RR o RS se vuelve a abrir tras haberse cerrado utilizando la cláusula WITH RELEASE, adquirirá nuevos bloqueos.
- Se aplican normas especiales a los cursores dentro de un procedimiento almacenado que no se haya cerrado antes de volver al programa que lo llama.
- Mientras un cursor está abierto (es decir, todavía no se ha cerrado), los cambios realizados en los valores de secuencia como resultado de las sentencias que

implican a ese cursor (por ejemplo FETCH o UPDATE utilizando el cursor que incluye una expresión NEXT VALUE para una secuencia) no darán como resultado una actualización en PREVIOUS VALUE para esas secuencias desde el punto de vista del cursor. Los valores PREVIOUS VALUE de esas secuencias afectadas se actualizarán cuando el cursor se cierre explícitamente con la sentencia CLOSE. En un entorno de base de datos particionado, si un cursor se cierra implícitamente mediante una operación de confirmar o retrotraer, puede que no se actualice PREVIOUS VALUE con el valor generado más recientemente para la secuencia.

Ejemplo:

Se utiliza un cursor para buscar una fila cada vez en las variables del programa C `dnum`, `dname` y `mnum`. Finalmente, se cierra el cursor. Si se vuelve a abrir el cursor, se sitúa de nuevo al principio de las filas en las que se debe buscar.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO
  FROM TDEPT
  WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {                                              .
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
     .
     .
}
EXEC SQL CLOSE C1;
```

Información relacionada:

- “CALL” en la página 107
- “DECLARE CURSOR” en la página 483

Ejemplos relacionados:

- “dynamic.sqb -- How to update table data with cursor dynamically (MF COBOL)”
- “tut_mod.sqc -- How to modify table data (C)”
- “tut_read.sqc -- How to read tables (C)”
- “tut_mod.sqC -- How to modify table data (C++)”
- “tut_read.sqC -- How to read tables (C++)”

COMMENT

La sentencia COMMENT añade o sustituye comentarios en las descripciones del catálogo de diversos objetos.

Invocación:

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

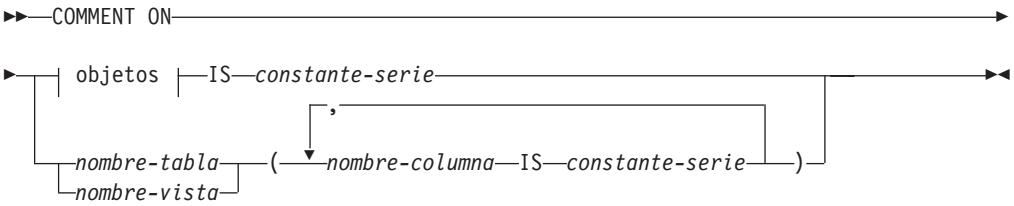
Autorización:

Los privilegios que debe tener el ID de autorización de la sentencia COMMENT deben incluir uno de los siguientes:

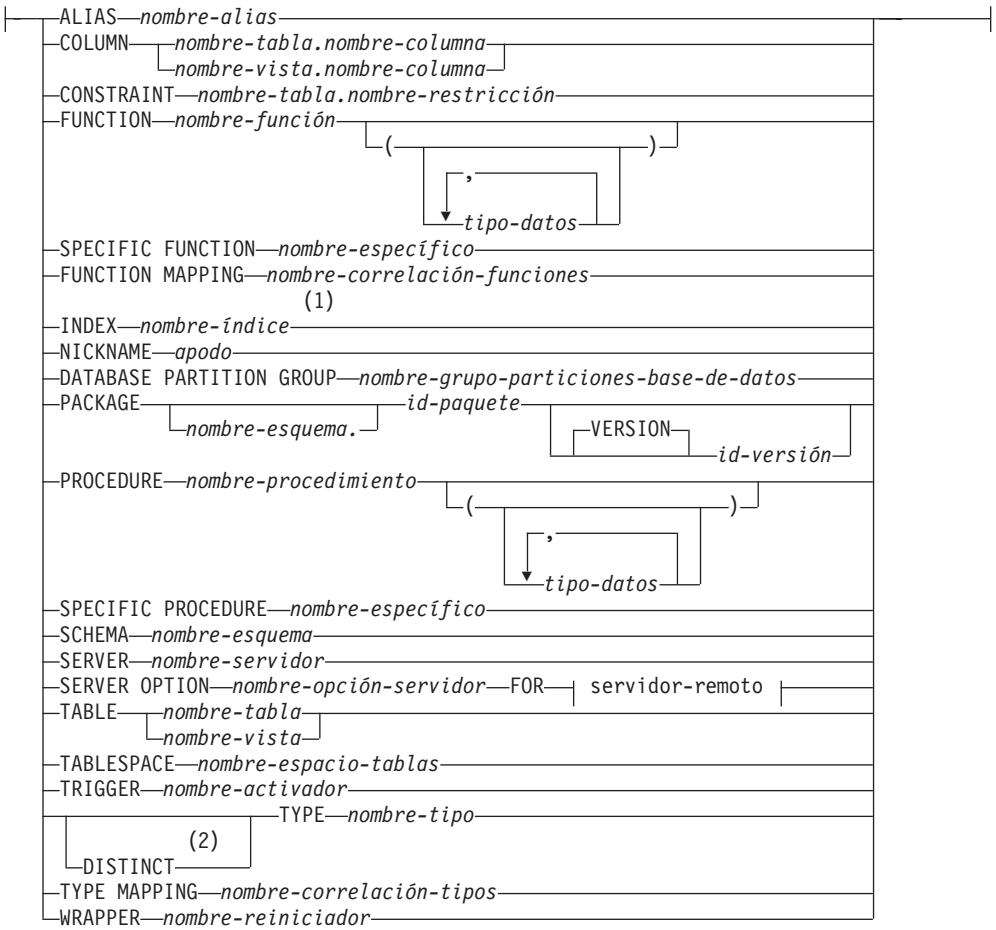
- SYSADM o DBADM
- definidor del objeto (tabla principal para la columna o restricción) tal como está registrado en la columna DEFINER de la vista de catálogo para el objeto (columna OWNER para un esquema)
- Privilegio ALTERIN para el esquema (sólo aplicable a los objetos que permiten nombres de más de una parte)
- Privilegio CONTROL para el objeto (sólo aplicable a los objetos de índice, paquete, tabla y vista)
- Privilegio ALTER para el objeto (sólo aplicable a los objetos de tabla)

Tenga en cuenta que para el espacio de tablas o el grupo de particiones de base de datos, el ID de autorización debe tener autorización SYSADM o SYSCTRL.

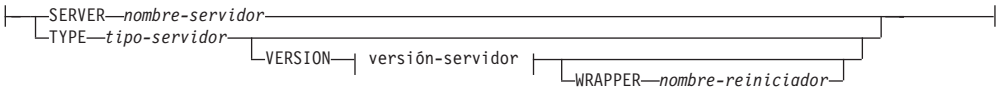
Sintaxis:



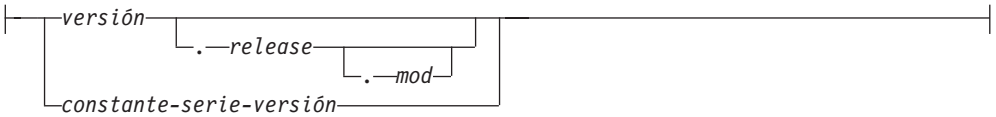
objetos:



servidor-remoto:



versión-servidor:



Notas:

1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.

2 La palabra clave DATA se puede utilizar como sinónimo de DISTINCT.

Descripción:

ALIAS *nombre-alias*
Indica un comentario que se añadirá o sustituirá para un alias. El *nombre-alias* debe designar un alias descrito en el catálogo (SQLSTATE 42704). El

comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES para la fila que describe el alias.

COLUMN *nombre-tabla.nombre-columna* o *nombre-vista.nombre-columna*
Indica un comentario que se añadirá o se sustituirá para una columna. La combinación *nombre-tabla.nombre-columna nombre-vista.nombre-columna* debe identificar una combinación de columna y tabla que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.COLUMNS para la fila que describe la columna.

No puede realizarse un comentario sobre una columna de una vista no operativa. (SQLSTATE 51024).

CONSTRAINT *nombre-tabla.nombre-restricción*
Indica que se añadirá o se sustituirá un comentario para una restricción. La combinación *nombre-tabla.nombre-restricción* debe identificar una restricción y la tabla que restringe; deben estar descritos en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABCONST para la fila que describe la restricción.

FUNCTION
Indica que se añadirá o se sustituirá un comentario para una función. La instancia de función especificada debe ser una función definida por el usuario o una plantilla de función descritas en el catálogo.

Hay varias maneras distintas disponibles para identificar la instancia de función:

FUNCTION *nombre-función*
Identifica la función en particular y sólo es válida si hay exactamente una función con ese *nombre-función*. La función así identificada puede tener cualquier número de parámetros definidos para la misma. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica de la función en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

FUNCTION *nombre-función* **(***tipo-datos***,...)**
Proporciona la signatura de la función, que identifica de manera exclusiva la función que hay que comentar. El algoritmo de selección de función *no* se utiliza.

nombre-función
Proporciona el nombre de la función que hay que comentar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(*tipo-datos***,...)**
Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION en la posición correspondiente. El

número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la función específica para la que se añade o se sustituye el comentario.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n, pues 0 <n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

(Observe que el atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la selección por comparación. Así, por ejemplo, un CHAR FOR BIT DATA especificado en la signatura coincidiría con una función definida sólo con CHAR, y viceversa.)

Si en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-específico*
Indica que se añadirán o se sustituirán comentarios para una función (vea FUNCTION para conocer otros métodos de identificar una función). Identifica la función en particular definida por el usuario que hay que comentar, utilizando el nombre específico que se ha especificado o tomado por omisión en el tiempo de creación de la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

No es posible comentar una función que está en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.ROUTINES de la fila que describe la función.

FUNCTION MAPPING *nombre-correlación-funciones*
Indica que se añadirá o sustituirá un comentario para la correlación de una función. El *nombre-correlación-funciones* debe identificar una correlación de funciones que está descrita en el catálogo (SQLSTATE 42704). El comentario

sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.FUNCMAPPINGS correspondiente a la fila que describe la correlación de funciones.

INDEX *nombre-índice*
Indica que se añadirá o sustituirá un comentario para un índice o especificación de índice. El *nombre-índice* debe designar un índice diferenciado o una especificación de índice que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.INDEXES correspondiente a la fila que describe el índice o especificación de índice.

NICKNAME *apodo*
Indica que se añadirá o sustituirá un comentario para un apodo. El *apodo* debe ser un apodo descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe el apodo.

DATABASE PARTITION GROUP *nombre-grupo-particiones-base-de-datos*
Indica que se añadirá o se sustituirá un comentario para un grupo de particiones de base de datos. El *nombre-grupo-particiones-base-de-datos* debe identificar a un grupo de particiones de base de datos diferenciado que se ha descrito en el catálogo (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.DBPARTITIONGROUPS de la fila que describe al grupo de particiones de base de datos.

PACKAGE *nombre-esquema.id-paquete*
Indica que se añadirá o se sustituirá un comentario para un paquete. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. El nombre de esquema y el ID de paquete, junto con el ID de versión especificado de forma implícita o explícita, deben identificar a un paquete que se ha descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.PACKAGES para la fila que describe el paquete.

VERSION *id-versión*
Identifica en qué versión del paquete va a realizarse el comentario. Si no se especifica un valor, la versión tomará por omisión una serie de caracteres vacía. Si existen varios paquetes con el mismo nombre de paquete pero con distintas versiones, sólo podrá comentarse una versión del paquete en una invocación de la sentencia COMMENT. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

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

PROCEDURE
Indica que se añadirá o sustituirá un comentario para un procedimiento. La instancia del procedimiento especificado debe ser un procedimiento almacenado descrito en el catálogo.

Hay varias maneras disponibles de identificar la instancia de procedimiento:

PROCEDURE *nombre-procedimiento*
Identifica el procedimiento en particular y sólo es válido si hay

exactamente un procedimiento con el *nombre-procedimiento* en el esquema. El procedimiento identificado de esta manera puede tener cualquier número de parámetros definido. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. Si no existe ningún procedimiento con este nombre en el esquema nombrado o implícito, se genera un error (SQLSTATE 42704). Si existe más de una instancia específica del procedimiento en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento* **(***tipo-datos***,...)**
Se utiliza para proporcionar la signatura del método, que identifica de manera exclusiva al procedimiento que se comenta.

nombre-procedimiento
Proporciona el nombre de procedimiento del procedimiento que hay que comentar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(*tipo-datos***,...)**
Deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar el procedimiento específico para el que se añade o se sustituye el comentario.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si la longitud, la precisión y la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE PROCEDURE.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que 0 <n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún procedimiento con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico*
Indica que se sustituirán o añadirán comentarios para un procedimiento (consulte PROCEDURE para ver otros métodos de identificar un procedimiento). Identifica el procedimiento almacenado en particular que

se debe comentar, utilizando el nombre específico que se ha especificado o que ha tomado por omisión en el tiempo de creación del procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

No es posible comentar un procedimiento que está en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.ROUTINES de la fila que describe al procedimiento.

SCHEMA *nombre-esquema*
Indica que se añadirá o sustituirá un comentario para un esquema. El *nombre-esquema* debe identificar un esquema que se describe en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SCHEMATA para la fila que describe el esquema.

SERVER *nombre-servidor*
Indica que se añadirá o sustituirá un comentario para una fuente de datos. El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVERS correspondiente a la fila que describe la fuente de datos.

SERVER OPTION *nombre-opción-servidor* **FOR** *servidor-remoto*
Indica que se añadirá o se sustituirá un comentario para una opción de servidor.

nombre-opción-servidor
Identifica una opción de servidor. Esta opción debe ser una que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVEROPTIONS correspondiente a la fila que describe la opción de servidor.

servidor-remoto
Describe la fuente de datos a la que se aplica la *opción-servidor*.

SERVER *nombre-servidor*
Indica el nombre de la fuente de datos a la que se aplica la *opción-servidor*. El *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

TYPE *tipo-servidor*
Especifica el tipo de fuente de datos—por ejemplo, DB2 Universal Database para OS/390 u Oracle—al que se aplica la *opción-servidor*. El *tipo-servidor* se puede especificar en minúsculas o mayúsculas; se guardará en mayúsculas en el catálogo.

VERSION
Especifica la versión de la fuente de datos identificada por *nombre-servidor*.

versión
Especifica el número de versión. *versión* debe ser un entero.

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

mod
Especifica el número de la modificación del release indicado por *release. mod* debe ser un entero.

constante-serie-versión
Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i'); o puede ser los valores concatenados de *versión*, *release* y, si es aplicable, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*
Identifica el reiniciador que se utiliza para acceder a la fuente de datos referenciada por *nombre-servidor.*

TABLE *nombre-tabla* o *nombre-vista*
Indica que se añadirá o se sustituirá un comentario para una tabla o vista. El *nombre-tabla* o *nombre-vista* debe identificar una tabla o vista (no un alias ni un apodo) que esté descrita en el catálogo (SQLSTATE 42704) y no debe identificar una tabla temporal declarada (SQLSTATE 42995). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe la tabla o vista.

TABLESPACE *nombre-espacio-tablas*
Indica que se añadirá o se sustituirá un comentario para un espacio de tablas. El *nombre-espacio-tablas* debe identificar un espacio de tablas diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TABLESPACES para la fila que describe el espacio de tablas.

TRIGGER *nombre-activador*
Indica que se añadirá o se sustituirá un comentario para un activador. El *nombre-activador* debe identificar un activador diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TRIGGERS para la fila que describe el activador.

TYPE *nombre-tipo*
Indica que se añadirá o se sustituirá un comentario para un tipo definido por el usuario. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704). Si se especifica DISTINCT, *nombre-tipo* debe identificar un tipo diferenciado que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.DATATYPES para la fila que describe el tipo definido por el usuario.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

TYPE MAPPING *nombre-correlación-tipos*
Indica que se añadirá o sustituirá un comentario para una correlación de tipos de datos definida por el usuario. El *nombre-correlación-tipos* debe identificar una

correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TYPEMAPPINGS correspondiente a la fila que describe la correlación.

WRAPPER *nombre-reiniciador*
Indica que se añadirá o sustituirá un comentario para un reiniciador. El *nombre-reiniciador* debe identificar un reiniciador que esté descrito en el catálogo (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WRAPPERS correspondiente a la fila que describe el reiniciador.

IS *constante-serie*
Especifica el comentario que va a añadirse o sustituirse. La *constante-serie* puede ser cualquier constante de serie de caracteres con un máximo de 254 bytes. (El retorno de carro y el salto de línea cuentan como 1 byte cada uno.)

nombre-tabla | *nombre-vista* **(** { *nombre-columna* **IS** *constante-serie* } **...)**
Este formato de la sentencia COMMENT proporciona la posibilidad de especificar comentarios para múltiples columnas de una tabla o vista. Los nombres de columna no deben estar calificados, cada nombre debe identificar una columna de la vista o tabla especificada y la tabla o vista debe estar descrita en el catálogo. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

No puede realizarse un comentario en una columna de una vista no operativa (SQLSTATE 51024).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP

Ejemplos:

Ejemplo 1: Añada un comentario para la tabla EMPLOYEE.

```
COMMENT ON TABLE EMPLOYEE
  IS 'Refleja la reorganización del primer trimestre'
```

Ejemplo 2: Añada un comentario para la vista EMP_VIEW1.

```
COMMENT ON TABLE EMP_VIEW1
  IS 'Vista de la tabla EMPLOYEE sin la información de salarios'
```

Ejemplo 3: Añada un comentario para la columna EDLEVEL de la tabla EMPLOYEE.

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
  IS 'curso más alto aprobado en la escuela'
```

Ejemplo 4: Añada comentarios para dos columnas diferentes de la tabla EMPLOYEE.

```
   COMMENT ON EMPLOYEE
(WORKDEPT IS 'vea los nombres en la tabla DEPARTMENT',
EDLEVEL IS 'curso más alto aprobado en la escuela' )
```

Ejemplo 5: Pellow desea realizar un comentario sobre la función CENTRE, que ha creado en su esquema PELLOW, utilizando la signatura para identificar la función específica que se debe comentar.

```
COMMENT ON FUNCTION CENTRE (INT,FLOAT)
  IS 'func CENTRE de Frank, utiliza método Chebychev'
```

Ejemplo 6: McBride desea realizar un comentario sobre otra función CENTRE, que ella creó en el esquema PELLOW, utilizando el nombre específico para identificar la instancia de función que hay que comentar:

```
COMMENT ON SPECIFIC FUNCTION PELLOW.FOCUS92 IS
  'La función CENTRE con mayor éxito de Louise, utiliza
       la técnica de foco borroso browniana'
```

Ejemplo 7: Comente la función ATOMIC_WEIGHT en el esquema CHEM, donde se sabe que sólo hay una función con ese nombre:

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
  IS 'toma núm. atómico, da peso atómico'
```

Ejemplo 8: Eigler desea realizar un comentario sobre el procedimiento SEARCH, que ha creado en su esquema EIGLER, utilizando la signatura para identificar el procedimiento específico que se debe comentar.

```
COMMENT ON PROCEDURE SEARCH (CHAR,INT)
  IS 'Buscar la masa de Frank y sustituir algoritmo'
```

Ejemplo 9: Macdonald desea realizar un comentario sobre otra función SEARCH, que creó en el esquema EIGLER, utilizando el nombre específico para identificar la instancia de procedimiento que hay que comentar:

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
  'Buscar masa de Patrick y destruir algoritmo'
```

Ejemplo 10: Realice un comentario sobre la función OSMOSIS en el esquema BIOLOGY, donde se sabe que sólo hay un procedimiento con dicho nombre:

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
  IS 'Modelo de cálculos de ósmosis'
```

Ejemplo 11: Realice un comentario sobre una especificación de índice denominada INDEXSPEC.

```
COMMENT ON INDEX INDEXSPEC
  IS 'Una especificación de índice que indica al optimizador
      que la tabla referenciada por el apodo NICK1 tiene un índice.
```

Ejemplo 12: Realice un comentario sobre el reiniciador cuyo nombre por omisión es NET8.

```
COMMENT ON WRAPPER NET8
  IS 'El reiniciador de las fuentes de datos asociadas al
      software cliente Net8 de Oracle.'
```

COMMIT

La sentencia COMMIT termina una unidad de trabajo y confirma los cambios de la base de datos que ha realizado esa unidad de trabajo.

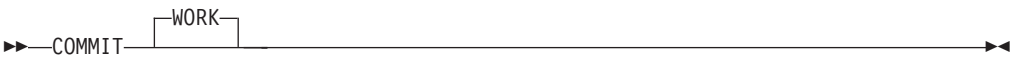
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:
Finaliza la unidad de trabajo en la que se ejecuta la sentencia COMMIT y se inicia una nueva unidad de trabajo. Se confirman todos los cambios realizados por las siguientes sentencias ejecutadas durante la unidad de trabajo: ALTER, COMMENT, CREATE, DROP, GRANT, LOCK TABLE, REVOKE, SET INTEGRITY, SET Variable y las sentencias de cambio de datos (INSERT, DELETE, MERGE, UPDATE), entre ellas las anidadas en una consulta.

Sin embargo, las sentencias siguientes no están bajo el control de la transacción y los cambios que éstas realizan son independientes de la sentencia COMMIT:

- SET CONNECTION
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT LOCK TIMEOUT
- SET CURRENT PACKAGESET
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET EVENT MONITOR STATE
- SET PASSTHRU

  Nota: Aunque la sentencia SET PASSTHRU no está bajo el control de la transacción, la sesión PASSTHRU que ha iniciado la sentencia está bajo el control de la transacción.
- SET PATH
- SET SCHEMA
- SET SERVER OPTION

Todos los bloqueos que ha adquirido la unidad de trabajo con posterioridad a su inicio se liberan, excepto los bloqueos que son necesarios para abrir los cursores que se han declarado como WITH HOLD. Todos los cursores abiertos no definidos

como WITH HOLD se cierran. Los cursores abiertos que se han definido como WITH HOLD siguen abiertos y el cursor se coloca delante de la siguiente fila lógica de la tabla de resultados. (FETCH debe realizarse antes de emitirse una sentencia UPDATE o DELETE con posición.) Se liberan todos los localizadores de LOB. Observe que esto es verdadero incluso cuando los localizadores están asociados a valores LOB recuperados mediante un cursor que tenga la propiedad WITH HOLD.

Se liberan todos los puntos de salvaguarda definidos dentro de la transacción.

Notas:

- Se recomienda encarecidamente que cada proceso de aplicación finalice explícitamente su unidad de trabajo antes de terminar. Si el programa de aplicación finaliza normalmente sin una sentencia COMMIT ni ROLLBACK entonces el gestor de bases de datos intenta una confirmación o retrotracción según el entorno de aplicación.
- Para obtener información acerca del efecto de COMMIT en las sentencias de SQL dinámico colocadas en la antememoria, consulte "EXECUTE".
- Para obtener información acerca de los posibles efectos de COMMIT en las tablas temporales declaradas, consulte "DECLARE GLOBAL TEMPORARY TABLE".

Ejemplo:

Confirme las modificaciones en la base de datos efectuadas desde el último punto de confirmación.

```
COMMIT WORK
```

Información relacionada:

- "EXECUTE" en la página 541
- "DECLARE GLOBAL TEMPORARY TABLE" en la página 489

Ejemplos relacionados:

- "dynamic.sqb -- How to update table data with cursor dynamically (MF COBOL)"
- "tbconstr.sqc -- How to create, use, and drop constraints (C)"
- "tbsavept.sqc -- How to use external savepoints (C)"
- "tut_mod.sqc -- How to modify table data (C)"
- "tut_use.sqc -- How to modify a database (C)"
- "tbconstr.sqC -- How to create, use, and drop constraints (C++)"
- "tut_mod.sqC -- How to modify table data (C++)"
- "tut_use.sqC -- How to modify a database (C++)"

SQL compuesto (Dinámico)

Una sentencia compuesta agrupa otras sentencias en un bloque ejecutable. Se pueden declarar variables de SQL en una sentencia compuesta atómica preparada dinámicamente.

Invocación:

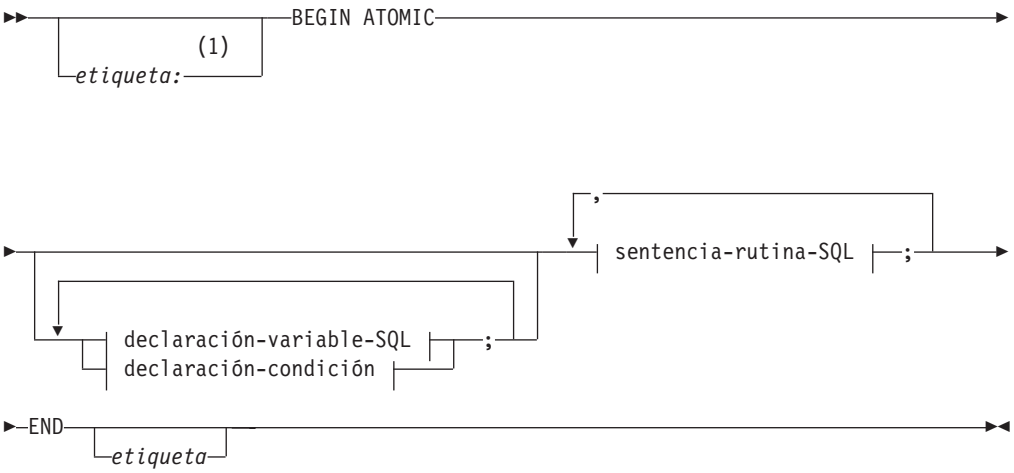
Esta sentencia puede incorporarse a un activador, una función de SQL o un método de SQL o bien emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

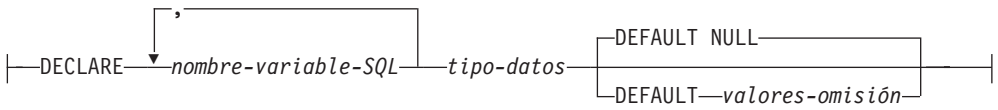
No se necesita ningún privilegio para invocar una sentencia dinámica compuesta. Sin embargo, el ID de autorización de la sentencia compuesta debe tener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta.

Sintaxis:

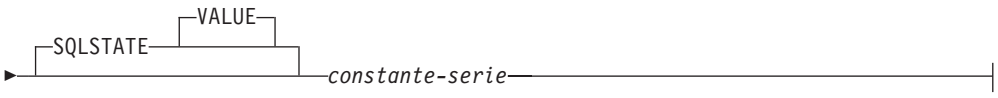
sentencia-dinámica-compuesta



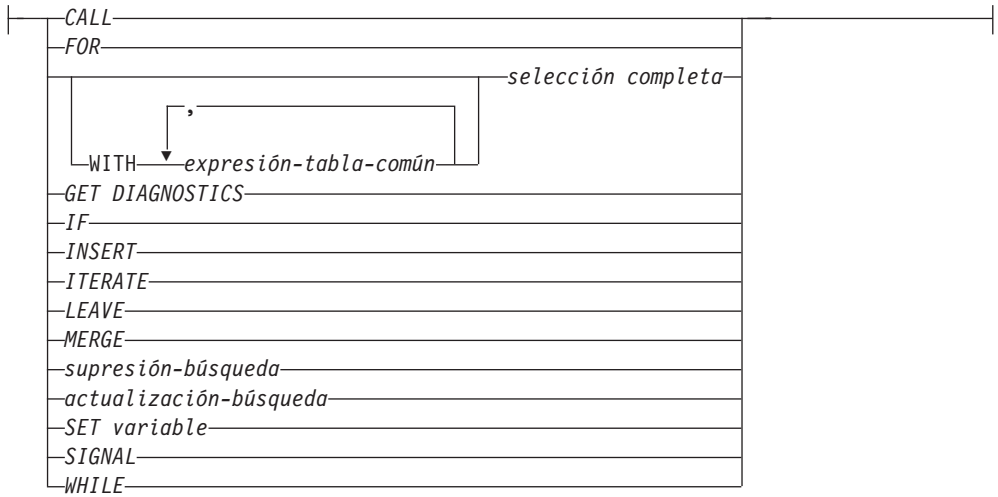
declaración-variable-SQL:



declaración-condición:



sentencia-rutina-SQL:



Notas:

1 Sólo se puede especificar una etiqueta cuando la sentencia está en una definición de función, método o activador.

Descripción:

etiqueta
Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, ésta puede utilizarse para calificar las variables SQL declaradas en la sentencia dinámica compuesta y también se puede especificar en una sentencia LEAVE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC
ATOMIC indica que, si se produce un error en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta y no se procesarán las restantes sentencias de SQL de la sentencia.

sentencia-rutina-SQL
Especifica las sentencias de SQL que se deben utilizar en la sentencia compuesta dinámica. La sentencia RETURN también se puede utilizar en una sentencia compuesta dinámica que se encuentre en una función de SQL o método de SQL. No se da soporte a una operación de actualización de búsqueda, supresión de búsqueda, inserción o fusión en apodos en SQL compuesto.

declaración-variable-SQL
Declara una variable que es local en la sentencia dinámica compuesta.

nombre-variable-SQL
Define el nombre de una variable local. DB2 convierte a mayúsculas todos los nombres de variables del SQL. El nombre no puede ser igual a:

- Otra variable de SQL de la sentencia compuesta
- Un nombre de parámetro

Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable SQL y una referencia a columna, DB2 interpreta el identificador como una columna.

tipo-datos
Especifica el tipo de datos de la variable.

DEFAULT *valores-omisión* **o NULL**
Define el valor por omisión de la variable SQL. La variable se inicializa al llamar a la sentencia dinámica compuesta. Si no se especifica un valor por omisión, el valor de la variable se inicializa en NULL.

declaración-condición
Declara el nombre de una condición y el valor SQLSTATE asociado.

nombre-condición
Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del cuerpo del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

FOR SQLSTATE *constante-serie*
Especifica el SQLSTATE asociado con la condición. La *constante-serie* se debe especificar como cinco caracteres entre comillas simples, y no puede ser '00000'.

Notas:

- DB2 compila las sentencias dinámicas compuestas como una sola sentencia. Esta sentencia está en vigor para scripts cortos que incluyan poca lógica de flujo de control pero un flujo de datos significativo. En el caso de construcciones grandes con requisitos de flujo de control complejo anidado o manejo de condiciones, una opción mejor es la utilización de procedimientos de SQL. Para obtener información más detallada acerca de la utilización de los procedimientos de SQL, consulte "CREATE PROCEDURE".
- Un procedimiento llamado dentro de una sentencia compuesta no debe emitir una sentencia COMMIT ni ROLLBACK (SQLSTATE 42985).
- ***Restricciones de acceso a las tablas:***

  Si un procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del procedimiento puede acceder a una tabla que la sentencia compuesta que ha invocado el procedimiento está modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del procedimiento puede modificar una tabla que la sentencia compuesta que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

Ejemplos:

Ejemplo 1:

Este ejemplo ilustra cómo se puede utilizar PL SQL incorporado en un escenario de depósito de datos para la limpieza de datos.

El ejemplo presenta tres tablas. La tabla "target" contiene los datos limpiados. La tabla "except" almacena filas que no se pueden limpiar (excepciones) y la tabla "source" contiene los datos sin procesar que se deben limpiar.

Se utiliza una función SQL simple llamada "discretize" para clasificar y modificar los datos. Ésta devuelve NULL para todos los datos incorrectos. Entonces la

sentencia dinámica compuesta limpia los datos. Recorre todas las filas de la tabla fuente en un bucle-FOR y decide si la fila actual se inserta en la tabla ″target″ o la tabla ″except″, dependiendo del resultado de la función ″discretize″. Con esta técnica es posible utilizar mecanismos más elaborados (limpieza en varias etapas).

Se puede escribir el mismo código utilizando un procedimiento SQL o cualquier otro procedimiento o aplicación en un lenguaje principal. Sin embargo, la sentencia dinámica compuesta ofrece una ventaja exclusiva porque el bucle-FOR no abre un cursor y las inserciones de fila individuales no son realmente inserciones de fila individuales. De hecho, la lógica es en efecto una inserción de múltiples tablas desde una selección compartida.

Esto se logra mediante la compilación de la sentencia dinámica compuesta como una sentencia individual. De forma similar a una vista cuyo cuerpo se integra en la consulta que la utiliza y entonces se compila y se optimiza como una totalidad en el contexto de consulta, el optimizador DB2 compila y optimiza el flujo de control y el flujo de datos. Por consiguiente, la lógica entera se ejecuta en tiempo de ejecución de DB2′. No se mueven datos fuera del motor central de DB2, como sucedería en el caso de un procedimiento almacenado.

El primer paso es crear las tablas necesarias:

```
CREATE TABLE target
  (pk INTEGER NOT NULL
    PRIMARY KEY, c1 INTEGER)
```

Esto crea una tabla denominada TARGET para contener los datos que se han limpiado.

```
CREATE TABLE except
  (pk INTEGER NOT NULL
    PRIMARY KEY, c1 INTEGER)
```

Esto crea una tabla denominada EXCEPT para contener las excepciones.

```
CREATE TABLE source
  (pk INTEGER NOT NULL
    PRIMARY KEY, c1 INTEGER)
```

Esto crea una tabla denominada SOURCE para contener los datos que van a limpiarse.

A continuación, creamos una función ″discretize″ para limpiar los datos echando fuera todos los valores [0..1000] y alineándolos en pasos de 10.

```
CREATE FUNCTION discretize(raw INTEGER) RETURNS INTEGER
  RETURN CASE
    WHEN raw < 0 THEN CAST(NULL AS INTEGER)
    WHEN raw > 1000 THEN NULL
    ELSE ((raw / 10) * 10) + 5
  END
```

A continuación, insertaremos los valores:

```
INSERT INTO source (pk, c1)
  VALUES (1,   -5),
   (2, NULL),
   (3, 1200),
   (4,   23),
   (5,   10),
   (6,  876)
```

Invoque la función:

```
BEGIN ATOMIC          FOR row AS
         SELECT pk, c1, discretize(c1) AS d FROM source
DO             IF row.d is NULL THEN
                INSERT INTO except VALUES(row.pk, row.c1);
ELSE                INSERT INTO target VALUES(row.pk, row.d);
            END IF;
          END FOR;
     END
```

Y compruebe los resultados:

```
SELECT * FROM except ORDER BY 1
PK          C1
----------- -----------
          1          -5
          2           -
          3        1200
  3 registros(s) seleccionado(s).

SELECT * FROM target ORDER BY 1
PK          C1
----------- -----------
          4          25
          5          15
          6         875
  3 registros(s) seleccionado(s).
```

El paso final consiste en limpiar:

```
DROP FUNCTION discretize
DROP TABLE source
DROP TABLE target
DROP TABLE except
```

Información relacionada:

- "CREATE PROCEDURE" en la página 307

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"

SQL compuesto (Incorporado)

Combina una o más sentencias de SQL distintas (*subsentencias*) en un bloque ejecutable.

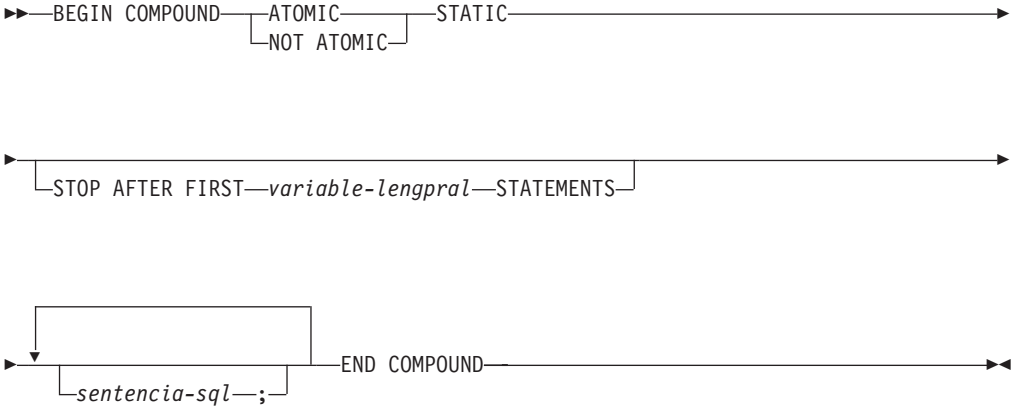
Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. Toda la construcción de la sentencia de SQL compuesta es una sentencia ejecutable que no puede prepararse de forma dinámica. La sentencia no está soportada en REXX.

Autorización:

Ninguna para la sentencia de SQL compuesta propiamente dicha. El ID de autorización de la sentencia de SQL compuesta debe tener la autorización adecuada en todas las sentencias individuales que están contenidas dentro de la sentencia de SQL compuesta.

Sintaxis:



Descripción:

ATOMIC
Especifica que, si falla alguna de las subsentencias de la sentencia de SQL compuesta, se deshacen todos los cambios efectuados en la base de datos por cualquiera de las subsentencias, incluidos los cambios realizados por subsentencias satisfactorias.

NOT ATOMIC
Especifica que, sin tener en cuenta si falla alguna subsentencia, la sentencia de SQL compuesta no deshará ningún cambio efectuado en la base de datos por las otras subsentencias.

STATIC
Especifica que las variables de entrada para todas las subsentencias conservan su valor original. Por ejemplo, si

```
SELECT ... INTO :abc ...
```

va seguido de:

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

la sentencia UPDATE utilizará el valor que :abc tenía al principio de la ejecución de la sentencia de SQL compuesta, no el valor que sigue a SELECT INTO.

Si más de una subsentencia establece la misma variable, el valor de dicha variable después de la sentencia de SQL compuesta es el valor establecido por la última subsentencia.

Nota: No se da soporte al funcionamiento no estático. Esto quiere decir que la ejecución de las subsentencias no es secuencial y que no deben tener interdependencias.

STOP AFTER FIRST
Especifica que sólo se ejecutarán un número determinado de subsentencias.

variable-lengpral
Un entero pequeño que especifica el número de subsentencias que se deben ejecutar.

STATEMENTS
Completa la cláusula STOP AFTER FIRST *variable-lengpral*.

sentencia-sql
Todas las sentencias ejecutables, excepto las siguientes, pueden estar contenidas en una sentencia de SQL compuesta estática incorporada:

```
CALL                FETCH
CLOSE               OPEN
CONNECT             PREPARE
SQL compuesto       RELEASE (conexión)
DESCRIBE            ROLLBACK (transacción)
DISCONNECT          SET CONNECTION
EXECUTE IMMEDIATE
```

Nota: INSERT, UPDATE y DELETE no reciben soporte en el SQL combinado para su utilización con apodos.

Si se incluye una sentencia COMMIT, debe ser la última subsentencia. Si COMMIT está en esta posición, se emitirá aunque la cláusula STOP AFTER FIRST *variable-lengpral* STATEMENTS indique que no deben ejecutarse todas las subsentencias. Por ejemplo, suponga que COMMIT es la última subsentencia en un bloque SQL compuesto formado por 100 subsentencias. Si la cláusula STOP AFTER FIRST STATEMENTS indica que sólo deben ejecutarse 50 subsentencias, entonces COMMIT será la subsentencia número 51.

Se devolverá un error si se incluye COMMIT al utilizar CONNECT TYPE 2 o ejecutar en un entorno de proceso de transacciones distribuidas XA (SQLSTATE 25000).

Normas:

- DB2 Connect no da soporte a las sentencias SELECT que seleccionan columnas LOB en un bloque de SQL compuesto.
- No está permitido ningún código de lenguaje principal en una sentencia de SQL compuesta; es decir, no está permitido ningún código de lenguaje principal entre las subsentencias que componen la sentencia de SQL compuesta.
- DB2 Connect sólo acepta sentencias de SQL compuestas no atómicas (NOT ATOMIC).
- Las sentencias de SQL compuestas no pueden anidarse.
- No se permite una sentencia COMMIT preparada en una sentencia de SQL compuesta ATOMIC

Notas:

Se devuelve una SQLCA para toda la sentencia de SQL compuesta. La mayor parte de la información de dicha SQLCA refleja los valores establecidos por el servidor de aplicaciones cuando ha procesado la última subsentencia. Por ejemplo:

- Normalmente, SQLCODE y SQLSTATE son los que corresponden a la última subsentencia (la excepción se describe en el punto siguiente).
- Si se ha devuelto el aviso 'no se han encontrado datos' (SQLSTATE '02000'), a ese aviso se da prioridad respecto a cualquier otro aviso con el fin de que pueda realizarse una acción en la excepción WHENEVER NOT FOUND. (Esto significa que los campos SQLCODE, SQLERRML, SQLERRMC y SQLERRP de la SQLCA que finalmente se devuelve a la aplicación son los campos de la subsentencia que ha activado el aviso 'no se han encontrado datos'. Si existe más de un aviso 'no se han encontrado datos' dentro de la sentencia de SQL compuesto, los campos de la última subsentencia serán los campos que se devuelven.)
- Los indicadores SQLWARN son una acumulación de los indicadores establecidos para todas las subsentencias.

Si se han producido uno o más errores durante la ejecución de NOT ATOMIC del SQL compuesto y ninguno de ellos es grave, SQLERRMC contendrá información sobre un máximo de siete errores. El primer símbolo de SQLERRMC indicará el número total de errores que se han producido. Los símbolos restantes contendrán cada uno la posición de orden y el SQLSTATE de la subsentencia anómala dentro de la sentencia de SQL compuesta. El formato es una serie de caracteres en el formato:

```
nnnXsssccccc
```

en la que la subserie que empieza por X se repite hasta seis veces más y los elementos de la serie están definidos de la manera siguiente.

nnn El número total de sentencias que han producido errores. (Si el número excede de 999, el recuento volverá a empezar desde cero.) Este campo se justifica por la izquierda y se rellena con blancos.

X El separador de símbolos X'FF'.

sss La posición ordinal de la sentencia que ha provocado el error. (Si el número excede de 999, el recuento volverá a empezar desde cero.) Por ejemplo, si la primera sentencia no se ha ejecutado satisfactoriamente, este campo contendrá el número uno justificado por la izquierda ('1 ').

ccccc El SQLSTATE del error.

El segundo campo SQLERRD contiene el número de sentencias que han fallado (han devuelto SQLCODE negativos).

El tercer campo SQLERRD de la SQLCA es una acumulación del número de filas afectadas por todas las subsentencias.

El cuarto campo SQLERRD de la SQLCA es una cuenta del número de subsentencias satisfactorias. Si, por ejemplo, falla la tercera subsentencia de una sentencia de SQL compuesta, el cuarto campo SQLERRD se establecería en 2, lo que indicaría que 2 subsentencias se habrían procesado de manera satisfactoria antes de encontrar el error.

El quinto campo SQLERRD de la SQLCA es una acumulación del número de filas actualizadas o suprimidas a causa de imposición de las restricciones de integridad de referencia para todas las subsentencias que han activado dicha actividad de restricción.

Ejemplos:

Ejemplo 1: En un programa C, emita una sentencia de SQL compuesto que actualice las tablas ACCOUNTS y TELLERS. Si hay un error en cualquiera de las sentencias, deshaga el efecto de todas las sentencias (ATOMIC). Si no hay errores, confirme la unidad de trabajo actual.

```
EXEC SQL BEGIN COMPOUND ATOMIC STATIC
 UPDATE ACCOUNTS SET ABALANCE = ABALANCE + :delta
   WHERE AID = :aid;
 UPDATE TELLERS SET TBALANCE = TBALANCE + :delta
   WHERE TID = :tid;
  INSERT INTO TELLERS (TID, BID, TBALANCE) VALUES (:i, :branch_id, 0);
 COMMIT;
END COMPOUND;
```

Ejemplo 2: En un programa C, inserte 10 filas de datos en la base de datos Suponga que la variable del lenguaje principal :nbr contiene el valor 10 y S1 es una sentencia INSERT preparada. Además, suponga que todas las inserciones deben intentarse sin tener en cuenta los errores (NOT ATOMIC).

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC STOP AFTER FIRST :nbr STATEMENTS
  EXECUTE S1 USING DESCRIPTOR :*sqlda0;
  EXECUTE S1 USING DESCRIPTOR :*sqlda1;
  EXECUTE S1 USING DESCRIPTOR :*sqlda2;
  EXECUTE S1 USING DESCRIPTOR :*sqlda3;
  EXECUTE S1 USING DESCRIPTOR :*sqlda4;
  EXECUTE S1 USING DESCRIPTOR :*sqlda5;
  EXECUTE S1 USING DESCRIPTOR :*sqlda6;
  EXECUTE S1 USING DESCRIPTOR :*sqlda7;
  EXECUTE S1 USING DESCRIPTOR :*sqlda8;
  EXECUTE S1 USING DESCRIPTOR :*sqlda9;
END COMPOUND;
```

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139

Ejemplos relacionados:

- "dbuse.sqc -- How to use a database (C)"
- "dbuse.sqC -- How to use a database (C++)"

SQL compuesto (procedimiento)

Una sentencia compuesta de procedimiento agrupa otras sentencias en un procedimiento SQL. Dentro de una sentencia compuesta se pueden declarar variables SQL, cursores y gestores de condiciones.

Invocación:

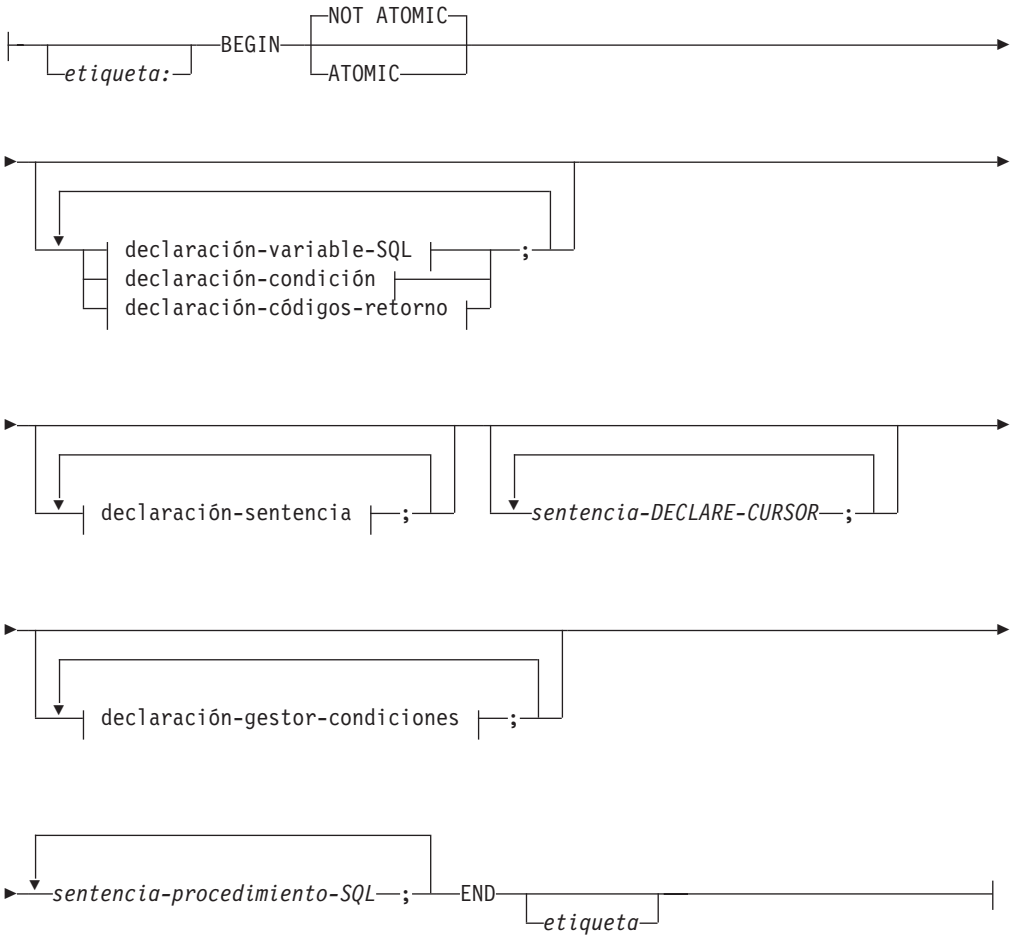
Esta sentencia sólo puede incorporarse en un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización:

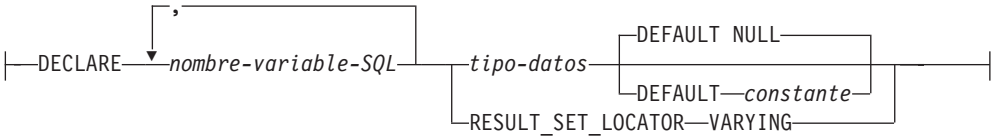
No se requieren privilegios para invocar una sentencia compuesta de procedimiento. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta de procedimiento. Para obtener información acerca de la autorización necesaria para utilizar un cursor, consulte la descripción de la sentencia DECLARE CURSOR.

Sintaxis:

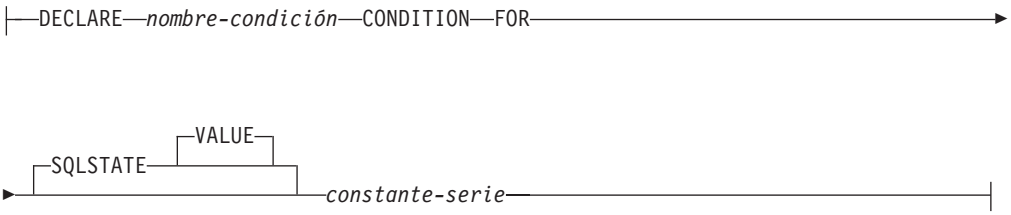
sentencia-compuesta-procedimiento:



declaración-variable-SQL:

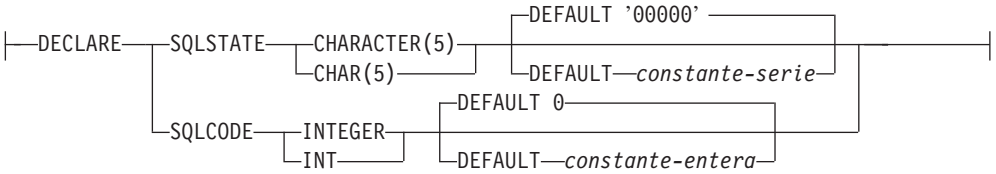


declaración-condición:

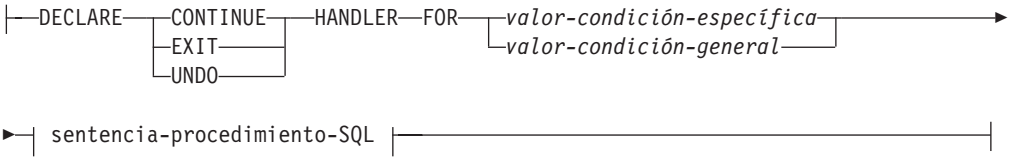


declaración-sentencia:

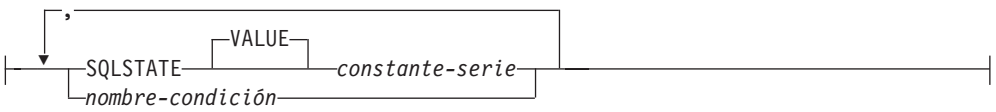
declaración-códigos-retorno:



declaración-gestor-condiciones:



valor-condición-específica:



valor-condición-general:

sentencia-procedimiento-SQL:

Descripción:

etiqueta
Define la etiqueta del bloque de programa. Si se especifica la etiqueta inicial, se puede utilizar para calificar variables SQL declaradas en la sentencia compuesta y también se puede especificar en una sentencia LEAVE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC o NOT ATOMIC
ATOMIC indica que si se produce una condición de excepción no manejada en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta. NOT ATOMIC indica que una condición de excepción no manejada dentro de la sentencia compuesta no da lugar a la retrotracción de la sentencia compuesta.

declaración-variable-SQL
Declara una variable que es local respecto a la sentencia compuesta.

nombre-variable-SQL
Define el nombre de una variable local. DB2 convierte a mayúsculas todos los nombres de variables del SQL. El nombre no puede ser el mismo que otra variable SQL existente en la misma sentencia compuesta y no puede ser igual que un nombre de parámetro. Los nombres de variables SQL no deben ser iguales que los nombres de columnas. Si una sentencia de SQL contiene un identificador que tiene el mismo nombre que una variable SQL y una referencia a columna, DB2 interpreta el identificador como una columna. Si la sentencia compuesta en la que se ha declarado la variable se ha etiquetado, las utilizaciones de la variable pueden calificarse con la etiqueta. Por ejemplo, a la variable V declarada en la sentencia compuesta que se ha etiquetado con C puede hacerse referencia como C.V.

tipo-datos
Especifica el tipo de datos de la variable. LONG VARCHAR, LONG VARGRAPHIC, DATALINK, REFERENCE y los tipos estructurados definidos por el usuario no reciben soporte (SQLSTATE 429BB).

DEFAULT *constante* **o NULL**
Define el valor por omisión de la variable SQL. La variable se inicializa cuando se invoca el procedimiento SQL. Si no se especifica un valor por omisión, el valor de la variable se inicializa en NULL.

RESULT_SET_LOCATOR VARYING
Especifica el tipo de datos de una variable localizadora de conjuntos de resultados.

declaración-condición
Declara el nombre de una condición y el valor SQLSTATE asociado.

nombre-condición
Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del cuerpo del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

FOR SQLSTATE *constante-serie*
Especifica el SQLSTATE que está asociado a la condición. La constante-serie se debe especificar en forma de cinco caracteres encerrados entre comillas simples y no puede ser '00000'.

declaración-sentencia
Declara una lista de uno o más nombres que son locales en la sentencia compuesta. Un nombre de sentencia no puede ser igual que otro nombre de sentencia en la misma sentencia compuesta.

declaración-códigos-retorno
Declara las variables especiales llamadas SQLSTATE y SQLCODE, que se establecen automáticamente en el valor que se devuelve tras procesar una sentencia de SQL. Las variables SQLSTATE y SQLCODE sólo se pueden declarar en la sentencia compuesta más externa del cuerpo del procedimiento SQL. Estas variables sólo se pueden declarar una vez para cada procedimiento SQL.

sentencia-declare-cursor
Declara un cursor en el cuerpo del procedimiento. Cada cursor debe tener un nombre exclusivo. El cursor sólo puede estar referenciado dentro de la sentencia compuesta. Utilice una sentencia OPEN para abrir el cursor, y una sentencia FETCH para leer filas utilizando el cursor. Para que el procedimiento SQL devuelva conjuntos de resultados a la aplicación cliente, el cursor se debe declarar utilizando la cláusula WITH RETURN. El ejemplo siguiente devuelve un conjunto de resultados a la aplicación cliente:

```
CREATE PROCEDURE RESULT_SET()
  LANGUAGE SQL
  RESULT SETS 1
  BEGIN
    DECLARE C1 CURSOR WITH RETURN FOR
      SELECT id, name, dept, job
        FROM staff;
    OPEN C1;
END
```

Nota: Para procesar conjuntos de resultados, debe escribir la aplicación cliente utilizando una de las interfaces de programación de aplicaciones siguientes: Interfaz a nivel de llamada de DB2 (CLI de DB2), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC) o SQL incorporado para Java (SQLJ).

Para obtener más información acerca de la declaración de un cursor, consulte "DECLARE CURSOR".

declaración-gestor-condiciones
Especifica un *descriptor de contexto*, una *sentencia-procedimiento-SQL* que debe ejecutarse cuando se produzca una condición de excepción o de terminación en la sentencia compuesta. *sentencia-procedimiento-SQL* es una sentencia que se ejecuta cuando el gestor de condiciones recibe el control.

Existe un descriptor de contexto activo para el conjunto de *sentencias-procedimiento-SQL* que siguen al conjunto de *declaraciones-descriptor-contexto* dentro de la sentencia compuesta en la que se ha declarado el descriptor de contexto, incluidas las sentencias compuestas anidadas.

Existen tres tipos de gestores de condiciones:

CONTINUE
Tras la invocación satisfactoria del gestor de condiciones, el control pasa a la sentencia de SQL que sigue a continuación de la sentencia que provocó la condición de excepción. Si el error que causó la excepción es una sentencia FOR, IF, CASE, WHILE o REPEAT (pero no una sentencia de procedimiento SQL incluida dentro de una de aquéllas), el control pasa a la sentencia que sigue a continuación de END FOR, END IF, END CASE, END WHILE o END REPEAT.

EXIT
Tras la invocación satisfactoria del gestor de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el gestor de condiciones.

UNDO
Antes de invocar el gestor de condiciones, se retrotraen los cambios de SQL que se hicieron en la sentencia compuesta. Tras la invocación satisfactoria del gestor de condiciones, el control se transfiere al final de la sentencia compuesta donde se declaró el gestor de condiciones. Si se especifica UNDO, la sentencia compuesta en la que se declara el descriptor de contexto debe ser ATOMIC.

Las condiciones que dan lugar a la activación del descriptor de contexto se definen en la declaración-descriptor-contexto, tal como se indica a continuación:

valor-condición-específica
Especifica que el descriptor de contexto es un *descriptor de contexto de condiciones específicas*.

SQLSTATE *serie*
Especifica un SQLSTATE para el cual se invoca el gestor de condiciones. Los dos primeros caracteres del valor SQLSTATE no deben ser "00".

nombre-condición
Especifica una condición para la cual se invoca el gestor de condiciones. El nombre de la condición debe estar definido previamente en una declaración de condición.

valor-condición-general
Especifica que el descriptor de contexto es un *descriptor de contexto de condiciones generales*.

SQLEXCEPTION
Especifica que el descriptor de contexto se invoca cuando se produce una condición de excepción. Una condición de excepción se representa por medio de un valor SQLSTATE cuyos dos primeros caracteres no son "00", "01" ó "02".

SQLWARNING
Especifica que el descriptor de contexto se invoca cuando se produce una condición de aviso. Una condición de aviso se representa por medio de un valor SQLSTATE cuyos dos primeros caracteres son "01".

NOT FOUND
Especifica que el gestor de condiciones se invoca cuando se produce una condición de ″no encontrado″ (NOT FOUND). Una

condición NOT FOUND se representa por medio de un valor SQLSTATE cuyos dos primeros caracteres son "02".

sentencia-procedimiento-SQL
Especifica la sentencia de procedimiento de SQL.

etiqueta
Especifica una etiqueta para la sentencia de procedimiento de SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control SQL admiten la especificación de una lista de sentencias de procedimiento SQL.

sentencia-SQL
Todas las sentencias de SQL ejecutables pueden estar contenidas en el cuerpo de un procedimiento de SQL, a excepción de las siguientes:

- ALTER
- CONNECT
- CREATE, para cualquier objeto excepto índices, tablas o vistas
- DESCRIBE
- DISCONNECT
- DROP, para cualquier objeto excepto índices, tablas o vistas
- FLUSH EVENT MONITOR
- REFRESH TABLE
- RELEASE (sólo conexión)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY
- SET PASSTHRU
- SET SERVER OPTION

Sólo se da soporte a las sentencias siguientes en el ámbito de un procedimiento de SQL:

- ALLOCATE CURSOR
- ASSOCIATE LOCATORS
- CASE
- GOTO
- LOOP
- SQL compuesto (procedimiento)
- REPEAT
- RESIGNAL

Normas:

- Las sentencias compuestas definidas como ATOMIC no se pueden anidar.
- Las normas siguientes son aplicables a la declaración de un gestor de condiciones:

- Una declaración de descriptor de contexto no puede contener el mismo *nombre-condición* o valor SQLSTATE más de una vez y no puede contener un valor SQLSTATE y un *nombre-condición* que representen el mismo valor SQLSTATE.
- Cuando se declaran dos o más descriptores de contexto de condiciones en una sentencia compuesta:
  - Dos declaraciones de descriptor de contexto no pueden especificar la misma categoría de condición general (SQLEXCEPTION, SQLWARNING, NOT FOUND).
  - Dos declaraciones de descriptor de contexto no pueden especificar la misma condición específica, ya sea como un valor SQLSTATE o como un *nombre-condición* que represente el mismo valor.
- Un gestor de condiciones se activa cuando es el gestor más apropiado para una condición de excepción o terminación. El descriptor de contexto más adecuado se determina basándose en las consideraciones siguientes:
  - El ámbito de una declaración de descriptor de contexto *H* es la lista de la *sentencia-procedimiento-SQL* que sigue a las declaraciones de descriptor de contexto contenidas dentro de la sentencia compuesta en la que aparece *H*. Esto significa que el ámbito de *H* no incluye las sentencias contenidas en el cuerpo del descriptor de contexto de condiciones *H*, lo que implica que un descriptor de contexto de condiciones no puede manejar las condiciones que se producen dentro de su propio cuerpo. De forma similar, en el caso de dos descriptores de contexto cualesquiera, *H1* y *H2*, declarados en la misma sentencia compuesta, *H1* no manejará las condiciones que se produzcan en el cuerpo de *H2* y *H2* no manejará las condiciones que se produzcan en el cuerpo de *H1*.
  - Un descriptor de contexto para un *valor-condición-específica* o un *valor-condición-general* C declarado en un ámbito interno tiene prioridad respecto a cualquier otro descriptor de contexto para C declarado en un ámbito de inclusión.
  - Cuando, en el mismo ámbito, se declaran un descriptor de contexto específico para una condición *C* y un descriptor de contexto general que también manejaría *C*, el descriptor de contexto específico tiene prioridad respecto al descriptor de contexto general.

  Si se produce una condición de excepción para la que no existe un descriptor de contexto adecuado, el procedimiento de SQL que contiene la sentencia anómala finaliza con una condición de excepción no manejada. Si se produce una condición de terminación para la que no existe un descriptor de contexto adecuado, la ejecución continúa con la siguiente sentencia de SQL.

Ejemplos:

Creación de un cuerpo de procedimiento con una sentencia compuesta que ejecuta las acciones siguientes:

1. Declara variables SQL
2. Declara un cursor para que proporcione el salario de los empleados de un departamento determinado de acuerdo con un parámetro IN. En la sentencia SELECT, convierte de DECIMAL a DOUBLE el tipo de datos de la columna *salary* (salario).
3. Declara un gestor de condiciones EXIT para la condición NOT FOUND (fin de archivo), que asigna el valor '6666' al parámetro de salida `medianSalary` (salario medio)

4. Selecciona el número de empleados del departamento especificado y lo coloca en la variable SQL numRecords
5. Lee filas desde el cursor en un bucle WHILE hasta llegar al 50% + 1 de los empleados
6. Devuelve el salario medio

```
CREATE PROCEDURE DEPT_MEDIAN
  (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
  LANGUAGE SQL
  BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
      SELECT CAST(salary AS DOUBLE) FROM staff
        WHERE DEPT = deptNumber
        ORDER BY salary;
    DECLARE EXIT HANDLER FOR NOT FOUND
      SET medianSalary = 6666;
-- inicializar parámetro de salida
      SET medianSalary = 0;
      SELECT COUNT(*) INTO v_numRecords FROM staff
        WHERE DEPT = deptNumber;
      OPEN c1;
      WHILE v_counter < (v_numRecords / 2 + 1) DO
        FETCH c1 INTO medianSalary;
        SET v_counter = v_counter + 1;
      END WHILE;
      CLOSE c1;
  END
```

En el ejemplo siguiente se muestra el flujo de ejecución en un caso hipotético donde se ha activado un descriptor de contexto UNDO desde otra condición como resultado de RESIGNAL:

```
CREATE PROCEDURE A()
LANGUAGE SQL
CS1: BEGIN ATOMIC
  DECLARE C CONDITION FOR SQLSTATE '12345';
  DECLARE D CONDITION FOR SQLSTATE '23456';

  DECLARE UNDO HANDLER FOR C
  H1: BEGIN
    -- Retrotraer tras error, realizar limpieza final y salir
    -- de procedimiento A.

    -- ...

    -- Cuando se completa este descriptor de contexto, la ejecución
    -- continúa tras la sentencia compuesta CS1; el procedimiento A
    -- terminará.
         END;

  -- Realizar aquí algún trabajo ...
  CS2: BEGIN
    DECLARE CONTINUE HANDLER FOR D
    H2: BEGIN
      -- Realizar recuperación local y reenviar la condición de
      -- error al descriptor de contexto externo para su proceso
      -- adicional.

      -- ...

      RESIGNAL C; -- activará el descriptor de contexto UNDO H1; la ejecución
                  -- WILL NOT no volverá aquí.  Los cursores locales
                  -- declarados en H2 y CS2 se cerrarán.
        END;
```

```
    -- Realizar aquí algún trabajo adicional ...

    -- Simular la generación de la condición D por parte de alguna
    -- sentencia de SQL en la sentencia compuesta CS2:
    SIGNAL D; -- activará H2
  END;
END
```

Información relacionada:

- "DECLARE CURSOR" en la página 483
- "Tipos de datos" en la publicación *Consulta de SQL, Volumen 1*

CONNECT (Tipo 1)

La sentencia CONNECT (Tipo 1) conecta un proceso de aplicación con el servidor de aplicaciones identificado según las normas para una unidad de trabajo remota.

Un proceso de aplicación sólo puede estar conectado a un único servidor de aplicaciones en un momento dado. Se denomina *servidor actual*. Puede establecerse un servidor de aplicaciones por omisión cuando se inicializa el peticionario de aplicaciones. Si la conexión implícita está disponible y se inicia un proceso de aplicación, se conecta de manera implícita al servidor de aplicaciones por omisión. El proceso de aplicación puede conectarse explícitamente a un servidor de aplicaciones distinto emitiendo una sentencia CONNECT TO. La conexión dura hasta que se emite una sentencia CONNECT RESET o una sentencia DISCONNECT o hasta que otra sentencia CONNECT TO cambia el servidor de aplicaciones.

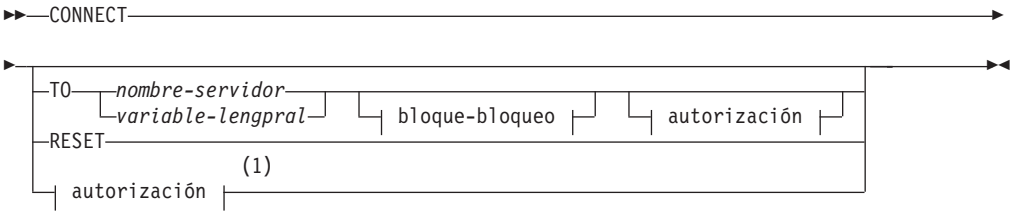
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

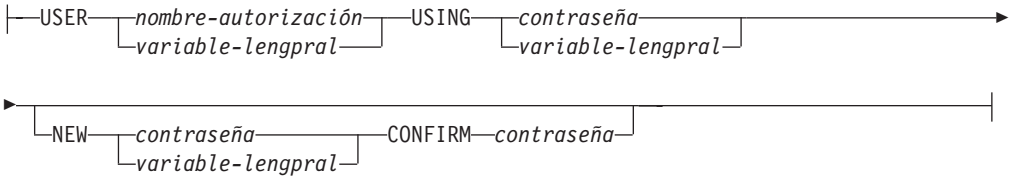
Autorización:

El ID de autorización de la sentencia debe estar autorizado para conectar con el servidor de aplicaciones identificado. Según cual sea el valor de autentificación definido para la base de datos, el control de autorizaciones puede realizarlo el cliente o el servidor. En una base de datos particionada, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones.

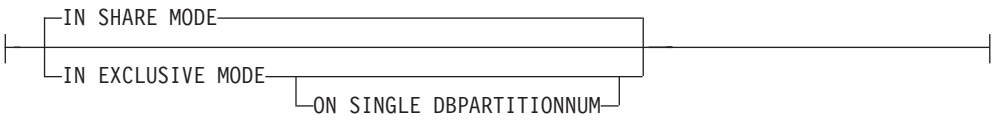
Sintaxis:



autorización:



bloque-bloqueo:



Notas:

1 Este formato sólo es válido si se ha habilitado la conexión implícita.

Descripción:

CONNECT (sin ningún operando)
Devuelve información acerca del servidor actual. La información se devuelve en el campo SQLERRP de la SQLCA, tal como se describe en "Conexión satisfactoria".

Si existe un estado de conexión, el ID de autorización y el alias de base de datos se colocan en el campo SQLERRMC de la SQLCA. Si el ID de autorización es mayor que 8 bytes, se truncará a 8 bytes, y el truncamiento se indicará en los campos SQLWARN0 y SQLWARN1 de la SQLCA, mediante 'W' y 'A', respectivamente. Si el parámetro de configuración de base de datos DYN_QUERY_MGMT está habilitado, los campos SQLWARN0 y SQLWARN7 de la SQLCA se marcarán con los distintivos 'W' y 'E' respectivamente.

Si no existe ninguna conexión y es posible la conexión implícita, se intenta efectuar una conexión implícita. Si no hay una conexión implícita disponible, este intento produce un error (no hay ninguna conexión existente). Si no hay conexión, el campo SQLERRMC está en blanco.

El código de territorio y la página de códigos del servidor de aplicaciones se colocan en el campo SQLERRMC (como sucede cuando una sentencia CONNECT TO se ejecuta satisfactoriamente).

Esta forma de CONNECT:

- No necesita que el proceso de aplicación esté en estado conectable.
- Si está conectado, no cambia el estado de conexión.
- Si está desconectado y está disponible la conexión implícita, se realiza una conexión con el servidor de aplicaciones por omisión. En este caso, el código de país o región y la página de códigos del servidor de aplicaciones se colocan en el campo SQLERRMC, como sucede cuando una sentencia CONNECT TO se ejecuta satisfactoriamente.
- Si está desconectado y no está disponible la conexión implícita, el proceso de aplicación permanece desconectado.
- No cierra los cursores.

TO *nombre-servidor* o *variable-lengpral*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lengpral* que contenga el nombre-servidor.

Si se especifica una *variable-lengpral*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable indicadora. El *nombre-servidor* que está contenido en la *variable-lengpral* debe estar justificado por la izquierda y no debe estar delimitado por comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del solicitante de la aplicación.

Nota: DB2 UDB para OS/390 y z/OS da soporte a un nombre de ubicación de 16 bytes y DB2 UDB para iSeries da soporte a un nombre de base de datos de destino de 18 bytes. DB2 Versión 8 sólo da soporte a la utilización de un nombre de alias de base de datos de 8 bytes en la sentencia SQL CONNECT. Sin embargo, el nombre de alias de base de datos puede correlacionarse con un nombre de base de datos de 18 bytes por medio del Directorio de servicio de conexión de bases de datos.

Cuando se ejecuta la sentencia CONNECT TO, el proceso de aplicación debe estar en estado de conexión.

Conexión satisfactoria:

Si la sentencia CONNECT TO es satisfactoria:

- Se cierran todos los cursores abiertos, se destruyen todas las sentencias preparadas y se liberan todos los bloqueos del servidor de aplicaciones anterior.
- El proceso de aplicación se desconecta de su servidor de aplicaciones anterior, si lo hay, y se conecta al servidor de aplicaciones identificado.
- El nombre real del servidor de aplicaciones (no un alias) se coloca en el registro especial CURRENT SERVER.
- Se coloca información acerca del servidor de aplicaciones en el campo SQLERRP de SQLCA. Si el servidor de aplicaciones es un producto IBM, la información tiene el formato *pppvvrrm*, donde:
  - *ppp* identifica el producto de la manera siguiente:
    - DSN para DB2 UDB para OS/390 y z/OS
    - ARI para DB2 Server para VSE & VM
    - QSQ para DB2 UDB para iSeries
    - SQL para DB2 UDB para UNIX y Windows
  - *vv* es un identificador de versión de dos dígitos como, por ejemplo, '08'
  - *rr* es un identificador de release de dos dígitos como, por ejemplo, '01'
  - *m* es un identificador de nivel de modificación de un dígito como, por ejemplo, '0'.

  Este release (Versión 8) de DB2 UDB para UNIX y Windows se identifica como 'SQL08010'.
- El campo SQLERRMC de la SQLCA se establece en los valores siguientes (separados por X'FF')
  1. el código de país o región del servidor de aplicaciones (o blancos si se utiliza DB2 Connect),
  2. la página de códigos del servidor de aplicaciones (o CCSID si se utiliza DB2 Connect),
  3. el ID de autorización (sólo los primeros 8 bytes como máximo),
  4. el alias de base de datos,
  5. el tipo de plataforma del servidor de aplicaciones. Los valores reconocidos actualmente son:

     Símbolo **Servidor**

QAS	DB2 Universal Database para iSeries
QDB2	DB2 Universal Database para OS/390 y z/OS
QDB2/2	DB2 Universal Database para OS/2
QDB2/6000	DB2 Universal Database para AIX
QDB2/HPUX	DB2 Universal Database para HP-UX
QDB2/LINUX	DB2 Universal Database para Linux
QDB2/NT	DB2 Universal Database para Windows NT, 2000 y XP
QDB2/SUN	DB2 Universal Database para el sistema operativo Solaris
QSQLDS/VM	DB2 Server para VM
QSQLDS/VSE	DB2 Server para VSE

6. El ID de agente. Identifica al agente que se ejecuta en el gestor de bases de datos en nombre de la aplicación. Este campo es el mismo que el elemento id_agente que devuelve el supervisor de bases de datos.
7. El índice de agente. Identifica el índice del agente y se utiliza para el servicio.
8. El número de partición. Para una base de datos no particionada, siempre es 0, si está presente.
9. La página de códigos de la aplicación cliente.
10. El número de particiones de una base de datos particionada. Si la base de datos no puede particionarse, el valor es 0 (cero). El símbolo sólo está presente con la Versión 5 o posterior.

- El campo SQLERRD(1) de la SQLCA indica la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) al convertirlos a la página de códigos de la base de datos a partir de la página de códigos de la aplicación. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción.
- El campo SQLERRD(2) de la SQLCA indica la diferencia máxima esperada en la longitud de los datos de caracteres mixtos (tipos de datos CHAR) al convertirlos a la página de códigos de la aplicación a partir de la página de códigos de la base de datos. Un valor de 0 ó 1 indica sin expansión; un valor mayor que 1 indica una posible expansión en longitud; un valor negativo indica una posible contracción.
- El campo SQLERRD(3) de la SQLCA indica si la base de datos de la conexión es actualizable o no. Inicialmente, una base de datos es actualizable, pero se cambia por de sólo lectura si una unidad de trabajo determina que el ID de autorización no puede efectuar actualizaciones. El valor es uno de los siguientes:
  - 1 - actualizable
  - 2 - sólo lectura
- El campo SQLERRD(4) de la SQLCA devuelve ciertas características de la conexión. El valor es uno de los siguientes:

  0 N/D (sólo es posible si se ejecuta desde un cliente de nivel inferior que tenga una confirmación de una fase y que sea un actualizador).

  1 confirmación de una fase.

2 confirmación de una fase; sólo lectura (únicamente aplicable a las conexiones con bases de datos DRDA1 en un entorno de supervisor de TP).

3 confirmación de dos fases.

- El campo SQLERRD(5) de la SQLCA devuelve el tipo de autentificación para la conexión. El valor es uno de los siguientes:

  0 Autentificado en el servidor.

  1 Autentificado en el cliente.

  2 Autentificado utilizando DB2 Connect.

  4 Autentificado en el servidor con cifrado.

  5 Autentificado utilizando DB2 Connect con cifrado.

  7 Autentificado utilizando un mecanismo de seguridad Kerberos externo.

  8 Autentificado utilizando un mecanismo de seguridad Kerberos externo o en el servidor con cifrado.

  9 Autentificado utilizando un mecanismo de seguridad de plugin de API GSS externo.

  10 Autentificado utilizando un mecanismo de seguridad de plugin de API GSS externo o en el servidor con cifrado.

  255 Autentificación no especificada.
- El campo SQLERRD(6) de SQLCA devuelve el número de partición para la que se ha realizado la conexión si la base de datos está particionada. De lo contrario, se devuelve un valor de 0.
- El campo SQLWARN1 de la SQLCA se establecerá en 'A' si el ID de autorización de la conexión satisfactoria es mayor que 8 bytes. Esto indica que se ha producido un truncamiento. El campo SQLWARN0 de la SQLCA se establecerá en 'W' para indicar este aviso.
- El campo SQLWARN7 de la SQLCA se establecerá en 'E' si el parámetro de configuración de base de datos DYN_QUERY_MGMT de la base de datos está habilitado. El campo SQLWARN0 de la SQLCA se establecerá en 'W' para indicar este aviso.

Conexión no satisfactoria:

Si la sentencia CONNECT TO no es satisfactoria:

- El campo SQLERRP de la SQLCA se establece en el nombre del módulo del peticionario de aplicaciones que ha detectado el error. Los tres primeros caracteres del nombre del módulo identifican el producto.
- Si la sentencia CONNECT TO no es satisfactoria porque el proceso de aplicación no se encuentra en estado conectable, el estado de conexión del proceso de aplicación no se modifica.
- Si la sentencia CONNECT TO no es satisfactoria porque el *nombre-servidor* no aparece listado en el directorio local, se emite un mensaje de error (SQLSTATE 08001) y el estado de conexión del proceso de aplicación no se modifica:
  - Si el peticionario de aplicación no estaba conectado a un servidor de aplicaciones, el proceso de aplicación sigue sin estar conectado.

  - Si el peticionario de aplicaciones ya estaba conectado a un servidor de aplicaciones, el proceso de aplicación permanece conectado a ese servidor de aplicaciones. Las sentencias posteriores se ejecutan en dicho servidor de aplicaciones.
- Si la sentencia CONNECT TO no es satisfactoria por cualquier otra razón, el proceso de aplicación se coloca en estado no conectado.

IN SHARE MODE
Permite otras conexiones simultáneas con la base de datos e impide que otros usuarios se conecten a la base de datos en modalidad exclusiva.

IN EXCLUSIVE MODE
Evita que los procesos de aplicación simultáneos ejecuten cualquier operación en el servidor de aplicaciones, a menos que tengan el mismo ID de autorización que el usuario que mantiene el bloqueo exclusivo. Esta opción no recibe el soporte de DB2 Connect.

ON SINGLE DBPARTITIONNUM
Especifica que la partición de base de datos coordinadora está conectada en modalidad exclusiva y que todas las demás particiones de base de datos están conectadas en modalidad de compartimiento. Esta opción sólo es efectiva en una base de datos particionada.

RESET
Desconecta el proceso de aplicación del servidor actual. Se realiza una operación de confirmación. Si no está disponible la conexión implícita, el proceso de aplicación continúa no conectado hasta que se emite una sentencia de SQL.

USER *nombre-autorización/variable-lengpral*
Identifica el ID de usuario que intenta conectarse con el servidor de aplicaciones. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener ninguna variable indicadora. El ID de usuario que está dentro de la *variable-lengpral* debe justificarse por la izquierda y no debe delimitarse utilizando comillas.

USING *contraseña/variable-lengpral*
Identifica la contraseña del ID de usuario que intenta conectarse con el servidor de aplicaciones. La *contraseña* o *variable-lengpral* pueden tener hasta 18 caracteres como máximo. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 18 y no debe incluir una variable indicadora.

NEW *contraseña/variable-lengpral* **CONFIRM** *contraseña*
Identifica la nueva contraseña que debe asignarse al ID de usuario que se identifica por medio de la opción USER. La *contraseña* o *variable-lengpral* pueden tener hasta 18 caracteres como máximo. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 18 y no debe incluir una variable indicadora. El sistema en que se cambiará la contraseña depende de cómo esté configurada la autentificación de usuario.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM

- Es recomendable que la primera sentencia de SQL que se ejecute en un proceso de aplicación sea la sentencia CONNECT TO.
- Si se emite una sentencia CONNECT TO para el servidor de aplicaciones actual con un ID de usuario y contraseña distintos, se eliminará la asignación de la conversación y volverá a asignarse. El gestor de bases de datos cierra todos los cursores (con pérdida de la posición del cursor si se ha utilizado la opción WITH HOLD).
- Si se emite una sentencia CONNECT TO para el servidor de aplicaciones actual con el mismo ID de usuario y contraseña, no se eliminará la asignación de la conversación ni se volverá a asignar. En este caso, los cursores no se cierran.
- Para utilizar un entorno de base de datos de varias particiones, el usuario o la aplicación debe conectarse con una de las particiones que aparecen en la lista del archivo `db2nodes.cfg`. Debe intentar asegurar que no todos los usuarios utilicen la misma partición como partición coordinadora.
- El *nombre-autorización* SYSTEM no se puede especificar explícitamente en la sentencia CONNECT. Sin embargo, en sistemas operativos Windows, las aplicaciones locales que se ejecutan bajo la cuenta del sistema local se pueden conectar implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Al conectar a un servidor Windows explícitamente, se puede especificar el *nombre-autorización* o la *variable-sistema-principal* del usuario si se utiliza el nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft Windows NT; por ejemplo, 'Dominio\Usuario'.

Ejemplos:

Ejemplo 1: En un programa en C, conéctese con el servidor de aplicaciones TOROLAB utilizando el alias de base de datos TOROLAB, el ID de usuario FERMAT y la contraseña THEOREM.

```
EXEC SQL  CONNECT TO TOROLAB USER FERMAT USING THEOREM;
```

Ejemplo 2: En un programa C, conéctese a un servidor de aplicaciones cuyo alias de base de datos esté almacenado en la variable del lenguaje principal APP_SERVER (varchar(8)). Después de haber establecido una conexión satisfactoria, copie el identificador del producto de 3 caracteres del servidor de aplicaciones en la variable PRODUCT (char(3)).

```
 EXEC SQL  CONNECT TO :APP_SERVER;
if (strncmp(SQLSTATE,'00000',5))
  strncpy(PRODUCT,sqlca.sqlerrp,3);
```

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*
- "Particionamiento de datos entre múltiples particiones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "advsql.sqb -- How to read table data using CASE (MF COBOL)"
- "dbmcon.sqc -- How to use multiple databases (C)"
- "dbmcon.sqC -- How to use multiple databases (C++)"

CONNECT (Tipo 2)

La sentencia CONNECT (Tipo 2) conecta un proceso de aplicación con el servidor de aplicaciones identificado y establece las normas para una unidad de trabajo distribuida y dirigida por aplicación. Este servidor es entonces el servidor actual para el proceso.

La mayor parte de los aspectos de una sentencia CONNECT (Tipo 1) son también aplicables a una sentencia CONNECT (Tipo 2). En lugar de repetir ahora lo mismo, esta sección sólo describe los elementos del Tipo 2 que difieren del Tipo 1.

Invocación:

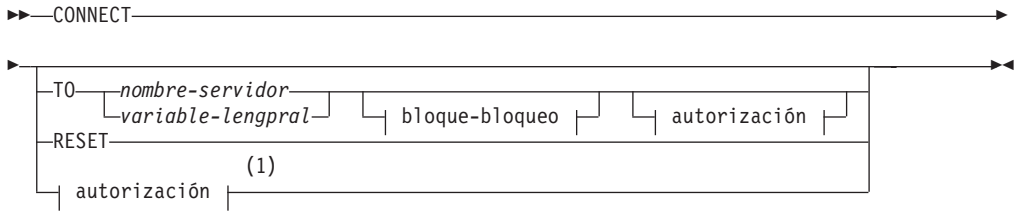
Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

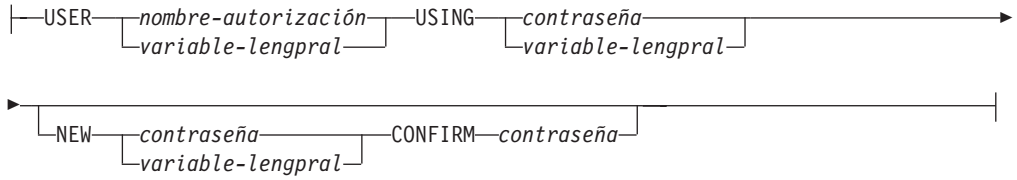
El ID de autorización de la sentencia debe estar autorizado para conectar con el servidor de aplicaciones identificado. Según cual sea el valor de autentificación definido para la base de datos, el control de autorizaciones puede realizarlo el cliente o el servidor. En una base de datos particionada, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones.

Sintaxis:

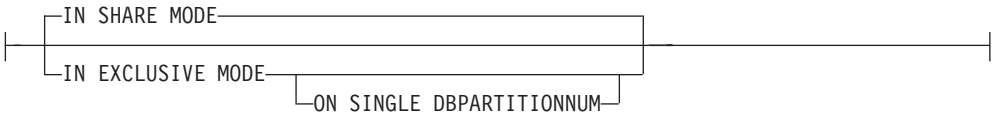
La selección entre el Tipo 1 y el Tipo 2 se determina según las opciones de precompilador. Encontrará una visión general de estas opciones en el apartado "Bases de datos relacionales distribuidas".



autorización:



bloque-bloqueo:



Notas:

1 Este formato sólo es válido si se ha habilitado la conexión implícita.

Descripción:

TO *nombre-servidor/variable-lengpral*
Las normas para codificar el nombre del servidor son las mismas que para el Tipo 1.

Si la opción SQLRULES(STD) está en vigor, el *nombre-servidor* no debe identificar una conexión existente del proceso de aplicación; de lo contrario, se produce un error (SQLSTATE 08002).

Si la opción SQLRULES(DB2) está en vigor y el *nombre-servidor* identifica una conexión existente del proceso de aplicación, esa conexión se convierte en la actual y la conexión anterior se coloca en estado inactivo. Es decir, el efecto de la sentencia CONNECT en esta situación es el mismo que el de la sentencia SET CONNECTION.

Para obtener información acerca de la especificación de SQLRULES, consulte el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida".

Conexión satisfactoria

Si la sentencia CONNECT TO es satisfactoria:

- Se crea (o deja de estar inactiva) una conexión con el servidor de aplicaciones y se coloca en estado actual y mantenido.
- Si CONNECT TO se dirige a un servidor diferente del actual, la conexión actual se coloca en estado inactivo.
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma forma que con CONNECT (Tipo 1).

Conexión no satisfactoria

Si la sentencia CONNECT TO no es satisfactoria:

- No importa cuál haya sido la razón de la anomalía, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.
- Al igual que para un CONNECT de Tipo 1 no satisfactorio, el campo SQLERRP de la SQLCA se establece en el nombre del módulo en el peticionario o servidor de aplicaciones que ha detectado el error.

CONNECT (sin ningún operando)**, IN SHARE/EXCLUSIVE MODE, USER,** y **USING**
Si existe una conexión, el Tipo 2 se comporta como un Tipo 1. El ID de autorización y el alias de base de datos se colocan en el campo SQLERRMC de la SQLCA. Si no existe una conexión, no se realiza ningún intento de establecer una conexión implícita y los campos SQLERRP y SQLERRMC devuelven un blanco. (Las aplicaciones pueden comprobarse si existe una conexión actual comprobando estos campos.)

CONNECT sin operandos que incluya USER y USING todavía puede conectar un proceso de aplicación a una base de datos mediante la variable de entorno DB2DBDFT. Este método es equivalente a CONNECT RESET Tipo 2, pero está permitido utilizar un ID de usuario y una contraseña.

RESET
Equivale a una conexión explícita con la base de datos por omisión, si está disponible. Si no hay una base de datos por omisión que esté disponible, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.

La disponibilidad de una base de datos por omisión se determina de acuerdo con las opciones de instalación, las variables de entorno y los valores de autentificación.

Normas:

- Como se describe en el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida", un conjunto de opciones rige la semántica de la gestión de las conexiones. Se asignan valores por omisión a todos los archivos fuente preprocesados. Una aplicación puede constar de múltiples archivos fuente precompilados con diferentes opciones de conexión.

  A menos que primero se haya ejecutado un mandato SET CLIENT o una API, las opciones de conexión que se utilizan al procesar previamente el archivo fuente que contiene la primera sentencia de SQL que se ejecuta en tiempo de ejecución se convierten en las opciones de conexión que están en vigor.

  Si posteriormente se ejecuta una sentencia CONNECT desde un archivo fuente previamente procesado con opciones de conexión distintas sin que intervenga en la ejecución ningún mandato SET CLIENT o la API, se devuelve un error (SQLSTATE 08001). Observe que, una vez que se ha ejecutado un mandato SET CLIENT o una API, se pasan por alto las opciones de conexión utilizadas al preprocesar todos los archivos fuente de la aplicación.

  En el Ejemplo 1 del apartado "Ejemplos" de esta sentencia se muestran estas normas.
- Aunque la sentencia CONNECT TO puede utilizarse para establecer o conmutar conexiones, sólo se aceptará CONNECT TO con la cláusula USER/USING cuando no haya ninguna conexión actual o inactiva con el servidor nombrado. La conexión debe liberarse antes de emitir una conexión con el mismo servidor con la cláusula USER/USING; de lo contrario, se rechazará (SQLSTATE 51022). Libere la conexión emitiendo una sentencia DISCONNECT o una sentencia RELEASE seguida de una sentencia COMMIT.

Notas:

- Se da soporte a la conexión implícita para la primera sentencia de SQL en una aplicación con conexiones de Tipo 2. Para ejecutar sentencias de SQL en la base de datos por omisión, primero debe utilizarse la sentencia CONNECT RESET o CONNECT USER/USING para establecer la conexión. La sentencia CONNECT sin operandos visualizará información acerca de la conexión actual si la hay, pero no establecerá una conexión con la base de datos por omisión si no hay ninguna conexión actual.
- El *nombre-autorización* SYSTEM no se puede especificar explícitamente en la sentencia CONNECT. Sin embargo, en sistemas operativos Windows, las aplicaciones locales que se ejecutan bajo la cuenta del sistema local se pueden conectar implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Al conectar a un servidor Windows explícitamente, se puede especificar el *nombre-autorización* o la *variable-sistema-principal* del usuario si se utiliza el

nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft Windows NT; por ejemplo, 'Dominio\Usuario'.

Comparación de las sentencias CONNECT de Tipo 1 y de Tipo 2:

La semántica de la sentencia CONNECT la determina la opción de precompilador CONNECT o la API SET CLIENT (consulte el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida"). Puede especificarse CONNECT Tipo 1 o CONNECT Tipo 2, y las sentencias CONNECT en esos programas se conocen como sentencias CONNECT Tipo 1 y Tipo 2 respectivamente. Su semántica se describe a continuación:

Utilización de **CONNECT TO**:

Tipo 1	Tipo 2
Cada unidad de trabajo sólo puede establecer conexión con un servidor de aplicaciones.	Cada unidad de trabajo puede establecer conexión con múltiples servidores de aplicaciones.
Se debe confirmar o retrotraer la unidad de trabajo actual antes de permitir una conexión con otro servidor de aplicaciones.	No es necesario confirmar ni retrotraer la unidad de trabajo actual antes de conectar con otro servidor de aplicaciones.
La sentencia CONNECT establece la conexión actual. Las peticiones SQL posteriores se reenvían a esta conexión hasta que otra CONNECT la modifique.	Igual que CONNECT Tipo 1 si se establece la primera conexión. Si se conmuta a una conexión inactiva y SQLRULES está establecido en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.
Conectar con la conexión actual es válido y no modifica la conexión actual.	Igual que CONNECT Tipo 1 si la opción de precompilador SQLRULES está establecida en DB2. Si SQLRULES está establecida en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.
Conectar con otro servidor de aplicaciones desconecta la conexión actual. La nueva conexión se convierte en la conexión actual. En una unidad de trabajo sólo se mantiene una conexión.	Conectar con otro servidor de aplicaciones pone la conexión actual en el *estado inactivo*. La nueva conexión se convierte en la conexión actual. Pueden mantenerse múltiples conexiones en una unidad de trabajo. Si CONNECT se ejecuta para un servidor de aplicaciones que está en una conexión inactiva, ésta se convierte en la conexión actual. Conectar con una conexión inactiva mediante CONNECT sólo está permitido si se ha especificado SQLRULES(DB2). Si se ha especificado SQLRULES(STD), debe utilizarse la sentencia SET CONNECTION en su lugar.
No se da soporte a la sentencia SET CONNECTION para conexiones de Tipo 1, pero el único destino válido es la conexión actual.	Se da soporte a la sentencia SET CONNECTION en conexiones de Tipo 2 para cambiar el estado de una conexión inactiva a actual.

Utilización de **CONNECT...USER...USING**:

Tipo 1	Tipo 2
Conectar con la cláusula USER...USING desconecta la conexión actual y establece una nueva conexión con el nombre de autorización y la contraseña proporcionados.	Conectar con la cláusula USER/USING sólo se aceptará cuando no haya una conexión actual o inactiva con el mismo servidor indicado.

Utilización de **CONNECT, CONNECT RESET implícitas** y **Desconexión**:

Tipo 1	Tipo 2
CONNECT RESET puede utilizarse para desconectar la conexión actual.	CONNECT RESET equivale a conectar explícitamente con el servidor de aplicaciones por omisión si hay uno definido en el sistema. La aplicación puede desconectar las conexiones en una COMMIT satisfactoria. Antes de la confirmación, utilice la sentencia RELEASE para marcar una conexión como pendiente de liberación. Tales conexiones se desconectarán en la siguiente COMMIT. Una alternativa consiste en utilizar las opciones de precompilador DISCONNECT(EXPLICIT), DISCONNECT(CONDITIONAL), DISCONNECT(AUTOMATIC) o la sentencia DISCONNECT en lugar de la sentencia RELEASE.
Tras utilizar CONNECT RESET para desconectar la conexión actual, si la siguiente sentencia de SQL no es una sentencia CONNECT, realizará una conexión implícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.	CONNECT RESET equivale a una conexión explícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.
Es un error emitir sentencias CONNECT RESET consecutivas.	SÓLO es un error emitir sentencias CONNECT RESET consecutivas si se ha especificado SQLRULES(STD), porque esta opción inhabilita el uso de CONNECT para la conexión existente.
CONNECT RESET también confirma implícitamente la unidad de trabajo actual.	CONNECT RESET no confirma la unidad de trabajo actual.
Si el sistema desconecta una conexión existente por cualquier motivo, las sentencias de SQL posteriores que no sean CONNECT para esta base de datos recibirán un SQLSTATE de 08003.	Si el sistema desconecta una conexión existente, siguen estando permitidas las sentencias COMMIT, ROLLBACK y SET CONNECTION.
La unidad de trabajo se confirmará implícitamente cuando el proceso de aplicación termine de manera satisfactoria.	Igual que el Tipo 1.
Todas las conexiones (sólo una) se desconectan cuando el proceso de aplicación termina.	Todas las conexiones (actuales, inactivas y las marcadas como pendientes de liberación) se desconectan cuando el proceso de aplicación termina.

Anomalías de CONNECT:

Tipo 1	Tipo 2
Sin tener en cuenta si hay una conexión actual cuando CONNECT falla (con un error que no sea que el nombre-servidor no está definido en el directorio local), el proceso de aplicación se coloca en estado no conectado. Las sentencias posteriores que no sean CONNECT recibirán un SQLSTATE de 08003.	Si hay una conexión actual cuando CONNECT falla, la conexión actual no se verá afectada. Si no hay una conexión actual cuando CONNECT falla, el programa pasa a estar en estado no conectado. Las sentencias posteriores que no sean CONNECT recibirán un SQLSTATE de 08003.

Ejemplos:

Ejemplo 1:

En este ejemplo se muestra la utilización de varios programas fuente (se muestran en los recuadros), algunos procesados previamente con opciones de conexión distintas (se muestran sobre el código) y uno de los cuales contiene una llamada a la API SET CLIENT.

PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
exec sql CONNECT TO OTTAWA;
exec sql SELECT col1 INTO :hv1
FROM tbl1;
...
```

PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)

```
...
exec sql CONNECT TO QUEBEC;
exec sql SELECT col1 INTO :hv1
FROM tbl2;
...
```

PGM3: CONNECT(2) SQLRULES(STD) DISCONNECT(EXPLICIT)

```
...
SET CLIENT CONNECT 2  SQLRULES DB2  DISCONNECT EXPLICIT [1]
exec sql CONNECT TO LONDON;
exec sql SELECT col1 INTO :hv1
FROM tbl3;
...
```

1 Nota: no es la sintaxis real de la API SET CLIENT

PGM4: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
exec sql CONNECT TO REGINA;
exec sql SELECT col1 INTO :hv1
FROM tbl4;
...
```

Si la aplicación ejecuta PGM1 y luego PGM2:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con QUEBEC es anómala con SQLSTATE 08001 porque SQLRULES y DISCONNECT son diferentes.

Si la aplicación ejecuta PGM1 y luego PGM3:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con LONDON ejecuta: connect=2, sqlrules=DB2, disconnect=EXPLICIT

Esto es correcto porque la API SET CLIENT se ejecuta antes que la segunda sentencia CONNECT.

Si la aplicación ejecuta PGM1 y luego PGM4:

- la conexión con OTTAWA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL
- la conexión con REGINA ejecuta: connect=2, sqlrules=DB2, disconnect=CONDITIONAL

Esto es correcto porque las opciones de preprocesador para PGM1 son las mismas que para PGM4.

Ejemplo 2:

Este ejemplo muestra las interrelaciones entre las sentencias CONNECT (Tipo 2), SET CONNECTION, RELEASE y DISCONNECT. S0, S1, S2 y S3 representan cuatro servidores.

Secuencia	Sentencia	Servidor actual	Conexiones inactivas	Pendiente de liberación
0	Ninguna sentencia	Ninguno	Ninguno	Ninguno
1	SELECT * FROM TBLA	S0 (valor por omisión)	Ninguno	Ninguno
2	CONNECT TO S1	S1	S0	Ninguno
	SELECT * FROM TBLB	S1	S0	Ninguno
3	CONNECT TO S2	S2	S0, S1	Ninguno
	UPDATE TBLC SET ...	S2	S0, S1	Ninguno
4	CONNECT TO S3	S3	S0, S1, S2	Ninguno
	SELECT * FROM TBLD	S3	S0, S1, S2	Ninguno
5	SET CONNECTION S2	S2	S0, S1, S3	Ninguno
6	RELEASE S3	S2	S0, S1	S3
7	COMMIT	S2	S0, S1	Ninguno
8	SELECT * FROM TBLE	S2	S0, S1	Ninguno
9	DISCONNECT S1	S2	S0	Ninguno
	SELECT * FROM TBLF	S2	S0	Ninguno

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*

Información relacionada:

- "CONNECT (Tipo 1)" en la página 148

Ejemplos relacionados:

- "dbmcon.sqc -- How to use multiple databases (C)"
- "dbmcon.sqC -- How to use multiple databases (C++)"

CREATE ALIAS

La sentencia CREATE ALIAS define un alias para una tabla, una vista, un apodo u otro alias.

Invocación:

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

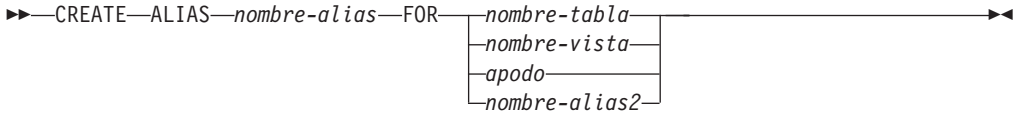
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del alias no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema del alias hace referencia a un esquema existente.

Para utilizar el objeto referenciado a través del alias, son necesarios los mismos privilegios para el objeto que los que serían necesarios si se utilizara el propio objeto.

Sintaxis:



Descripción:

nombre-alias
Indica el nombre del alias. El nombre no debe identificar una tabla, una vista, un apodo o un alias que exista en la base de datos actual.

Si se especifica un nombre compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS' (SQLSTATE 42939).

Las normas que se emplean para definir un alias son las mismas que las que se emplean para definir un nombre de tabla.

FOR *nombre-tabla, nombre-vista, apodo,* o *nombre-alias2*
Identifica la tabla, la vista, el apodo o el alias para el que se ha definido *nombre-alias*. En caso de proporcionar otro alias (*nombre-alias2*), éste no debe ser el mismo que el nuevo *nombre-alias* que se está definiendo (en su forma completamente calificada). El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - SYNONYM puede especificarse en lugar de ALIAS

- La definición del alias recién creado se almacena en SYSCAT.TABLES.
- Se puede definir un alias para un objeto que no exista en el momento de la definición. Si no existe, aparece un mensaje de aviso (SQLSTATE 01522). No obstante, el objeto al que se hace referencia debe existir al compilar una sentencia de SQL que contenga dicho alias; de lo contrario, aparece un mensaje de error (SQLSTATE 52004).
- Se puede definir un alias para hacer referencia a otro alias como parte de una cadena de alias, pero dicha cadena está sujeta a las mismas restricciones que un alias normal cuando se utiliza en una sentencia de SQL. La cadena de alias se resuelve de la misma manera que un solo alias. Si un alias que se utiliza en una definición de vista, en una sentencia de un paquete o en un activador apunta a una cadena de alias, esta relación de dependencia queda registrada para la vista, paquete o activador en cada uno de los alias de la cadena. Los ciclos repetidos no están permitidos en una cadena de alias y se detectan durante el proceso de definición.
- La creación de un alias con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

Ejemplos:

Ejemplo 1: HEDGES intenta crear un alias para una tabla T1 (ambos sin calificar).

```
CREATE ALIAS A1 FOR T1
```

Se crea el alias HEDGES.A1 para HEDGES.T1.

Ejemplo 2: HEDGES intenta crear un alias para una tabla (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

Se crea el alias HEDGES.A1 para MCKNIGHT.T1.

Ejemplo 3: HEDGES intenta crear un alias para una tabla (el alias en un esquema diferente; HEDGES no es DBADM; HEDGES no tiene el privilegio CREATEIN para el esquema MCKNIGHT).

```
CREATE ALIAS  MCKNIGHT.A1 FOR MCKNIGHT.T1
```

Este ejemplo falla (SQLSTATE 42501).

Ejemplo 4: HEDGES intenta crear un alias para una tabla no definida (ambos calificados; FUZZY.WUZZY no existe).

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

Esta sentencia resulta satisfactoria pero emite un mensaje de aviso (SQLSTATE 01522).

Ejemplo 5: HEDGES intenta crear un alias para un alias (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

La primera sentencia resulta satisfactoria (como en el ejemplo 2).

La segunda sentencia es satisfactoria y crea una cadena de alias, según la cual HEDGES.A2 hace referencia a HEDGES.A1, que a su vez hace referencia a MCKNIGHT.T1. Observe que no importa si HEDGES tiene privilegios o no sobre MCKNIGHT.T1. El alias se crea sean cuales sean los privilegios de las tablas.

Ejemplo 6: Designar A1 como alias para el apodo FUZZYBEAR.

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

Ejemplo 7: Una gran organización tiene un departamento financiero con el número D108 y un departamento de personal con el número D577. D108 conserva determinada información en una tabla que reside en DB2 RDBMS. D577 conserva determinados registros en una tabla que reside en Oracle RDBMS. Un DBA define los dos RDBMS como fuentes de datos de un sistema federado y asigna a las tablas los apodos de DEPTD108 y DEPTD577, respectivamente. El usuario de un sistema federado necesita crear vínculos entre estas tablas, pero preferiría hacer referencia a ellas por los nombres que tienen más sentido que sus apodos alfanuméricos. Por lo tanto, el usuario define FINANCE como un alias para DEPTD108 y PERSONNEL como un alias para DEPTD577.

```
CREATE ALIAS FINANCE FOR DEPTD108
CREATE ALIAS PERSONNEL FOR DEPTD577
```

CREATE BUFFERPOOL

La sentencia CREATE BUFFERPOOL crea una agrupación de almacenamientos intermedios nueva para que la utilice el gestor de bases de datos.

En una base de datos particionada, se especifica una definición de agrupación de almacenamientos intermedios por omisión para cada partición, existiendo la posibilidad de alterar temporalmente el tamaño de particiones específicas. Asimismo, en una base de datos particionada, la agrupación de almacenamientos intermedios se define en todas las particiones a menos que se especifiquen grupos de particiones de base de datos. Si se especifican grupos de particiones de base de datos, la agrupación de almacenamientos intermedios sólo se creará en las particiones que se encuentran en esos grupos de particiones de base de datos.

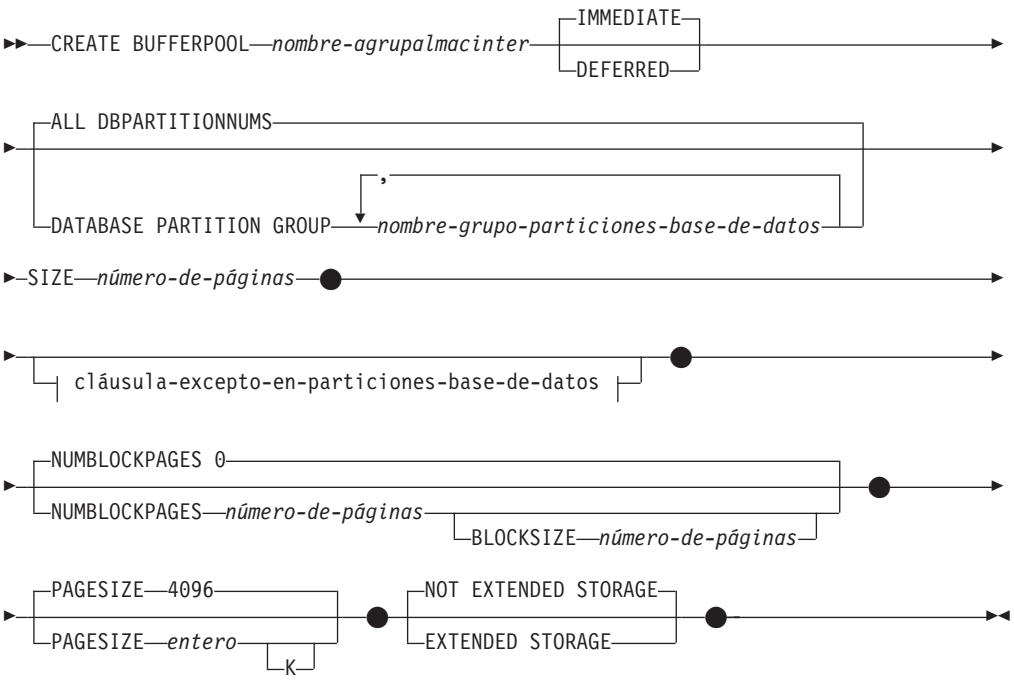
Invocación:

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

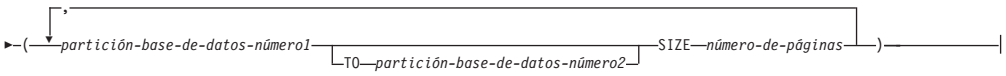
Autorización:

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

Sintaxis:



cláusula-excepto-en-particiones-base-de-datos:



Descripción:

nombre-agrupalmacinter
Indica el nombre de la agrupación de almacenamientos intermedios. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-agrupalmcinter* no debe identificar una agrupación de almacenamientos intermedios que ya exista en un catálogo (SQLSTATE 42710). El *nombre-agrupalmacinter* no debe empezar por los caracteres 'SYS' e 'IBM' (SQLSTATE 42939).

IMMEDIATE
La agrupación de almacenamientos intermedios se creará inmediatamente. Si no existe suficiente espacio reservado en la memoria compartida de base de datos para asignar la nueva agrupación de almacenamientos intermedios, se devolverá un mensaje de aviso (SQLSTATE 01657) y la sentencia se ejecutará como DEFERRED.

DEFERRED
La agrupación de almacenamientos intermedios se creará cuando se desactive la base de datos (es necesario desconectar todas las aplicaciones de la base de datos). No es necesario espacio de memoria reservado; DB2 asignará la memoria necesaria del sistema.

ALL DBPARTITIONNUMS
Esta agrupación de almacenamientos intermedios se creará en todas las particiones de la base de datos.

DATABASE PARTITION GROUP *nombre-grupo-particiones-base-de-datos, ...*
Identifica el grupo o grupos de particiones de base de datos a los que se aplica la definición de agrupación de almacenamientos intermedios. Si se especifica, esta agrupación de almacenamientos intermedios sólo se creará en las particiones de esos grupos de particiones de base de datos. Cada grupo de particiones de base de datos debe existir actualmente en la base de datos (SQLSTATE 42704). Si no se han especificado las palabras clave DATABASE PARTITION GROUP, esta agrupación de almacenamientos intermedios se creará en todas las particiones (y en cualquier partición que posteriormente se añada a la base de datos).

SIZE *número-de-páginas*
El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas. En una base de datos particionada, será el tamaño por omisión para todas las particiones en las que exista la agrupación de almacenamientos intermedios.

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

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

cláusula-excepto-en-particiones-base-de-datos
Especifica la partición o particiones para las que el tamaño de la agrupación de almacenamientos intermedios será diferente del valor por omisión. Si no se especifica esta cláusula, entonces todas las particiones tendrán el mismo tamaño que el especificado para esta agrupación de almacenamientos intermedios.

EXCEPT ON DBPARTITIONNUMS
Palabras clave que indican que se han especificado particiones específicas. DBPARTITIONNUM es un sinónimo de DBPARTITIONNUMS.

partición-base-de-datos-número1
Especifica un número de partición específica que se incluye en las particiones para las que se crea la agrupación de almacenamientos intermedios.

TO *partición-base-de-datos-número2*
Especifique un rango de números de partición. El valor de *partición-base-de-datos-número2* debe ser mayor que o igual al valor de *partición-base-de-datos-número1* (SQLSTATE 428A9). Todas las particiones entre los números de partición, inclusive los especificados, deben incluirse en las particiones para las que se ha creado la agrupación de almacenamientos intermedios (SQLSTATE 42729).

SIZE *número-de-páginas*
El tamaño de agrupación de almacenamientos intermedios especificado como el número de páginas.

PAGESIZE *entero* **[K]**
Define el tamaño de las páginas utilizadas para la agrupación de almacenamientos intermedios. Los valores válidos de *entero* sin el sufijo K son 4 096, 8 192, 16 384 ó 32 768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE). El valor por omisión es páginas de 4 096 bytes (4K). Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio.

EXTENDED STORAGE
Si el almacenamiento ampliado está habilitado, las páginas que se eliminen de esta agrupación de almacenamientos intermedios se colocarán en antememoria en el almacenamiento ampliado. (El almacenamiento ampliado se habilita estableciendo los parámetros de configuración de base de datos NUM_ESTORE_SEGS y ESTORE_SEG_SIZE en valores distintos de cero.)

NOT EXTENDED STORAGE
Aunque el almacenamiento ampliado esté habilitado, las páginas que se eliminan de esta agrupación de almacenamientos intermedios no se colocan en antememoria en el almacenamiento ampliado.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - NODES puede especificarse en lugar de DBPARTITIONNUMS
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
- Si la agrupación de almacenamientos intermedios se crea mediante la utilización de la opción DEFERRED, cualquier espacio de tablas que se cree en esta

agrupación de almacenamientos intermedios utilizará una pequeña agrupación de almacenamientos intermedios del sistema con el mismo tamaño de página hasta la próxima vez que se active la base de datos. La base de datos deberá reiniciarse para que la agrupación de almacenamientos intermedios se active y para que entren en vigor las asignaciones de espacio de tablas para la nueva agrupación de almacenamientos intermedios. La opción por omisión es IMMEDIATE.

- No puede crearse una agrupación de almacenamientos intermedios con almacenamiento ampliado y también con soporte basado en bloques.
- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones. Si DB2 no puede obtener memoria para las agrupaciones de almacenamientos intermedios normales, intentará iniciarse con pequeñas agrupaciones de almacenamientos intermedios del sistema, una para cada tamaño de página (4K, 8K, 16K y 32K). Ante esta situación, se enviará un mensaje de aviso al usuario (SQLSTATE 01626) y las páginas de todos los espacios de tabla utilizarán las agrupaciones de almacenamientos intermedios del sistema.

Información relacionada:

- "database_memory - Database shared memory size configuration parameter" en la publicación *Administration Guide: Performance*

Ejemplos relacionados:

- "tscreate.sqc -- How to create and drop buffer pools and table spaces (C)"
- "tscreate.sqC -- How to create and drop buffer pools and table spaces (C++)"

CREATE DATABASE PARTITION GROUP

La sentencia CREATE DATABASE PARTITION GROUP crea un nuevo grupo de particiones de base de datos dentro de la base de datos, asigna particiones al grupo de particiones de base de datos y graba la definición del grupo de particiones de base de datos en el catálogo.

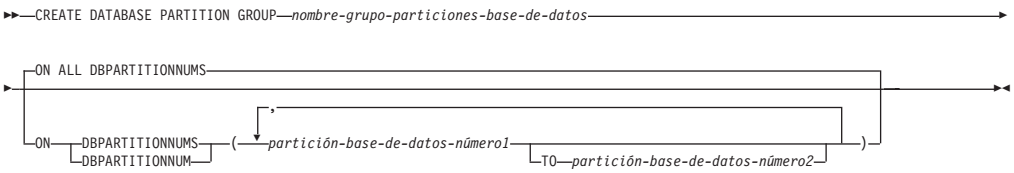
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

nombre-grupo-particiones-base-de-datos
Indica el nombre del grupo de particiones de base de datos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-grupo-particiones-base-de-datos* no debe identificar un grupo de particiones de base de datos que ya exista en el catálogo (SQLSTATE 42710). El *nombre-grupo-particiones-base-de-datos* no debe empezar por los caracteres 'SYS' o 'IBM' (SQLSTATE 42939).

ON ALL DBPARTITIONNUMS
Especifica que el grupo de particiones de base de datos se define en todas las particiones que estaban definidas en la base de datos (archivo `db2nodes.cfg`) en el momento de crearse el grupo de particiones de base de datos.

Si se añade una partición al sistema de base de datos, debe emitirse la sentencia ALTER DATABASE PARTITION GROUP para incluir esta nueva partición en un grupo de particiones de base de datos (incluido IBMDEFAULTGROUP). Además, también debe emitirse el mandato REDISTRIBUTE DATABASE PARTITION GROUP para mover los datos a la partición.

ON DBPARTITIONNUMS
Especifica las particiones específicas que se encuentran en el grupo de particiones de base de datos. DBPARTITIONNUM es un sinónimo de DBPARTITIONNUMS.

partición-base-de-datos-número1
Especifique un número de partición determinado. (Para mantener la compatibilidad con la versión anterior, puede especificarse un *nombre-nodo* en el formato `NODE`*nnnnn*`.`)

TO *partición-base-de-datos-número2*
Especifique un rango de números de partición. El valor de *partición-base-de-datos-número2* debe ser mayor que o igual al valor de *partición-base-de-datos-número1* (SQLSTATE 428A9). Todas las particiones que se encuentren entre los números de partición especificados, éstos inclusive, se incluirán en el grupo de particiones de base de datos.

Normas:

- Cada partición que se haya especificado mediante un número deberá definirse en el archivo `db2nodes.cfg` (SQLSTATE 42729).
- Cada *número-partición-base-de-datos* de la lista de la cláusula ON DBPARTITIONNUMS deberá aparecer una vez como máximo (SQLSTATE 42728).
- Un *número-partición-base-de-datos* válido es un número comprendido entre 0 y 999, ambos inclusive (SQLSTATE 42729).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - NODES puede especificarse en lugar de DBPARTITIONNUMS
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
- Esta sentencia crea una correlación de particionamiento para el grupo de particiones de base de datos. Se genera un identificador de mapa de particionamiento (PMAP_ID) para cada mapa de particionamiento. Esta información se graba en el catálogo y puede recuperarse desde SYSCAT.DBPARTITIONGROUPS y SYSCAT.PARTITIONMAPS. Cada entrada del mapa de particionamiento especifica la partición de destino donde residen todas las filas que se generan aleatoriamente. Para un grupo de particiones de base de datos de una sola partición, la correlación de particionamiento correspondiente sólo tiene una entrada. Para un grupo de particiones de base de datos de varias particiones, la correlación de particionamiento correspondiente tiene 4 096 entradas donde, por omisión, los números de partición se asignan a las entradas de la correlación rotativamente.

Ejemplos:

Suponga que tiene una base de datos particionada con seis particiones definidas como: 0, 1, 2, 5, 7 y 8.

- Supongamos que desea crear un grupo de particiones de base de datos denominado MAXGROUP en las seis particiones. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- Supongamos que desea crear un grupo de particiones de base de datos denominado MEDGROUP en las particiones 0, 1, 2, 5 y 8. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- Supongamos que desea crear un grupo de particiones de base de datos de una sola partición denominado MINGROUP en la partición 7. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

Conceptos relacionados:

- "Particionamiento de datos entre múltiples particiones" en la publicación *Consulta de SQL, Volumen 1*

CREATE DISTINCT TYPE

La sentencia CREATE DISTINCT TYPE define un tipo diferenciado. El tipo diferenciado siempre tiene su origen en los tipos de datos incorporados. La ejecución satisfactoria de la sentencia también genera funciones para la conversión entre el tipo diferenciado y su tipo de fuente y, opcionalmente, genera el soporte para utilizar los operadores de comparación (=, <>, <, <=, > y >=) con el tipo diferenciado.

Invocación:

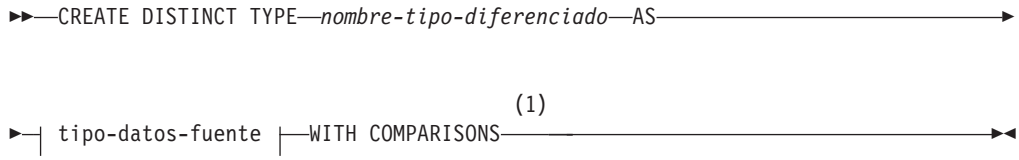
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

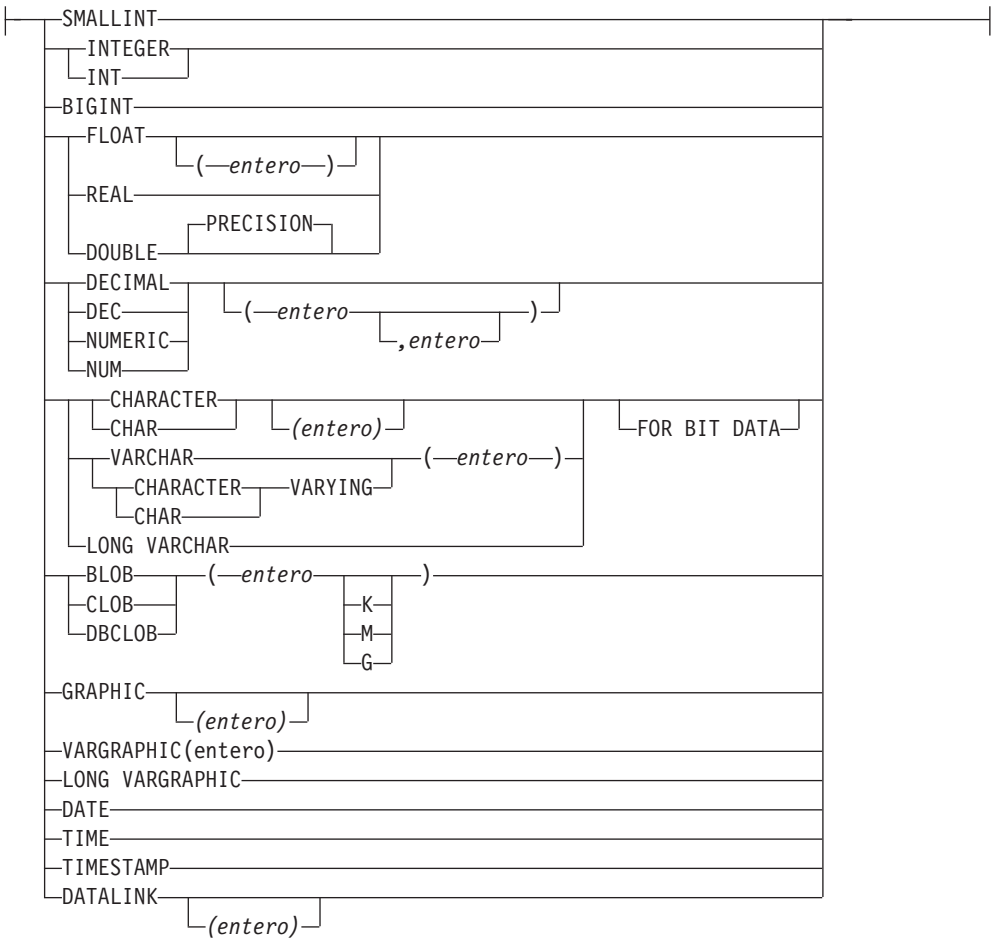
El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo diferenciado no hace referencia a un esquema existente.
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo diferenciado hace referencia a un esquema existente.

Sintaxis:



tipo-datos-fuente:



Notas:

1 Necesario para todos los tipos-datos-fuente excepto LOB, LONG VARCHAR, LONG VARGRAPHIC y DATALINK, a los que no se da soporte.

Descripción:

nombre-tipo-diferenciado
Indica el nombre del tipo diferenciado. El nombre (incluido el calificador implícito o explícito) no debe designar un tipo diferenciado descrito en el catálogo. El nombre no calificado no debe ser igual que el nombre de un tipo de datos fuente ni BOOLEAN (SQLSTATE 42918).

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre de esquema (implícito o explícito) no debe ser mayor que 8 bytes (SQLSTATE 42622).

Algunos nombres utilizados como palabras clave en predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-tipo-diferenciado*. Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR,

MATCH y los operadores de comparación. Cualquier incumplimiento de esta norma provocará la aparición de un error (SQLSTATE 42939).

Si se especifica un *nombre-tipo-diferenciado* compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS'; de lo contrario, se generará un error (SQLSTATE 42939).

tipo-datos-fuente
Especifica el tipo de datos utilizado como base para la representación interna del tipo diferenciado.

WITH COMPARISONS
Especifica que se deben crear los operadores de comparación generados por el sistema para comparar dos instancias de un tipo diferenciado. Estas palabras clave no deben especificarse si el tipo-datos-fuente es BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC o DATALINK; de lo contrario, se devolverá un aviso (SQLSTATE 01596) y no se generarán los operadores de comparación. Para todos los demás tipos-datos-fuente, son necesarias las palabras clave WITH COMPARISONS.

Notas:

- *Privilegios*

  El usuario que define el tipo definido por el usuario siempre recibe el privilegio EXECUTE y WITH GRANT OPTION para todas las funciones que automáticamente se generan para el tipo diferenciado.

  A PUBLIC se otorga el privilegio EXECUTE para todas las funciones que automáticamente se generan durante CREATE DISTINCT TYPE.
- La creación de un tipo diferenciado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Se generan las funciones siguientes para convertir el tipo de datos utilizando el tipo de fuente como origen o destino de la conversión:
  - Una función para convertir del tipo diferenciado al tipo de fuente
  - Una función para convertir del tipo de fuente al tipo diferenciado
  - Una función para convertir de INTEGER al tipo diferenciado si el tipo de fuente es SMALLINT
  - Una función para convertir de VARCHAR al tipo diferenciado si el tipo de fuente es CHAR
  - Una función para convertir de VARGRAPHIC al tipo diferenciado si el tipo de fuente es GRAPHIC.

En general estas funciones tendrán el siguiente formato:

```
CREATE FUNCTION nombre-tipo-fuente (nombre-tipo-diferenciado)
   RETURNS nombre-tipo-fuente ...

CREATE FUNCTION nombre-tipo-diferenciado (nombre-tipo-fuente)
  RETURNS nombre-tipo-diferenciado ...
```

En los casos en que el tipo de fuente es un tipo con parámetros, la función para convertir el tipo diferenciado al tipo de fuente tendrá como nombre de función el nombre del tipo de fuente sin los parámetros (consulte los detalles en la Tabla 3 en la página 176). El tipo de valor de retorno de esta función incluirá los parámetros dados en la sentencia CREATE DISTINCT TYPE. La función para convertir del tipo

de fuente al tipo diferenciado tendrá un parámetro de entrada cuyo tipo es el tipo de fuente incluyendo sus parámetros. Por ejemplo,

```
CREATE DISTINCT TYPE T_SHOESIZE AS CHAR(2)
          WITH COMPARISONS

CREATE DISTINCT TYPE T_MILES AS DOUBLE
          WITH COMPARISONS
```

generará las funciones siguientes:

```
FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)

FUNCTION T_SHOESIZE (CHAR (2))
RETURNS T_SHOESIZE

FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE

FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

El esquema de las funciones de conversión generadas es el mismo que el esquema del tipo diferenciado. No debe existir ninguna otra función con este nombre y con la misma signatura en la base de datos (SQLSTATE 42710).

La tabla siguiente ofrece los nombres de las funciones para la conversión del tipo diferenciado al tipo de fuente y del tipo de fuente al tipo diferenciado para todos los tipos de datos predefinidos.

Tabla 3. Funciones CAST en tipos diferenciados

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
CHAR	*nombre-tipo-diferenciado*	CHAR (*n*)	*nombre-tipo-diferenciado*
	CHAR	*nombre-tipo-diferenciado*	CHAR (*n*)
	nombre-tipo-diferenciado	VARCHAR (*n*)	*nombre-tipo-diferenciado*
VARCHAR	*nombre-tipo-diferenciado*	VARCHAR (*n*)	*nombre-tipo-diferenciado*
	VARCHAR	*nombre-tipo-diferenciado*	VARCHAR (*n*)
LONG VARCHAR	*nombre-tipo-diferenciado*	LONG VARCHAR	*nombre-tipo-diferenciado*
	LONG_VARCHAR	*nombre-tipo-diferenciado*	LONG VARCHAR
CLOB	*nombre-tipo-diferenciado*	CLOB (*n*)	*nombre-tipo-diferenciado*
	CLOB	*nombre-tipo-diferenciado*	CLOB (*n*)
BLOB	*nombre-tipo-diferenciado*	BLOB (*n*)	*nombre-tipo-diferenciado*
	BLOB	*nombre-tipo-diferenciado*	BLOB (*n*)
GRAPHIC	*nombre-tipo-diferenciado*	GRAPHIC (*n*)	*nombre-tipo-diferenciado*
	GRAPHIC	*nombre-tipo-diferenciado*	GRAPHIC (*n*)
	nombre-tipo-diferenciado	VARGRAPHIC (*n*)	*nombre-tipo-diferenciado*
VARGRAPHIC	*nombre-tipo-diferenciado*	VARGRAPHIC (*n*)	*nombre-tipo-diferenciado*
	VARGRAPHIC	*nombre-tipo-diferenciado*	VARGRAPHIC (*n*)
LONG VARGRAPHIC	*nombre-tipo-diferenciado*	LONG VARGRAPHIC	*nombre-tipo-diferenciado*
	LONG_VARGRAPHIC	*nombre-tipo-diferenciado*	LONG VARGRAPHIC
DBCLOB	*nombre-tipo-diferenciado*	DBCLOB (*n*)	*nombre-tipo-diferenciado*
	DBCLOB	*nombre-tipo-diferenciado*	DBCLOB (*n*)

Tabla 3. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
SMALLINT	*nombre-tipo-diferenciado*	SMALLINT	*nombre-tipo-diferenciado*
	nombre-tipo-diferenciado	INTEGER	*nombre-tipo-diferenciado*
	SMALLINT	*nombre-tipo-diferenciado*	SMALLINT
INTEGER	*nombre-tipo-diferenciado*	INTEGER	*nombre-tipo-diferenciado*
	INTEGER	*nombre-tipo-diferenciado*	INTEGER
BIGINT	*nombre-tipo-diferenciado*	BIGINT	*nombre-tipo-diferenciado*
	BIGINT	*nombre-tipo-diferenciado*	BIGINT
DECIMAL	*nombre-tipo-diferenciado*	DECIMAL (*p*,*s*)	*nombre-tipo-diferenciado*
	DECIMAL	*nombre-tipo-diferenciado*	DECIMAL (*p*,*s*)
NUMERIC	*nombre-tipo-diferenciado*	DECIMAL (*p*,*s*)	*nombre-tipo-diferenciado*
	DECIMAL	*nombre-tipo-diferenciado*	DECIMAL (*p*,*s*)
REAL	*nombre-tipo-diferenciado*	REAL	*nombre-tipo-diferenciado*
	nombre-tipo-diferenciado	DOUBLE	*nombre-tipo-diferenciado*
	REAL	*nombre-tipo-diferenciado*	REAL
FLOAT(*n*) donde *n*<=24	*nombre-tipo-diferenciado*	REAL	*nombre-tipo-diferenciado*
	nombre-tipo-diferenciado	DOUBLE	*nombre-tipo-diferenciado*
	REAL	*nombre-tipo-diferenciado*	REAL
FLOAT(*n*) donde *n*>24	*nombre-tipo-diferenciado*	DOUBLE	*nombre-tipo-diferenciado*
	DOUBLE	*nombre-tipo-diferenciado*	DOUBLE
FLOAT	*nombre-tipo-diferenciado*	DOUBLE	*nombre-tipo-diferenciado*
	DOUBLE	*nombre-tipo-diferenciado*	DOUBLE
DOUBLE	*nombre-tipo-diferenciado*	DOUBLE	*nombre-tipo-diferenciado*
	DOUBLE	*nombre-tipo-diferenciado*	DOUBLE
DOUBLE PRECISION	*nombre-tipo-diferenciado*	DOUBLE	*nombre-tipo-diferenciado*
	DOUBLE	*nombre-tipo-diferenciado*	DOUBLE
DATE	*nombre-tipo-diferenciado*	DATE	*nombre-tipo-diferenciado*
	DATE	*nombre-tipo-diferenciado*	DATE
TIME	*nombre-tipo-diferenciado*	TIME	*nombre-tipo-diferenciado*
	TIME	*nombre-tipo-diferenciado*	TIME
TIMESTAMP	*nombre-tipo-diferenciado*	TIMESTAMP	*nombre-tipo-diferenciado*
	TIMESTAMP	*nombre-tipo-diferenciado*	TIMESTAMP
DATALINK	*nombre-tipo-diferenciado*	DATALINK	*nombre-tipo-diferenciado*
	DATALINK	*nombre-tipo-diferenciado*	DATALINK

Nota: NUMERIC y FLOAT no son aconsejables para crear un tipo de datos definido por el usuario para una aplicación portátil. Deben utilizarse DECIMAL y DOUBLE en su lugar.

Las funciones descritas en la tabla anterior son las únicas funciones que se generan automáticamente cuando se definen los tipos diferenciados. Como consecuencia, no se da soporte a ninguna de las funciones incorporadas (AVG, MAX, LENGTH, etcétera) en los tipos diferenciados hasta que se utiliza la sentencia CREATE FUNCTION para registrar las funciones definidas por el usuario para el tipo

diferenciado y esas funciones definidas por el usuario deben tener su origen en las funciones incorporadas adecuadas. En particular, observe que es posible registrar funciones definidas por el usuario que tienen su origen en funciones de columna incorporadas.

Cuando se crea un tipo diferenciado utilizando la cláusula WITH COMPARISONS, se crean operadores de comparación generados por el sistema. La creación de esos operadores de comparación generará entradas en la vista de catálogo SYSCAT.ROUTINES de las nuevas funciones.

El nombre de esquema del tipo diferenciado debe incluirse en la vía de acceso de SQL o en la opción FUNCPATH BIND para que la utilización de tales operadores y las funciones de conversión en las sentencias de SQL sean satisfactorias.

Ejemplos:

Ejemplo 1: Cree un tipo diferenciado llamado SHOESIZE que se base en un tipo de datos INTEGER.

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

Esto también dará como resultado la creación de operadores de comparación (=, <>, <, <=, >, >=) y que la función de conversión INTEGER(SHOESIZE) devuelva INTEGER y la función de conversión SHOESIZE(INTEGER) devuelva SHOESIZE.

Ejemplo 2: Cree un tipo diferenciado llamado MILES que se base en un tipo de datos DOUBLE.

```
CREATE DISTINCT TYPE MILES AS DOUBLE WITH COMPARISONS
```

Esto también dará como resultado la creación de operadores de comparación (=, <>, <, =, >, >=) y que la función de conversión DOUBLE(MILES) devuelva DOUBLE y la función de conversión MILES(DOUBLE) devuelva MILES.

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE FUNCTION" en la página 197
- "CREATE TABLE" en la página 341
- "SET PATH" en la página 744
- "Tipos definidos por el usuario" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtudt.sqc -- How to create, use, and drop user-defined distinct types (C)"
- "udfcli.sqc -- Call a variety of types of user-defined functions (C)"
- "dtudt.sqC -- How to create, use, and drop user-defined distinct types (C++)"
- "udfcli.sqC -- Call a variety of types of user-defined functions (C++)"
- "DtUdt.java -- How to create, use and drop user defined distinct types (JDBC)"
- "DtUdt.sqlj -- How to create, use and drop user defined distinct types (SQLj)"

CREATE EVENT MONITOR

La sentencia CREATE EVENT MONITOR define un supervisor que registrará ciertos sucesos que se produzcan cuando se utilice la base de datos. La definición de cada supervisor de sucesos especifica también el lugar donde la base de datos debe registrar los sucesos.

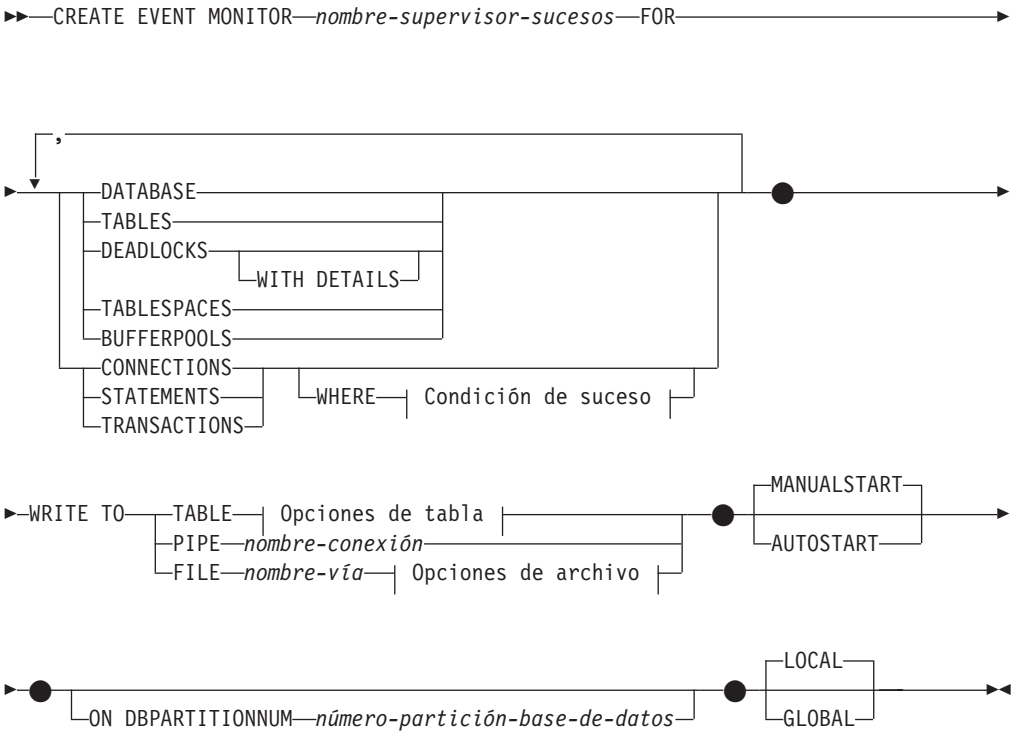
Invocación:

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

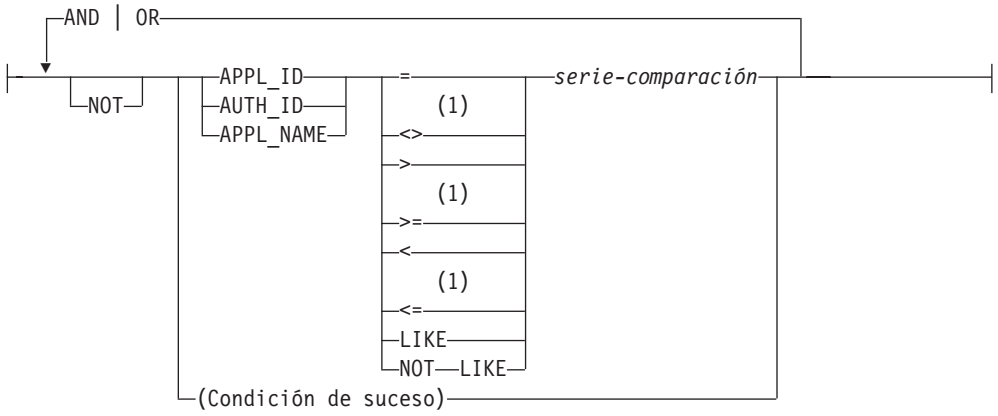
Autorización:

Los privilegios del ID de autorización deben incluir la autorización SYSADM o DBADM (SQLSTATE 42502).

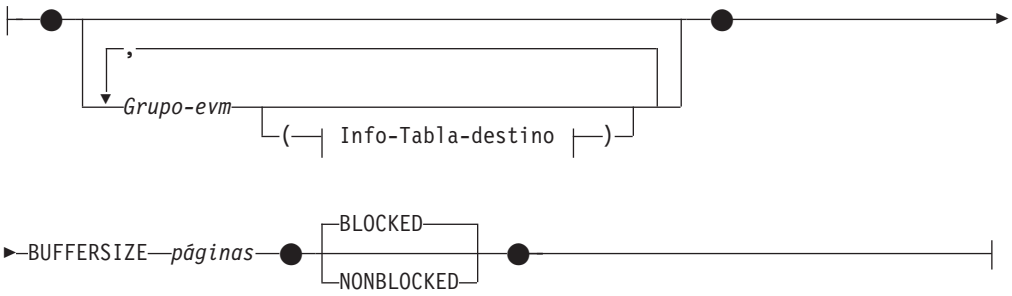
Sintaxis:



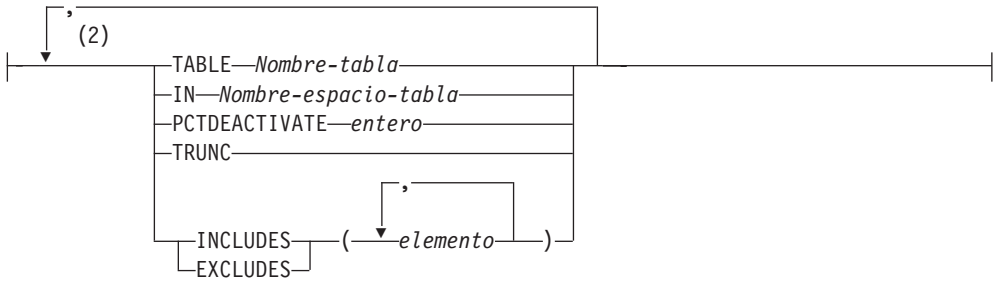
Condición de suceso:



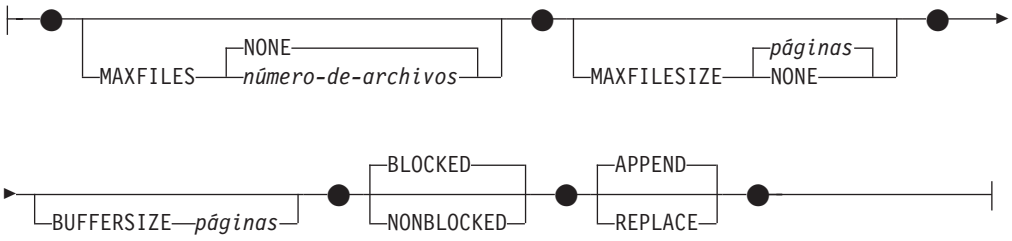
Opciones de tabla:



Info-Tabla-destino:



Opciones de archivo:



Notas:

1 También se da soporte a otras formas de estos operadores.

2 Cada cláusula sólo puede especificarse una vez.

Descripción:

nombre-supervisor-sucesos
Indica el nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o bien delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR
Introduce el tipo de suceso que se debe registrar.

DATABASE
Especifica que el supervisor de sucesos registre un suceso de base de datos cuando se desconecte la última aplicación de la base de datos.

TABLES
Especifica que el supervisor de sucesos registre un suceso de tabla para cada tabla activa cuando se desconecte la última aplicación de la base de datos. Una tabla activa es una tabla que ha cambiado desde la primera conexión con la base de datos.

DEADLOCKS
Especifica que el supervisor de sucesos registre un suceso de punto muerto siempre que se produzca un punto muerto. La especificación de la opción WITH DETAILS indica que el supervisor de sucesos generará un suceso de conexión de punto muerto más detallado para cada aplicación que esté implicada en un punto muerto. Estos detalles adicionales incluyen:

- Información acerca de la sentencia que la aplicación estaba ejecutando al producirse en punto muerto como, por ejemplo, el texto de la sentencia.
- Los bloqueos que mantenía la aplicación al producirse el punto muerto. En un entorno de bases de datos particionadas, los bloqueos que se incluyen son los correspondientes a la partición de base de datos donde la aplicación estaba esperando su bloqueo al producirse el punto muerto.

DEADLOCKS y DEADLOCKS WITH DETAILS no pueden especificarse a la vez en la misma sentencia (SQLSTATE 42613).

TABLESPACES
Especifica que el supervisor de sucesos registre un suceso de espacio de tablas para cada espacio de tablas cuando se desconecte la última aplicación de la base de datos.

BUFFERPOOLS
Especifica que el supervisor de sucesos registre un suceso de agrupación de almacenamientos intermedios cuando se desconecte la última aplicación de la base de datos.

CONNECTIONS
Especifica que el supervisor de sucesos registre un suceso de conexión cuando una aplicación se desconecta de la base de datos.

STATEMENTS
Especifica que el supervisor de sucesos registre un suceso de sentencia siempre que una sentencia de SQL termine su ejecución.

TRANSACTIONS
Especifica que el supervisor de sucesos registre un suceso de transacción siempre que se complete una transacción (es decir, siempre que haya una operación de confirmación o retrotracción).

WHERE *condición de suceso*
Define un filtro que determina qué conexiones hacen que se produzca un suceso de CONNECTION, STATEMENT o TRANSACTION. Si el resultado de la condición de suceso es TRUE para una conexión en particular, dicha conexión generará los sucesos pedidos.

Esta cláusula es una forma especial de la cláusula WHERE que no debe confundirse con una condición de búsqueda estándar.

Para determinar si una aplicación generará los sucesos para un supervisor de sucesos en particular, se evalúa la cláusula WHERE:

1. En cada conexión activa cuando se activa un supervisor de sucesos.
2. Posteriormente en cada conexión nueva con la base de datos en el tiempo de conexión.

La cláusula WHERE no se evalúa para cada suceso.

Si no se ha especificado ninguna cláusula WHERE, se supervisarán todos los sucesos del tipo de suceso especificado.

La cláusula de condición de suceso no debe exceder de 32.678 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

APPL_ID
Especifica que el ID de aplicación para cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

AUTH_ID
Especifica que el ID de autorización de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

APPL_NAME
Especifica que el nombre de programa de aplicación de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

El nombre de programa de aplicación son los 20 primeros bytes del nombre de archivo del programa de aplicación, después del último separador de la vía de acceso.

serie-comparación
Es una serie de caracteres que debe compararse con el APPL_ID, AUTH_ID o APPL_NAME de cada aplicación que se conecta con la base de datos. *serie-comparación* debe ser una constante de serie (es decir, las variables del lenguaje principal y otras expresiones de serie no están permitidas).

WRITE TO
Introduce el destino para los datos.

TABLE
Indica que el destino de los datos del supervisor de sucesos es un conjunto de tablas de base de datos. El supervisor de sucesos separa la corriente de datos en uno o más grupos de datos lógicos e inserta cada grupo en una tabla separada. Los datos para los grupos que tienen una tabla de destino

se conservan, mientras que los datos para los grupos que no tienen una tabla de destino se descartan. Cada elemento del supervisor que está contenido dentro de un grupo se correlaciona con una columna de tabla que tiene el mismo nombre. En la tabla sólo se insertan los elementos para los que existe una columna de tabla correspondiente. Los demás elementos se descartan.

Opciones de tabla
Especifican opciones de formato de tabla.

Info-Grupo-evm
Define la tabla de destino de un grupo de datos lógicos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula Info-Grupo-evm, se registrarán todos los grupos del tipo de supervisor de sucesos.

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

Tipo de supervisor de sucesos	Valor de Grupo-evm
Base de datos	DB CONTROL[1] DBMEMUSE
Tablas	TABLE CONTROL[1]
Puntos muertos	CONNHEADER DEADLOCK DLCONN CONTROL[1]
Puntos muertos con detalles	CONNHEADER DEADLOCK DLCONN[2] DLLOCK[3] CONTROL[1]
Espacios de tabla	TABLESPACE CONTROL[1]
Agrupaciones de almacenamientos intermedios	BUFFERPOOL CONTROL[1]
Conexiones	CONNHEADER CONN CONTROL[1] CONMEMUSE
Sentencias	CONNHEADER STMT SUBSECTION[4] CONTROL[1]

Tipo de supervisor de sucesos	Valor de Grupo-evm
Transacciones	CONNHEADER
	XACT
	CONTROL[1]

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

[2] Corresponde al suceso DETAILED_DLCONN.

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

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

Info-Tabla-destino

Identifica la tabla de destino de un grupo. Si no se especifica un valor para *Info-Tabla-destino*, el proceso de CREATE EVENT MONITOR continúa como se indica a continuación:

- Se utiliza un nombre de tabla que se ha obtenido (se describe a continuación).
- Se elige un espacio de tablas por omisión (se describe a continuación).
- Se incluyen todos los elementos.
- PCTDEACTIVATE y TRUNC no se especifican.

TABLE *Nombre-tabla*

Especifica el nombre de la tabla de destino. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el esquema del ID de autorización actual. Si no se proporciona ningún nombre, el nombre no calificado se obtiene de *Grupo-evm* y de *nombre-supervisor-sucesos*, como se indica a continuación:

```
subserie(Grupo-evm || "_" || nombre-supervisor-sucesos,
  1,128)
```

IN *Nombre-espacio-tabla*

Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona ningún nombre de espacio de tabla, el espacio de tablas se elige como se indica a continuación:

```
IF espacio de tablas IBMDEFAULTGROUP para el que el
 usuario tiene privilegio USE existe
 THEN elegirlo
ELSE IF un espacio de tablas para el que el usuario
 usuario tiene privilegio USE existe
 THEN elegirlo
ELSE emitir un error (SQLSTATE 42727)
```

PCTDEACTIVATE *entero*

Si está creándose una tabla en un espacio de tablas DMS, el parámetro PCTDEACTIVATE especifica hasta qué punto debe llenarse el espacio de tablas antes de que el supervisor de sucesos se desactive automáticamente. El valor especificado, que representa un porcentaje, puede ser de 0 a 100. El valor por omisión es 100 (significa que el supervisor de sucesos se desactivará cuando el espacio de tablas se haya llenado por completo). Esta opción no puede especificarse con espacios de tabla SMS.

TRUNC
Especifica que la columna STMT_TEXT se ha definido como VARCHAR(*n*), donde *n* es el tamaño máximo que puede caber en la fila de tabla. En este caso, el texto de la sentencia que supere el valor de *n* se truncará. En el ejemplo siguiente se muestra cómo se calcula el valor de *n*. Supongamos que:

- La tabla STMT se ha creado en un espacio de tablas que utiliza páginas de 32K.
- La longitud total de todas las demás columnas de la tabla es de 357 bytes.

En este caso, el tamaño de fila máximo para la tabla es de 32677 bytes. Por lo tanto, STMT_TEXT debería definirse como VARCHAR(32316); es decir, 32677 - 357 - 4. Si no se especifica TRUNC, la columna STMT_TEXT se definirá como CLOB(64K). Observe que STMT_TEXT se encuentra en el grupo STMT y en el grupo DLCONN (para los puntos muertos con supervisores de sucesos de detalles).

INCLUDES
Especifica que los elementos siguientes van a incluirse en la tabla.

EXCLUDES
Especifica que los elementos siguientes *no* van a incluirse en la tabla.

elemento
Identifica un elemento del supervisor. La información de los elementos puede proporcionarse de una de las formas siguientes:

- No especificar información de los elementos. En este caso, todos los elementos se incluyen en la sentencia CREATE TABLE.
- Especificar los elementos que deben incluirse en el formato: INCLUDES (elemento1, elemento2, ..., elemento*n*). Para estos elementos, sólo se crean columnas de tabla.
- Especificar los elementos que deben excluirse en el formato: EXCLUDES (elemento1, elemento2, ..., elemento*n*). Sólo se crean columnas de tabla para todos los elementos que no se han indicado aquí.

Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.

BUFFERSIZE *páginas*
Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Los supervisores de sucesos de tabla insertan todos los datos de un almacenamiento intermedio y emiten COMMIT cuando se ha procesado el almacenamiento intermedio. Cuanto más grandes sean los almacenamientos intermedios, mayor será el ámbito de confirmación utilizado por el supervisor de sucesos. Los supervisores de sucesos altamente activos deben tener almacenamientos intermedios mayores que los supervisores de sucesos relativamente inactivos. Cuando se

inicia un supervisor, se asignan dos almacenamientos intermedios del tamaño especificado. Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño mínimo (y el valor por omisión) de cada almacenamiento intermedio es de 4 páginas (es decir, 2 almacenamientos intermedios, donde cada uno de los cuales tiene 16K de tamaño). El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esa pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos *mon_heap_sz*.

BLOCKED
Especifica que cada agente que genera un suceso debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Debe seleccionarse BLOCKED para garantizar que no se van a perder datos. Es la opción por omisión.

NONBLOCKED
Especifica que cada agente que genera un suceso no debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Los supervisores de sucesos NONBLOCKED no hacen que las operaciones vayan más despacio como los supervisores de sucesos BLOCKED. Sin embargo, los supervisores de sucesos NONBLOCKED están sujetos a la pérdida de datos en sistemas muy activos.

PIPE
Especifica que el destino para los datos del supervisor de sucesos es una conexión con nombre. El supervisor de sucesos graba los datos en la conexión en una sola corriente (es decir, como si fuera un solo archivo infinitamente largo). Cuando se graban los datos en una conexión, el supervisor de sucesos no realiza grabaciones por bloques. Si no hay espacio en el almacenamiento intermedio de conexión, el supervisor de sucesos desechará los datos. Es responsabilidad de la aplicación supervisora el leer los datos pronto si desea asegurar que no se pierdan datos.

nombre-conexión
El nombre de la conexión (FIFO en AIX) en la que el supervisor de sucesos grabará los datos.

Las normas de denominación para las conexiones son específicas de las plataformas. En sistemas operativos UNIX, los nombres de conexiones se tratan como nombres de archivo. Como resultado, están permitidos los nombres de conexiones relativos y se tratan como nombres-vía relativas (consulte *nombre-vía* más abajo). Sin embargo, en Windows NT o Windows 2000, existe una sintaxis especial para un nombre de conexión. Como resultado de ello, en Windows NT o Windows 2000, se necesitan nombres de conexión absolutos.

La existencia de una conexión no se comprobará en el tiempo de creación del supervisor de sucesos. Es responsabilidad de la aplicación supervisora haber creado y abierto la conexión para lectura en el momento en que se activa el supervisor de sucesos. Si la conexión no está disponible en ese momento, el supervisor de sucesos se desactivará por sí solo y anotará cronológicamente un error. (Es decir, si se ha activado el supervisor de

sucesos en el momento del inicio de la base de datos como resultado de la opción AUTOSTART, el supervisor de sucesos anotará un error en el registro cronológico de errores del sistema.) Si se activa el supervisor de sucesos mediante la sentencia SET EVENT MONITOR STATE SQL, dicha sentencia fallará (SQLSTATE 58030).

FILE
Indica que el destino para los datos del supervisor de sucesos es un archivo (o conjunto de archivos). El supervisor de sucesos graba la corriente de datos como una serie de archivos numerados de 8 caracteres, con la extensión "evt". (por ejemplo, 00000000.evt, 00000001.evt y 00000002.evt). Los datos deben considerarse un archivo lógico incluso cuando se dividen los datos en fragmentos más pequeños (es decir, el inicio de la corriente de datos es el primer byte del archivo 00000000.evt; el final de la corriente de datos es el último byte del archivo nnnnnnnn.evt).

El tamaño máximo de cada archivo se puede definir también como el número máximo de archivos. Un supervisor de sucesos no dividirá nunca un solo registro de sucesos entre dos archivos. Sin embargo, el supervisor de sucesos puede grabar registros relacionados en dos archivos distintos. Es responsabilidad de la aplicación que utiliza estos datos realizar un seguimiento de dicha información relacionada cuando procese los archivos de sucesos.

nombre-vía
El nombre del directorio en el que el supervisor de sucesos debe grabar los datos de los archivos de sucesos. El servidor debe conocer la vía de acceso; sin embargo, la propia vía de acceso puede residir en otra partición (por ejemplo, en un sistema basado en UNIX, podría ser un archivo montado NFS). Se debe utilizar una constante de serie cuando se especifica el *nombre-vía*.

El directorio no tiene que existir en el momento de CREATE EVENT MONITOR. Sin embargo, se realiza una comprobación de la existencia de la vía de acceso de destino cuando se activa el supervisor de sucesos. En este momento, si la vía de acceso de destino no existe, se genera un error (SQLSTATE 428A3).

Si se especifica una vía de acceso absoluta (una vía de acceso que empieza por el directorio raíz en AIX o por un identificador de disco en Windows NT o Windows NT), se utilizará la vía de acceso especificada. Si se especifica una vía de acceso relativa (una vía que no empieza en la raíz), se utilizará la vía de acceso al directorio DB2EVENT en el directorio de la base de datos.

Cuando se especifique una vía de acceso relativa, se utiliza el directorio DB2EVENT para convertirla en una vía de acceso absoluta. En adelante, no se distinguirá entre las vías de acceso absoluta y relativa. La vía de acceso absoluta se almacena en la vista de catálogo SYSCAT.EVENTMONITORS.

Es posible especificar dos o más supervisores de sucesos que tengan la misma vía de acceso de destino. Sin embargo, cuando uno de los supervisores de sucesos se ha activado por primera vez, y siempre que el directorio de destino no esté vacío, será imposible activar cualquiera de los supervisores de sucesos.

Opciones de archivo
Especifica las opciones para el formato del archivo.

MAXFILES NONE
Especifica que no hay ningún límite en el número de archivos de sucesos que vaya a crear el supervisor de sucesos. Este es el valor por omisión.

MAXFILES *número-de-archivos*
Especifica que hay un límite en el número de archivos del supervisor de sucesos que vayan a existir para un supervisor de sucesos en particular en cualquier momento. Siempre que un supervisor de sucesos tenga que crear otro archivo, comprobará que el número de archivos .evt del directorio sea inferior al *número-de-archivos*. Si ya se ha alcanzado este límite, el supervisor de sucesos se desactivará por sí solo.

Si una aplicación elimina los archivos de sucesos del directorio después de haberlos grabado, el número total de archivos que un supervisor de sucesos puede producir puede exceder el *número-de-archivos*. Esta opción se ha proporcionado para permitir a un usuario garantizar que los datos de sucesos no consumirán más de una cantidad de espacio de disco especificada.

MAXFILESIZE *páginas*
Especifica que hay un límite en el tamaño de cada archivo del supervisor de sucesos. Siempre que un supervisor de sucesos grabe un nuevo registro de suceso en un archivo, comprueba que el archivo no vaya a crecer de manera que sobrepase las *páginas* (en unidades de páginas de 4K). Si el archivo resultante fuese a ser demasiado grande, entonces el supervisor de sucesos conmutará al siguiente archivo. El valor por omisión para esta opción es:

- Windows NT o Windows 2000 - 200 páginas de 4 K
- UNIX - 1000 páginas de 4K

El número de páginas debe ser mayor que el tamaño del almacenamiento intermedio de sucesos en páginas, como mínimo. Si no se cumple este requisito, se generará un error (SQLSTATE 428A4).

MAXFILESIZE NONE
Especifica que no hay ningún límite establecido en el tamaño de un archivo. Si se especifica MAXFILESIZE NONE, también debe especificarse MAXFILES 1. Esta opción significa que un archivo contendrá todos los datos de sucesos para un supervisor de sucesos en particular. En este caso el único archivo de sucesos será 00000000.evt.

BUFFERSIZE *páginas*
Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Todas las E/S del archivo del supervisor de sucesos se almacenan temporalmente para mejorar el rendimiento de los supervisores de sucesos. Cuanto más grandes sean los almacenamientos intermedios, menos E/S realizará el supervisor de sucesos. Los supervisores de sucesos altamente activos deben tener almacenamientos intermedios mayores que los supervisores de sucesos relativamente inactivos. Cuando se inicia el supervisor, se asignan dos almacenamientos intermedios del tamaño especificado. Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño mínimo y por omisión de cada almacenamiento intermedio (si no se especifica esta opción) es 4 páginas (es decir, 2 almacenamientos intermedios, cada uno con un tamaño de 16 K). El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor (MON_HEAP), pues los almacenamientos intermedios se asignan a partir de la pila. Si utiliza muchos supervisores de sucesos al mismo tiempo, aumente el tamaño del parámetro de configuración de base de datos MON_HEAP.

Los supervisores de sucesos que escriben sus datos en una conexión ("pipe") también tienen dos almacenamientos intermedios internos (no configurables) que tienen un tamaño de 1 página cada uno. Estos almacenamientos intermedios también se asignan a partir de la pila del supervisor (MON_HEAP). Para cada supervisor de sucesos activo que tiene un destino de conexión, aumente el tamaño de la pila de la base de datos en 2 páginas.

BLOCKED
Especifica que cada agente que genera un suceso debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Debe seleccionarse BLOCKED para garantizar que no se van a perder datos. Es la opción por omisión.

NONBLOCKED
Especifica que cada agente que genera un suceso no debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Los supervisores de sucesos NONBLOCKED no hacen que las operaciones vayan más despacio como los supervisores de sucesos BLOCKED. Sin embargo, los supervisores de sucesos NONBLOCKED están sujetos a la pérdida de datos en sistemas muy activos.

APPEND
Especifica que si ya existen archivos de datos de sucesos cuando se activa el supervisor de sucesos, éste añadirá los nuevos datos de sucesos a los archivos de corriente de datos existentes. Cuando el supervisor de sucesos se reactiva, reanudará la escritura en los archivos de sucesos como si nunca se hubiese desactivado. APPEND es la opción por omisión.

La opción APPEND no se aplica al momento de CREATE EVENT MONITOR, si hay datos de sucesos existentes en el directorio donde el supervisor de sucesos que se acaba de crear va a grabar sus datos de sucesos.

REPLACE
Especifica que si ya existen archivos de datos de sucesos cuando se activa el supervisor de sucesos, éste borrará todos los archivos de sucesos y empezará a grabar los datos en el archivo 00000000.evt.

MANUALSTART
Especifica que el supervisor de sucesos no se iniciará automáticamente cada vez que se inicie la base de datos. Los supervisores de sucesos con la opción MANUALSTART deben activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Es la opción por omisión.

AUTOSTART
Especifica que el supervisor de sucesos se iniciará automáticamente cada vez que se inicie la base de datos.

ON DBPARTITIONNUM *número-partición-base-de-datos*
Especifica la partición de base de datos en la que se debe ejecutar el supervisor de sucesos. Esta cláusula es válida para los supervisores de sucesos de archivos y de conexiones, pero no para los supervisores de sucesos de grabación en tabla. En un entorno de bases de datos particionadas, los supervisores de sucesos de grabación en tabla se ejecutarán y grabarán los sucesos en todas las particiones en las que se hayan definido espacios de tablas para tablas de destino.

Cuando el ámbito de supervisión se define como GLOBAL, todas las particiones de base de datos informan al número de partición de base de datos especificado. El componente de E/S se ejecutará físicamente en la partición de base de datos especificada y escribirá los registros en el archivo o la conexión que se ha especificado.

GLOBAL
El supervisor de sucesos informa acerca de todas las particiones de base de datos. Para una base de datos particionada en DB2 Universal Database Versión 8, sólo los puntos muertos y los puntos muertos con supervisores de sucesos de detalles pueden definirse como GLOBAL.

LOCAL
El supervisor de sucesos sólo informa acerca de la partición de base de datos que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Este es el valor por omisión.

Normas:

- Cada uno de los tipos de sucesos (DATABASE, TABLES, DEADLOCK,...) sólo pueden especificarse una vez en la definición de un supervisor de sucesos en particular.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
- Las definiciones del supervisor de sucesos se registran en la vista de catálogo SYSCAT.EVENTMONITORS. Los sucesos en sí se registran en la vista de catálogo SYSCAT.EVENTS. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Cuando se utiliza DEADLOCKS WITH DETAILS en lugar de DEADLOCKS, ello afecta al rendimiento. Cuando se produce un punto muerto, el gestor de bases de datos necesita más tiempo para grabar la información adicional del punto muerto.
- Normalmente, siempre que se establece una conexión se graba un suceso CONNHEADER. Sin embargo, si se ha creado un supervisor de sucesos sólo para DEADLOCKS WITH, sólo se grabará un suceso CONNHEADER la primera vez que la conexión participe en un punto muerto.
- El parámetro BUFFERSIZE limita el tamaño de los sucesos STMT y DETAILED_DLCONN. Si un suceso STMT no puede caber dentro de un almacenamiento intermedio, éste se trunca, afectando el truncamiento al texto de la sentencia. Si un suceso DETAILED_DLCONN no puede caber dentro de un

almacenamiento intermedio, éste se trunca eliminando bloqueos. Si todavía no tiene cabida, el texto de la sentencia se trunca.

- ***Supervisores de sucesos de grabación en tabla:***
  - Notas generales:
    - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
    - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
    - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.
    - Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
    - Deberán existir espacios de tabla para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tabla.
    - Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada partición de base de datos de la instancia, y cada uno de estos procesos sólo informa datos a la partición de base de datos en la que está ejecutándose.
    - Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
      - LOG_STREAM_HEADER
      - LOG_HEADER
      - DB_HEADER (Los elementos db_name y db_path no se graban. El elemento conn_time se graba en CONTROL.)
    - En un entorno de bases de datos particionadas, los datos sólo se graban en las tablas de destino de las particiones en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones con el fin de supervisarlas, creando un espacio de tablas que sólo exista en determinadas particiones.

      En un entorno de bases de datos particionadas, si algunas tablas de destino no residen en una partición, pero otras tablas de destino sí que residen en esa misma partición, sólo se registrarán los datos para las tablas de destino que residan en esa partición.
    - Los usuarios pueden podar manualmente todas las tablas de destino.
  - Columnas de tabla:
    - Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. Las variables del supervisor de tipo sqlm_time (tiempo transcurrido) son una excepción. Los nombres de

columna de tales tipos son TYPE_NAME_S y TYPE_NAME_MS, que representan las columnas que almacenan el tiempo en segundos y en milisegundos respectivamente. Cualquier elemento del supervisor de sucesos que *no* tenga una columna de tabla de destino correspondiente se pasa por alto.

- Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.
- Los tipos de columnas que se utilizan para los elementos del supervisor corresponden a la correlación siguiente:

```
SQLM_TYPE_STRING                    CHAR[n], VARCHAR[n] o CLOB(n)
                                    (Si los datos del registro del
                                    supervisor de sucesos exceden de n
                                    bytes, se truncan.)
SQLM_TYPE_U8BIT y SQLM_TYPE_8BIT   SMALLINT, INTEGER o BIGINT
SQLM_TYPE_16BIT y SQLM_TYPE_U16BIT SMALLINT, INTEGER o BIGINT
SQLM_TYPE_32BIT y SQLM_TYPE_U32BIT INTEGER o BIGINT
SQLM_TYPE_U64BIT y SQLM_TYPE_64BIT BIGINT
sqlm_timestamp                      TIMESTAMP
sqlm_time(tiempo transcurrido)      BIGINT
sqlca:
  sqlerrmc                          VARCHAR[72]
  sqlstate                          CHAR[5]
  sqlwarn                           CHAR[11]
  otros campos                      INTEGER o BIGINT
```

- Las columnas se han definido para que sean NOT NULL.
- Puesto que el rendimiento de las tablas con columnas CLOB es inferior al de las tablas que tienen columnas VARCHAR, considere la utilización de la palabra clave TRUNC al especificar el Grupo-evm de STMT (o Grupo-evm de DLCONN, si se utiliza el tipo de evento DEADLOCKS WITH DETAILS).
- A diferencia de otras tablas de destino, las columnas de la tabla CONTROL no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

```
Nombre columna   Tipo datos    Nulos     Descripción
-----------      ---------     --------  -----------
PARTITION_KEY    INTEGER       N         Clave de partición (sólo base
                                         de datos particionada)
PARTITION_NUMBER INTEGER       N         Número de partición (sólo base
                                         de datos particionada)
EVMONNAME        VARCHAR(128)  N         Nombre de supervisor de sucesos
MESSAGE          VARCHAR(128)  N         Describe la naturaleza de
                                         la columna MESSAGE_TIME.
                                         Puede ser uno de
                                         los siguientes:
                                         - FIRST_CONNECT (la hora de
                                           la primera conexión con la
                                           base de datos tras la
                                           activación)
                                         - EVMON_START (la hora en que
                                           se inició el supervisor de
                                           sucesos listado en EVMONNAME)
                                         - OVERFLOWS:n (indica que se
                                           han descartado n registros
                                           por desbordamiento del
                                           almacenamiento intermedio)
MESSAGE_TIME   TIMESTAMP       N         Indicación de la hora
```

- En un entorno de bases de datos particionadas, la primera columna de cada tabla se denomina PARTITION_KEY, es NOT NULL y es de tipo INTEGER. Esta columna se utiliza como clave de particionamiento para la tabla. El valor de esta columna se elige de forma que cada proceso del supervisor de

sucesos inserte datos en la partición de base de datos en la que está ejecutándose el proceso; es decir, se realizan operaciones de inserción localmente en la partición de base de datos en la que está ejecutándose el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se elimina una partición de base de datos y se realiza la redistribución de los datos, todos los datos de la partición de base de datos que se ha eliminado se dirigirán a otra partición de base de datos en lugar de distribuirse equitativamente. Por lo tanto, antes de eliminar una partición de base de datos, considere la supresión de todas las filas de tabla de esa partición de base de datos.

- En un entorno de bases de datos particionadas, puede definirse una columna denominada PARTITION_NUMBER para cada tabla. Esta columna es NOT NULL y es de tipo INTEGER. Contiene el número de la partición en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de bases de datos no particionadas.

– Atributos de tabla:
  - Se utilizan atributos de tabla por omisión. Además de la clave de particionamiento (sólo en las bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
  - En la tabla puede crearse índices.
  - Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
  - Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.
– Activación del supervisor de sucesos:
  - Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.
  - En un entorno de bases de datos particionadas, el proceso de activación se produce en cada partición de la instancia. En una partición en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en una partición si existe, como mínimo, una tabla de destino en esa partición. Además, si no se encuentra alguna tabla de destino en una partición, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.
  - Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de bases de datos particionadas, si el espacio de tablas no reside en una partición de base de datos), la activación continúa y los datos que, de otro modo, se insertarían en esta tabla, se pasan por alto.
  - El proceso de activación valida cada tabla de destino. Si la validación no se realiza satisfactoriamente, la activación del supervisor de sucesos no tiene lugar y se graban mensajes en el archivo de anotaciones cronológicas de administración.
  - Durante la activación en un entorno de bases de datos particionadas, las filas de la tabla CONTROL de FIRST_CONNECT y de EVMON_START sólo se insertan en la partición de base de datos de catálogo. Para ello es necesario que el espacio de tablas de la tabla CONTROL exista en la partición de base de datos de catálogo. Si no existe en la partición de base de datos de catálogo, dichas inserciones no se realizan.

- En un entorno de bases de datos particionadas, si una partición todavía no está activa al activarse el supervisor de sucesos de grabación en tabla, esa partición se activará antes de activarse el supervisor de sucesos. En este caso, la activación de la base de datos se comporta como si una sentencia CONNECT de SQL hubiera activado la base de datos en todas las particiones.

– Tiempo de ejecución:
  - Un supervisor de sucesos se ejecuta con la autorización DBADM.
  - Si, mientras existe un supervisor de sucesos activo, no se ejecuta satisfactoriamente una operación de inserción en una tabla de destino:
    • Los cambios no confirmados se retrotraen.
    • Se graba un mensaje en el archivo de anotaciones cronológicas de administración.
    • El supervisor de sucesos se desactiva.
  - Si existe un supervisor de sucesos activo, ejecuta COMMIT localmente cuando ha terminado de procesar un almacenamiento intermedio del supervisor de sucesos.
  - En un entorno de bases de datos particionadas, el texto de la sentencia real, que puede tener una longitud de hasta 65 535 bytes, sólo lo almacena (en la tabla STMT o DLCONN) el proceso de supervisor de sucesos que se ejecuta en la partición de bases de datos coordinadora de la aplicación. En otras particiones de base de datos, este valor tiene una longitud cero.
  - En un entorno de bases de datos no particionadas, todos los supervisores de sucesos de grabación en tabla se desactivan cuando finaliza la última aplicación (y si la base de datos no se ha activado explícitamente). En un entorno de bases de datos particionadas, los supervisores de sucesos de grabación en tabla se desactivan al desactivarse la partición de catálogo.
  - La sentencia DROP EVENT MONITOR no elimina tablas de destino.

Ejemplos:

Ejemplo 1: El siguiente ejemplo crea un supervisor de sucesos denominado SMITHPAY. Este supervisor de sucesos, reunirá los datos de sucesos para la base de datos así como para las sentencias de SQL realizadas por la aplicación PAYROLL propiedad del ID de autorización JSMITH. Los datos se añadirán a la vía de acceso absoluta /home/jsmith/event/smithpay/. Se crearán un máximo de 25 archivos. Cada archivo tendrá una longitud máxima de 1 024 páginas de 4K. La E/S del archivo no estará bloqueada.

```
CREATE EVENT MONITOR SMITHPAY
   FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
   WRITE TO FILE '/home/jsmith/event/smithpay'
   MAXFILES 25
   MAXFILESIZE 1024
   NONBLOCKED
   APPEND
```

Ejemplo 2: El ejemplo siguiente crea un supervisor de sucesos denominado DEADLOCKS_EVTS. Este supervisor de sucesos reunirá los sucesos de puntos muertos y los grabará en la vía de acceso relativa DLOCKS. Se grabará un archivo y no hay tamaño de archivo máximo. Cada vez que se active el supervisor de sucesos, se añadirán los datos de sucesos al archivo 00000000.evt si existe. El supervisor de sucesos se iniciará cada vez que se inicie la base de datos. La E/S estará bloqueada por omisión.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
   FOR DEADLOCKS
   WRITE TO FILE 'DLOCKS'
   MAXFILES 1
   MAXFILESIZE NONE
   AUTOSTART
```

Ejemplo 3: Este ejemplo crea un supervisor de sucesos denominado DB_APPLS. Este supervisor de sucesos reúne sucesos de conexión y graba datos en la conexión con nombre /home/jsmith/applpipe.

```
CREATE EVENT MONITOR DB_APPLS
   FOR CONNECTIONS
   WRITE TO PIPE '/home/jsmith/applpipe'
```

Ejemplo 4: Este ejemplo, en el que se da por supuesto un entorno de bases de datos particionadas, crea un supervisor de sucesos denominado FOO. Este supervisor de sucesos recopila los sucesos de la sentencia de SQL y los graba en tablas de SQL con los siguientes nombres que ha obtenido:

- CONNHEADER_FOO
- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

Puesto que no se ha proporcionado información de espacio de tabla, todas las tablas se crearán en un espacio de tablas seleccionado por el sistema, basándose en las normas que se describen en la cláusula IN *Nombre-espacio-tabla*. Todas las tablas incluyen todos los elementos de su grupo (es decir, se definen columnas cuyos nombres son equivalentes a los nombres de los elementos.)

```
CREATE EVENT MONITOR FOO
  FOR STATEMENTS
  WRITE TO TABLE
```

Ejemplo 5: En este ejemplo, en el que se da por supuesto un entorno de bases de datos particionadas, se crea un supervisor de sucesos denominado BAR. Este supervisor de sucesos recopila la sentencia de SQL y los sucesos de la transacción y los graba en tablas, como se indica a continuación:

- Los datos del grupo STMT se graban en la tabla MYDEPT.MYSTMTINFO. La tabla se crea en el espacio de tablas MYTABLESPACE. Cree columnas sólo para los elementos siguientes: ROWS_READ, ROWS_WRITTEN y STMT_TEXT. Los demás elementos del grupo se descartarán.
- Los datos del grupo SUBSECTION se graban en la tabla MYDEPT.MYSUBSECTIONINFO. La tabla se crea en el espacio de tablas MYTABLESPACE. La tabla incluye todas las columnas, excepto START_TIME, STOP_TIME y PARTIAL_RECORD.
- Los datos del grupo XACT se graban en la tabla XACT_BAR. Puesto que no se ha proporcionado información de espacio de tabla, la tabla se creará en un espacio de tablas seleccionado por el sistema, basándose en las normas que se describen en la cláusula IN *Nombre-espacio-tabla*. Esta tabla incluye todos los elementos que están contenidos en el grupo XACT.
- No se crea ninguna tabla para connheader o control; todos los datos de esos grupos se descartan.

```
CREATE EVENT MONITOR BAR
  FOR STATEMENTS, TRANSACTIONS
  WRITE TO TABLE
  STMT(TABLE MYDEPT.MYSTMTINFO, IN MYTABLESPACE,
```

```
    INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
  SUBSECTION(TABLE MYDEPT.MYSUBSECTIONINFO, IN MYTABLESPACE,
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "Event monitor logical data groups and monitor elements" en la publicación *System Monitor Guide and Reference*

CREATE FUNCTION

Esta sentencia se utiliza para registrar o definir una función definida por el usuario o una plantilla de función en un servidor de aplicaciones.

Existen cinco tipos diferentes de funciones que se pueden crear utilizando esta sentencia. Cada una de ellas se describe por separado.

- Escalar externo. La función se escribe en un lenguaje de programación y devuelve un valor escalar. El ejecutable externo se registra en la base de datos, junto con diversos atributos de la función.
- Tabla externa. La función se escribe en un lenguaje de programación y devuelve una tabla completa. El ejecutable externo se registra en la base de datos junto con diversos atributos de la función.
- Tabla externa OLE DB. Una función de tabla externa OLE DB definida por el usuario está registrada en la base de datos para acceder a los datos de un proveedor OLE DB.
- Con origen o plantilla. Una función fuente se implanta invocando otra función (incorporada, externa, SQL o fuente) que ya está registrada en la base de datos.

  Es posible crear una función parcial, denominada *función de plantilla*, que define qué tipos de valores van a devolverse, pero que no contiene código ejecutable. El usuario la correlaciona con una función fuente de datos dentro de un sistema federado, de esta forma se puede invocar la función fuente de datos desde una base de datos federada. Una plantilla de función sólo se puede registrar en un servidor de aplicaciones que esté designado como servidor federado.
- Escalar, tabla o fila de SQL. El cuerpo de la función está escrito en SQL y se define junto con el registro en la base de datos. Devuelve un valor escalar, una tabla o una fila individual.

Información relacionada:

- "CREATE FUNCTION (Tabla externa OLE DB)" en la página 241
- "CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 259
- "CREATE FUNCTION (Escalar externa)" en la página 198
- "CREATE FUNCTION (Tabla externa)" en la página 223
- "CREATE FUNCTION (Con origen o plantilla)" en la página 249

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"
- "udfcli.sqc -- Call a variety of types of user-defined functions (C)"
- "udfemcli.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "udfcli.c -- How to work with different types of user-defined functions (UDFs)"
- "udfcli.sqC -- Call a variety of types of user-defined functions (C++)"
- "udfemcli.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"
- "UDFCreate.db2 -- How to catalog the Java UDFs contained in UDFsrv.java "
- "UDFjCreate.db2 -- How to catalog the Java UDFs contained in UDFjsrv.java "

CREATE FUNCTION (Escalar externa)

Esta sentencia se utiliza para registrar una función escalar externa definida por el usuario en un servidor de aplicaciones. Una *función escalar* devuelve un solo valor cada vez que se invoca y en general es válida donde una expresión SQL sea válida

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema de la función no hace referencia a un esquema existente.
  - Privilegio CREATEIN para el esquema, si el nombre de esquema de la función no hace referencia a un esquema existente.

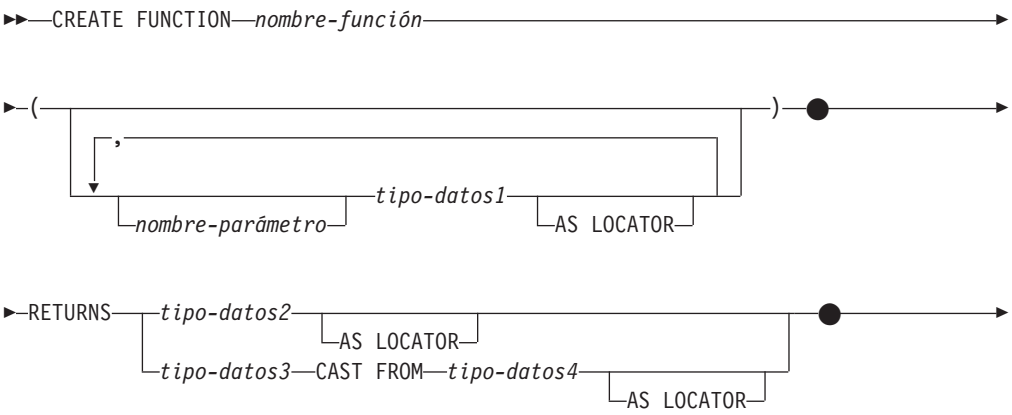
Para crear una función no restringida, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

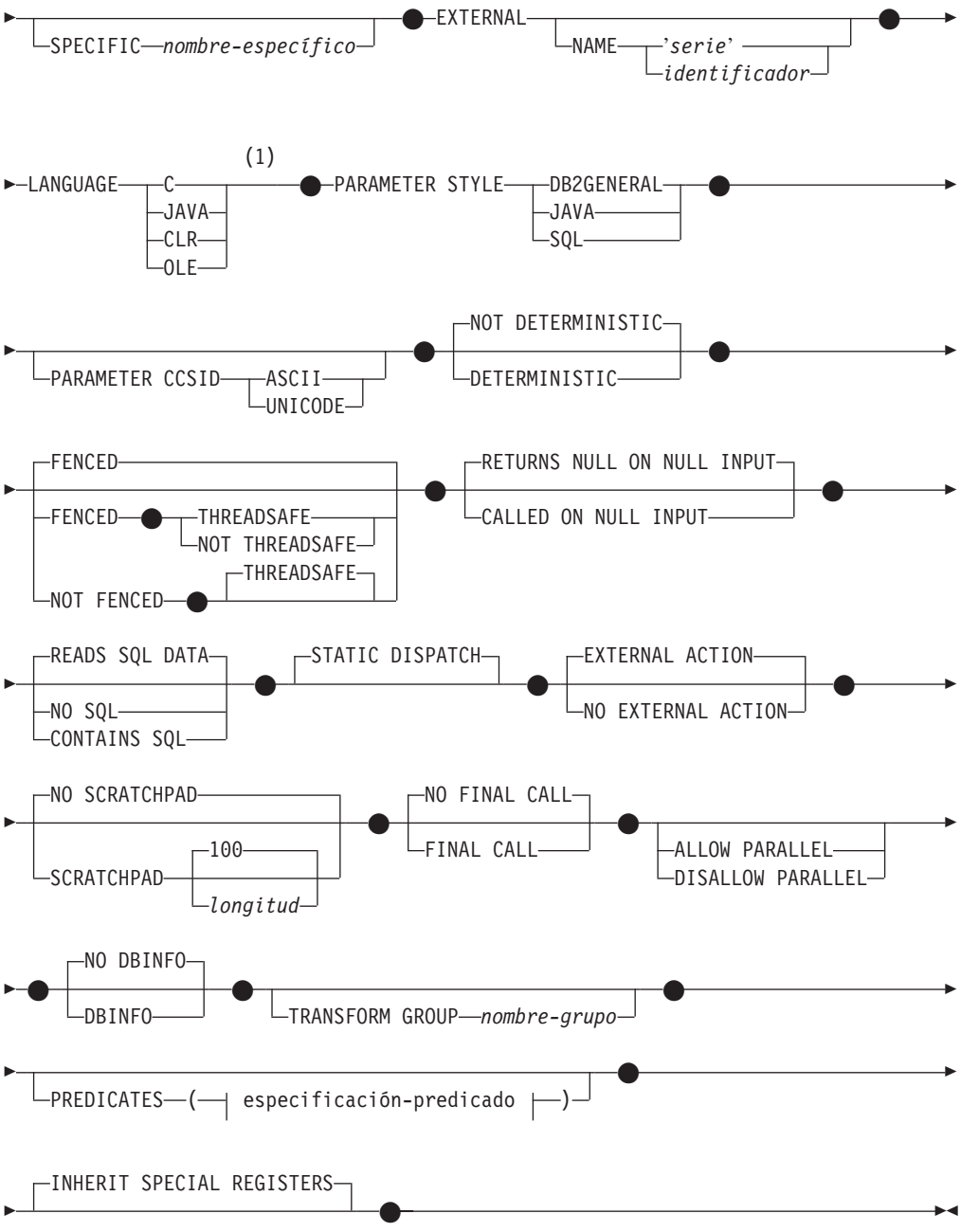
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM.

Para crear una función restringida, no se necesitan autorizaciones ni privilegios adicionales.

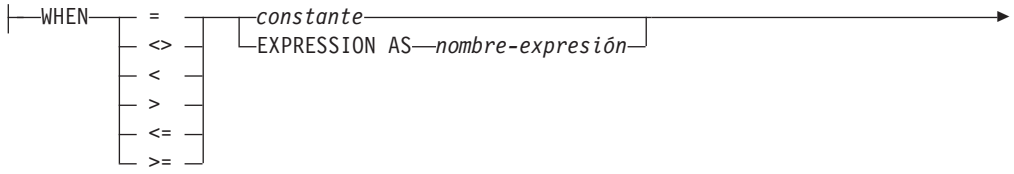
Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

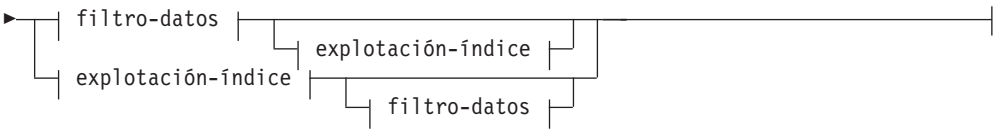
Sintaxis:



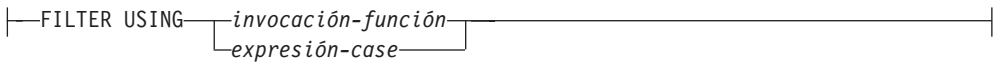


especificación-predicado:

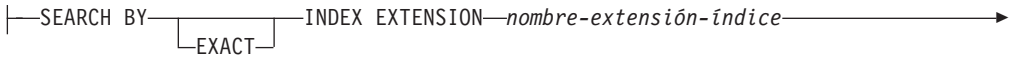




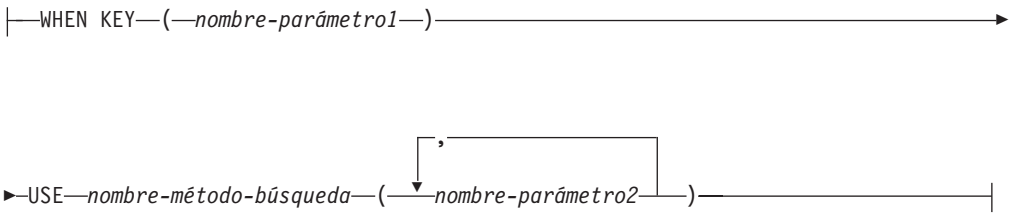
filtro-datos:



explotación-índice:



norma-explotación:



Notas:

1 También se da soporte a LANGUAGE SQL.

Descripción:

nombre-función
Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre calificado no debe ser el mismo que el tipo de datos del primer parámetro, si ese parámetro es un tipo estructurado.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función o método descritos en el catálogo SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS'. De lo contrario, se genera un error (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función*. Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación. Cualquier incumplimiento de esta norma provocará la aparición de un error (SQLSTATE 42939).

En general, puede utilizarse el mismo nombre para más de una función si existe alguna diferencia en la signatura de las funciones.

Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada, a menos que sea una alteración temporal intencionada. Si se otorga a una función que tiene un significado diferente el mismo nombre (por ejemplo, LENGTH, VALUE, MAX), con argumentos coherentes (caso de una función escalar incorporada o una función de columna), se corre el riesgo de provocar errores en las sentencias del SQL dinámico o al volver a enlazar las aplicaciones del SQL estático; es posible que la aplicación falle, o incluso peor, que parezca que se ejecuta satisfactoriamente y genere un resultado diferente.

nombre-parámetro
Designa el parámetro que se puede utilizar en la definición de función subsiguiente. Los nombres de parámetro son necesarios para hacer referencia a los parámetros de una función en la cláusula de *explotación-índice* de una especificación de predicado.

(tipo-datos1,...)
Identifica el número de parámetros de entrada de la función, al tiempo que especifica el tipo de datos de cada parámetro. En la lista debe especificarse una entrada por cada parámetro que la función espera recibir. No se permiten más de 90 parámetros. En caso de sobrepasarse este límite, se produce un error (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, que son DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...
CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

Las sentencias segunda y cuarta no se ejecutarían satisfactoriamente porque se considera que son funciones duplicadas.

tipo-datos1
Especifica el tipo de datos del parámetro.

- Pueden proporcionarse especificaciones y abreviaturas de tipo de datos SQL que puedan especificarse en la definición de *tipo-datos1* de la sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje que se utiliza para escribir la función.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- REF(*nombre-tipo*) puede especificarse como tipo de datos de un parámetro. Sin embargo, dicho parámetro no debe tener ámbito.
- Se pueden especificar tipos estructurados, con la condición de que existan las funciones de transformación apropiadas en el grupo de transformación asociado.

AS LOCATOR
Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB a la UDF en lugar del valor real. Esto reduce considerablemente el número de bytes que se pasan a la UDF y puede también mejorar el rendimiento, especialmente cuando la UDF sólo necesite unos pocos bytes.

El siguiente ejemplo ilustra la utilización de la cláusula AS LOCATOR en las definiciones de parámetros:

```
CREATE FUNCTION foo (CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
  ...
```

lo que supone que IMAGE es un tipo diferenciado basado en uno de los tipos LOB.

Observe también que para fines de promoción de argumentos, la cláusula AS LOCATOR no tiene ningún efecto. En el ejemplo, los tipos se consideran CLOB e IMAGE respectivamente, lo que significa que podría pasarse un argumento CHAR o VARCHAR a la función como el primer argumento. Igualmente, AS LOCATOR no tiene ningún efecto en la signatura de la función, que se utiliza en la comparación de la función (a) cuando se hace referencia en DML, por un proceso llamado "resolución de función" y (b) cuando se hace referencia en una sentencia DDL como, por ejemplo, COMMENT ON o DROP. De hecho, la cláusula puede utilizarse o no en una sentencia COMMENT ON o DROP sin que tenga ningún significado.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS
Esta cláusula obligatoria identifica el resultado de la función.

tipo-datos2
Especifica el tipo de datos de la salida.

En este caso, se aplican exactamente las mismas consideraciones que para los parámetros de funciones externas descritas arriba bajo *tipo-datos1* para parámetros de función.

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB de la UDF en lugar del valor real.

tipo-datos3 **CAST FROM** *tipo-datos4*
Especifica el tipo de datos de la salida.

Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función. Por ejemplo, en

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))
  RETURNS DATE CAST FROM CHAR(10)
  ...
```

el código de función devuelve un valor CHAR(10) al gestor de bases de datos que, a su vez, lo convierte en DATE y pasa dicho valor a la sentencia que realiza la invocación. El *tipo-datos4* debe ser convertible al parámetro *tipo-datos3*. Si no es convertible, se genera un error (SQLSTATE 42880).

Debido a que la longitud, precisión o escala de *tipo-datos3* puede inferirse de *tipo-datos4*, no es necesario (pero está permitido) especificar la longitud, precisión o escala de los tipos con parámetros especificados para *tipo-datos3*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo, puede utilizarse VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

No pueden especificarse tipos diferenciados ni tipos estructurados como tipo para *tipo-datos4* (SQLSTATE 42815).

La operación de conversión del tipo de datos también está sujeta a comprobaciones durante la ejecución que pueden dar lugar a errores de conversión.

AS LOCATOR
Para especificaciones de *tipo-datos4* que sean tipos LOB o tipos diferenciados que estén basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe devolver un localizador de LOB de una UDF en lugar del valor actual.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar otra instancia de función ni especificación de método que exista en el servidor de aplicaciones; de lo contrario, se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL
Esta cláusula indica que la sentencia CREATE FUNCTION se emplea para registrar una nueva función basada en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

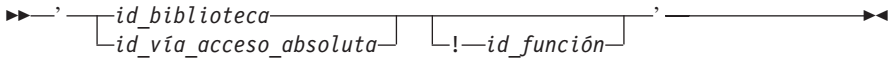
Si no se especifica la cláusula NAME, se da por supuesto "NAME *nombre-función*".

NAME *'serie'*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo.

La opción 'serie' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando (CREATE). Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función dentro de la biblioteca). Sin embargo, cuando la función se utiliza en una sentencia de SQL, la biblioteca y la función que está dentro de la biblioteca deben existir y debe poder accederse a éstas desde la máquina servidor de base de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).



  Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

  id_biblioteca
  Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:
  - En los sistemas basados en UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.
  - En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

  id_vía_acceso_absoluta
  Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, si se especifica '/u/jchui/mylib/myfunc' en sistemas basados en UNIX, el gestor de bases de datos buscará la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id_función
Identifica el nombre del punto de entrada de la función que debe invocarse. El signo ! sirve como delimitador entre el ID de biblioteca y el ID de función. Si se omite *! id_función*, el gestor de bases de datos utilizará el punto de entrada por omisión que se ha establecido al enlazarse la biblioteca.

Por ejemplo, en un sistema basado en UNIX, 'mimodelo!función8' indicaría al gestor de bases de datos que debe buscar la biblioteca $dir_inst_inicial/sqllib/function/mimodelo y que debe utilizar el punto de entrada función8 que está dentro de esa biblioteca.

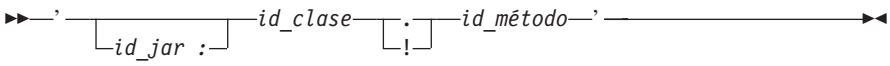
En los sistemas operativos Windows, 'mimodelo!función8' indicaría al gestor de bases de datos que debe cargar el archivo mimodelo.dll y que debe llamar a la función función8() en la biblioteca de enlace dinámico (DLL).

Si la composición de la serie de caracteres no es correcta, se produce un error (SQLSTATE 42878).

El cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste deberá existir cuando se ejecute la sentencia CREATE FUNCTION. Sin embargo, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).



Dentro de los apóstrofos no puede haber ningún espacio en blanco adicional.

id_jar
Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'.

id_clase
Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo 'miPacks.UserFuncs'. La

máquina virtual Java buscará en el directorio '.../miPacks/UserFuncs/' correspondiente a las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '...\miPacks\UserFuncs\'.

id_método
Identifica el nombre de método del objeto Java que se invoca.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se ejecuta la sentencia CREATE FUNCTION. Sin embargo, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42724).

  Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.

  ```
  ►►—'—ensamblaje—:—id_clase—!—id_método—'—————————————————►◄
  ```

  El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

  ensamblaje
  Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de instalación de DB2 (por ejemplo, c:\sqllib\function). Si el archivo reside en un subdirectorio del directorio function de la instalación, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instalación es c:\sqllib y el archivo del ensamblaje es c:\sqllib\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

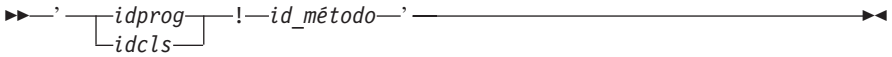
  id_clase
  Especifica el nombre de la clase del ensamblaje determinado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el

compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id_método
Especifica el método de la clase determinada que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La *serie* especificada es el identificador del programa OLE (idprog) o identificador de clase (idcls) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando (CREATE). No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al emitir la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).



  Dentro de los apóstrofos no puede haber ningún espacio en blanco adicional.

  idprog
  Identifica el identificador de programa del objeto OLE.

  idprog no se interpreta por el gestor de bases de datos sino que sólo se reenvía a la API OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y dar soporte a un enlace lógico posterior (denominado también enlace lógico basado en IDispatch).

  idcls
  Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa para especificar *idprog* en el caso de que el objeto OLE no esté registrado con un idprog. El *idcls* tiene el formato:

  {nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

  donde 'n' es un carácter alfanumérico. *idcls* no se interpreta por el gestor de bases de datos, sólo se reenvía a las API OLE en el momento de la ejecución.

  id_método
  Identifica el nombre de método del objeto OLE que se debe invocar.

NAME *identificador*
Este *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre de esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo de la función definida por el usuario.

C Esto significa que el gestor de bases de datos llamará a la función

definida por el usuario como si se tratase de una función en C. La función definida por el usuario debe ajustarse al convenio de llamadas y enlaces del lenguaje C, tal y como se define en el prototipo de C estándar de ANSI.

JAVA Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase Java.

CLR Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En este momento, sólo se da soporte a LANGUAGE CLR para las funciones definidas por el usuario que se ejecutan en sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para las funciones definidas por el usuario que se han almacenado en DB2 para sistemas operativos Windows. THREADSAFE no puede especificarse para las UDF que se han definido con LANGUAGE OLE (SQLSTATE 42613).

PARAMETER STYLE
Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2GENERAL
Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, las cuales están definidas como método en una clase Java. Sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

JAVA
Significa que la función utilizará un convenio para el envío de parámetros que se ajusta a la especificación para el lenguaje Java y las Rutinas SQLJ. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA y no existen tipos estructurados como tipos de parámetro o de retorno (SQLSTATE 429B8). Las funciones PARAMETER STYLE JAVA no dan soporte a las cláusulas FINAL CALL, SCRATCHPAD ni DBINFO.

SQL
Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo de gráfico ni tipos definidos por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, se pueden crear funciones con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. Un ejemplo de una función NOT DETERMINISTIC sería un generador de números aleatorios. Un ejemplo de una función DETERMINISTIC sería una función que determinara la raíz cuadrada de los datos de entrada.

FENCED o **NOT FENCED**
Esta cláusula especifica si se considera que la función es “segura” o no lo es para ejecutarse en un proceso o espacio de dirección del entorno operativo del gestor de bases de datos.

Si una función se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. La mayoría de las funciones tienen la opción de ejecutarse como FENCED o NOT FENCED. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para funciones no codificadas correctamente puede comprometer la integridad de DB2. DB2 dispone de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no puede garantizar la integridad completa cuando se utilizan funciones NOT FENCED definidas por el usuario.

Para una función con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar una función definida por el usuario como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

| No se pueden crear funciones LANGUAGE CLR definidas por el usuario al
| especificar la cláusula NOT FENCED (SQLSTATE 42601).

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

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

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

Para las funciones FENCED, THREADSAFE es el valor por omisión si
| LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o **CALLED ON NULL INPUT**
Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la ejecución, cualquiera de los argumentos de la función es nulo, no se invoca la función definida por el usuario y el resultado es el valor nulo.

Si se especifica CALLED ON NULL INPUT, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero corresponde a la UDF comprobar si los valores de los argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO SQL, CONTAINS SQL, READS SQL DATA
Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL
Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

NO EXTERNAL ACTION o **EXTERNAL ACTION**
Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que la función no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

NO SCRATCHPAD o **SCRATCHPAD** *longitud*
Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Es muy recomendable que las funciones definidas por el usuario sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que la función "guarde el estado" entre una llamada y la siguiente.)

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación de la función definida por el usuario, se asigna memoria para que la función externa utilice una memoria de trabajo. Esta memoria de trabajo tiene las siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El tamaño por omisión es 100 bytes.
- El valor de inicialización consta sólo de ceros hexadecimales (X'00').
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para referencia a la función externa en la sentencia de SQL. Por tanto, si la función UDFX de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarían tres memorias de trabajo.

```
SELECT A, UDFX(A) FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si la función se ejecuta en múltiples particiones, se asigna una memoria de trabajo en cada partición donde se procesa la función, para cada referencia a la función en la sentencia de SQL. De manera similar, si la consulta se ejecuta con el paralelismo de intrapartición habilitado, pueden asignarse más de tres memorias de trabajo.

- Su contenido se conserva. Se conserva el contenido de una llamada de función externa a la llamada siguiente. Los cambios hechos en la memoria de trabajo por la función externa en una llamada seguirán allí en la llamada siguiente. El gestor de bases de datos inicializa las memorias de trabajo al principio de la ejecución de cada sentencia de SQL. El gestor de bases de datos puede restaurar las memorias de trabajo al comienzo de la ejecución de cada subconsulta. El sistema emite una llamada final antes de restaurar una memoria de trabajo si se especifica la opción FINAL CALL.
- Puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir la función externa. La función podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

  (En el caso en que se adquiere el recurso del sistema, también debe especificarse la palabra clave FINAL CALL; esto provoca una llamada especial al fin de sentencia dirigida a permitir que la función externa libere cualquier recurso del sistema adquirido.)

Si se especifica SCRATCHPAD, en cada invocación de la función definida por el usuario se pasa un argumento adicional a la función externa que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo a la función externa.

SCRATCHPAD no puede utilizarse para funciones PARAMETER STYLE JAVA.

FINAL CALL o **NO FINAL CALL**

Esta cláusula opcional especifica si debe realizarse una llamada final a una función externa. La finalidad de la misma es hacer que la función externa libere los recursos del sistema que haya adquirido. Junto con la palabra clave SCRATCHPAD, puede ser útil en situaciones donde la función externa adquiera recursos del sistema tales como memoria y los fije en la memoria de trabajo. Si se especifica FINAL CALL, durante la ejecución se produce lo siguiente:

- Se pasa un argumento adicional a la función externa que especifica el tipo de llamada. Los tipos de llamada son:
  - Llamada normal: Se pasan los argumentos SQL y se espera la devolución de un resultado.
  - Primera llamada: la primera llamada a la función externa para esta referencia a la función definida por el usuario en esta sentencia de SQL. La primera llamada es una llamada normal.
  - Llamada final: una llamada final a la función externa para permitir que la función libere recursos. La llamada final no es una llamada normal. Esta llamada final tiene lugar en las siguientes circunstancias:
    - Fin de sentencia: Este caso se produce cuando se cierra el cursor para las sentencias orientadas a cursor o cuando la sentencia termina la ejecución de otra manera.

- Fin de tarea paralela: Este caso se produce cuando la función la ejecutan tareas paralelas.
- Fin de transacción o interrupción: Este caso se produce cuando no tiene lugar la finalización normal de la sentencia. Por ejemplo, cuando por alguna razón la lógica de una aplicación ignora el cierre del cursor. Durante este tipo de llamada de finalización, no puede emitirse ninguna sentencia de SQL, a excepción del cursor CLOSE (SQLSTATE 38505). Este tipo de llamada final se indica con un valor especial en el argumento de "tipo de llamada".

Si se produce una operación de confirmación mientras está abierto un cursor definido como WITH HOLD, se realiza una llamada final en el cierre subsiguiente del cursor o al final de la aplicación.

Si no se especifica NO FINAL CALL, no se pasa ningún argumento de "tipo de llamada" a la función externa y no se realiza ninguna llamada final.

FINAL CALL no recibe soporte para funciones PARAMETER STYLE JAVA.

ALLOW PARALLEL o **DISALLOW PARALLEL**

Esta cláusula opcional especifica si, para una referencia individual a la función, puede paralelizarse la invocación de la función. En general, las invocaciones de la mayoría de funciones escalares deben poder paralelizarse, pero pueden haber funciones (por ejemplo, las que dependen de una sola copia de una memoria de trabajo) que no puedan. Si se especifica ALLOW PARALLEL o DISALLOW PARALLEL para una función escalar, entonces DB2 aceptará esta especificación. Deben tenerse en cuenta las siguientes cuestiones al determinar qué palabra clave es la adecuada para la función.

- ¿Son todas las invocaciones de la UDF completamente independientes las unas de las otras? En caso afirmativo, especifique ALLOW PARALLEL.
- ¿Se actualiza la memoria de trabajo con cada invocación de la UDF, proporcionando valores que son de interés para la siguiente invocación? (Por ejemplo, el incremento de un contador.) En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Realiza la UDF alguna acción externa que sólo deba producirse en una sola partición? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique ALLOW PARALLEL.

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

El valor por omisión es ALLOW PARALLEL, excepto si se especifica una o varias de las opciones siguientes en la sentencia.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

Si se especifica o está implícita alguna de estas opciones, el valor por omisión es DISALLOW PARALLEL.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la

sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

NO DBINFO o **DBINFO**
Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 a la UDF como un argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no está soportada para LANGUAGE OLE (SQLSTATE 42613) ni para PARAMETER STYLE JAVA.

Si se especifica DBINFO, entonces se pasa una estructura a la UDF que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, independientemente de las UDF anidadas existentes entre esta UDF y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - si se dan las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - sólo si la referencia a UDF está en el lado derecho de una cláusula SET en una sentencia UPDATE o un elemento de la lista VALUES de una sentencia INSERT, contiene el nombre no calificado de la tabla que se está actualizando o insertando; de lo contrario, está en blanco.
- Nombre de columna - exactamente bajo las mismas condiciones que para el Nombre de tabla, contiene el nombre de la columna que se está actualizando o insertando; de lo contrario está en blanco.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca la UDF.
- Plataforma - contiene el tipo de plataforma del servidor.
- Números de columna del resultado de función de la tabla - no es aplicable a las funciones escalares externas.

TRANSFORM GROUP *nombre-grupo*
Indica el grupo de transformación que se debe utilizar, cuando se invoca la función, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación si la definición de la función incluye un tipo estructurado definido por el usuario, ya sea como parámetro o como tipo de datos. Si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, DB2_FUNCTION. Si el *nombre-grupo* especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Si una función de transformación necesaria FROM SQL o TO SQL no está definida para el nombre de grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

Las funciones de transformación, FROM SQL y TO SQL, ya sea especificadas explícita o implícitamente, deben ser funciones SQL que realicen una transformación adecuada entre el tipo estructurado y sus atributos de tipo incorporados.

PREDICATES
Define el filtrado o la utilización de la extensión de índice que se lleva a cabo cuando la función se utiliza en un predicado. Una especificación de predicado permite especificar la cláusula opcional SELECTIVITY de una condición de búsqueda. Si se especifica la cláusula PREDICATES, la función debe definirse como DETERMINISTIC con NO EXTERNAL ACTION (SQLSTATE 42613). Si se especifica la cláusula PREDICATES y la base de datos no es Unicode, no se debe especificar PARAMETER CCSID UNICODE (SQLSTATE 42613).

WHEN *operador-comparación*
Determina un uso especifico de la función en un predicado mediante un operador de comparación ("=", "<", ">", ">=", "<=", "<>").

constante
Especifica un valor constante con un tipo de datos comparable con el tipo RETURNS de la función (SQLSTATE 42818). Cuando un predicado utiliza esta función con el mismo operador de comparación y esta constante, el optimizador puede utilizar el filtrado y la explotación del índice.

EXPRESSION AS *nombre-expresión*
Proporciona un nombre para una expresión. Cuando un predicado utiliza esta función con el mismo operador de comparación y una expresión, puede utilizarse el filtrado y la explotación del índice. La expresión se asigna como nombre de expresión, para que pueda utilizarse como argumento de una función de búsqueda. El *nombre-expresión* no puede ser igual a ningún *nombre-parámetro* de la función que se está creando (SQLSTATE 42711). Cuando se especifica una expresión, se identifica el tipo de la expresión.

FILTER USING
Permite especificar la utilización de una función externa o expresión CASE para un mayor filtrado de la tabla de resultados.

invocación-función
Especifica una función de filtro que se puede utilizar para realizar un filtrado adicional de la tabla de resultados. Esto es una versión de la función definida (utilizada en el predicado) que reduce el número de filas en el predicado definido por el usuario, para determinar si las filas cumplen los requisitos. Si los resultados producidos por el índice son cercanos a los resultados previstos para el predicado definido por el usuario, puede no ser necesario aplicar la función de filtrado. Si no se especifica esta opción, no se realiza el filtrado de datos.

Esta función puede utilizar como argumento cualquier *nombre-parámetro*, el *nombre-expresión* o constantes (SQLSTATE 42703), y devuelve un entero (SQLSTATE 428E4). Si se devuelve el valor 1, significa que se conserva la fila; en otro caso se descarta.

Esta función también debe cumplir estas condiciones:

- no estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- no estar definida con NOT DETERMINISTIC ni EXTERNAL ACTION (SQLSTATE 42845)

- no tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- no incluir una subconsulta (SQLSTATE 428E4).

Si un argumento invoca otra función u otro método, estas cuatro normas también se aplican para la función o el método anidado. Sin embargo, los métodos de observador generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o método utilizado como argumento), siempre que el resultado de evaluar el argumento sea un tipo de datos incorporado.

El usuario que define la función debe disponer de privilegio EXECUTE para la función de filtro especificada.

La cláusula *invocación-función* no debe exceder de 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

expresión-case
Especifica una expresión CASE para realizar un mayor filtrado de la tabla de resultados. La *cláusula-when-buscada* y la *cláusula-when-simple* pueden utilizar *nombre-parámetro*, *nombre-expresión* o una constante (SQLSTATE 42703). Como expresión-resultado se puede utilizar una función externa con las normas especificadas en FILTER USING *invocación-función*. Cualquier función o método al que se hace referencia en la *expresión-case* también debe seguir las cuatro normas indicadas en la *invocación-función*.

No se pueden utilizar subconsultas en ningún lugar de la *expresión-case* (SQLSTATE 428E4).

La expresión case debe devolver un valor entero (SQLSTATE 428E4). Un valor de retorno de 1 en la expresión-resultado significa que se conserva la fila; de lo contrario, se elimina.

La cláusula *invocación-case* no debe exceder de 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

explotación-índice
Define un conjunto de normas para el método de búsqueda de una extensión de índice que se puede utilizar para explotar el índice.

SEARCH BY INDEX EXTENSION *nombre-extensión-índice*
Identifica la extensión de índice. El *nombre-extensión-índice* debe identificar una extensión de índice existente.

EXACT
Indica que la búsqueda por índice es exacta respecto a la evaluación del predicado. Utilice EXACT para indicar a DB2 que no es necesario aplicar la función de predicado original definida por el usuario ni el filtro después la búsqueda por índice. El predicado definido como EXACT es útil cuando la búsqueda por índice devuelve los mismos resultados que el predicado.

Si no se especifica EXACT, después de la búsqueda por índice se aplica el predicado original definido por el usuario. Si se prevé que el índice sólo proporcione una aproximación del predicado, no especifique la opción EXACT.

Si no se utiliza la búsqueda por índice, es necesario aplicar la función de filtro y el predicado original.

norma-explotación
Describe los patrones y argumentos de la búsqueda y cómo se pueden utilizar para realizar una búsqueda por índice mediante un método de búsqueda definido en la extensión de índice.

WHEN KEY (*nombre-parámetro1***)**
Define el patrón de búsqueda. Se puede especificar un solo patrón de búsqueda para una clave. El valor *nombre-parámetro1* identifica los nombres de parámetros de la función definida (SQLSTATE 42703 ó 428E8).

El tipo de datos de *nombre-parámetro1* debe coincidir con el de la clave de origen especificada en la extensión de índice (SQLSTATE 428EY). La coincidencia debe ser exacta para los tipos de datos incorporados y diferenciados y debe darse dentro de la misma jerarquía de tipos estructurados para los tipos estructurados.

El resultado de esta cláusula es verdadero cuando los valores del parámetro indicado son columnas que están abarcadas por un índice, de acuerdo con la extensión de índice especificada.

USE *nombre-método-búsqueda***(***nombre-parámetro2,...* **)**
Define el argumento de búsqueda. Identifica qué método de búsqueda utilizar de entre los definidos en la extensión de índice. El *nombre-método-búsqueda* debe coincidir con un método de búsqueda definido en la extensión de índice (SQLSTATE 42743). Los valores de *nombre-parámetro2* identifican nombres de parámetros de la función definida o el *nombre-expresión* de la cláusula EXPRESSION AS (SQLSTATE 42703). Debe ser diferente de cualquier nombre de parámetro especificado en el patrón de búsqueda (SQLSTATE 428E9). El número de parámetros y el tipo de datos de cada *nombre-parámetro2* debe coincidir con los parámetros definidos para el método de búsqueda en la extensión de índice (SQLSTATE 42816). La coincidencia debe ser exacta para los tipos de datos incorporados y diferenciados y debe darse dentro de la misma jerarquía de tipos en el caso de tipos estructurados.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - ASUTIME NO LIMIT
      - NO COLLID
      - PROGRAM TYPE SUB
      - STAY RESIDENT NO
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL
    - NOT VARIANT puede utilizarse en lugar de DETERMINISTIC, y VARIANT puede especificarse en lugar de NOT DETERMINISTIC

  - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT, y NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT

- La determinación de si un tipo de datos es convertible a otro no tiene en cuenta la longitud, precisión ni escala de los tipos de datos con parámetros, como son CHAR y DECIMAL. Por esta razón, pueden producirse errores al utilizar una función como resultado del intento de conversión de un valor del tipo de datos fuente a un valor del tipo de datos de destino. Por ejemplo, VARCHAR es convertible a DATE, pero si el tipo de fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, considere las normas de promoción que afectarán a sus valores de entrada (consulte el apartado "Promoción de tipos de datos"). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  – INTEGER en lugar de SMALLINT
  – DOUBLE en lugar de REAL
  – VARCHAR en lugar de CHAR
  – VARGRAPHIC en lugar de GRAPHIC
- Para asegurar la portabilidad de las UDF de una plataforma a otra, no deben utilizarse los tipos de datos siguientes:
  – FLOAT- utilice DOUBLE o REAL en su lugar.
  – NUMERIC- utilice DECIMAL en su lugar.
  – LONG VARCHAR- utilice CLOB (o BLOB) en su lugar.
- Una función y un método no pueden estar en una relación de alteración temporal (SQLSTATE 42745). Para obtener más información acerca de la alteración temporal, consulte "CREATE TYPE (Estructurado)".
- Una función no puede tener la misma signatura que un método (cuando se compara el primer *tipo-parámetro* de la función con el *tipo-sujeto* del método) (SQLSTATE 42723).
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- En un entorno de bases de datos particionadas, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Si la función permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- ***Restricciones de acceso a las tablas***

  Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se

invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.

- *Privilegios*
  - El usuario que define una función siempre recibe el privilegio EXECUTE y WITH GRANT para la función, así como el derecho para poder eliminar la función.
  - Cuando la función se utiliza en una sentencia de SQL, el usuario que define la función debe disponer de privilegio EXECUTE para cualquiera de los paquetes que la función utiliza.

Ejemplos:

Ejemplo 1: Pellow registra la función CENTRE en su esquema PELLOW. Permitiremos que las palabras clave que se tomarán por omisión se encarguen de realizar tal acción y que el sistema proporcione un nombre específico de la función:

```
CREATE FUNCTION CENTRE (INT,FLOAT)
  RETURNS FLOAT
  EXTERNAL NAME 'mod!middle'
  LANGUAGE C
  PARAMETER STYLE SQL
  DETERMINISTIC
  NO SQL
  NO EXTERNAL ACTION
```

Ejemplo 2: Ahora, McBride (que tiene la autorización DBADM) registra otra función CENTRE en el esquema PELLOW, dándole un nombre explícito específico para el uso posterior del lenguaje de definición de datos y proporcionando explícitamente todos los valores de palabras clave. Observe que esta función utiliza una memoria de trabajo y que, presumiblemente, está acumulando datos que afectan a los resultados posteriores. Debido a que está especificado DISALLOW PARALLEL, cualquier referencia a la función no se paraleliza y, por lo tanto, se utiliza una sola memoria de trabajo para realizar cierta inicialización una sola vez y guardar los resultados.

```
CREATE FUNCTION PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
  RETURNS DECIMAL(8,4) CAST FROM FLOAT
  SPECIFIC FOCUS92
  EXTERNAL NAME 'effects!focalpt'
  LANGUAGE C   PARAMETER STYLE SQL
  DETERMINISTIC   FENCED   NOT NULL CALL   NO SQL   NO EXTERNAL ACTION
  SCRATCHPAD   NO FINAL CALL
  DISALLOW PARALLEL
```

Ejemplo 3: A continuación se muestra el programa de función definida por el usuario en lenguaje C para implantar esta norma:

```
output = 2 * input - 4
```

devolviéndose NULL si y sólo si la entrada es nula. Podría haberse escrito incluso de forma más sencilla (es decir, sin comprobación de nulos) si la sentencia CREATE FUNCTION hubiera utilizado NOT NULL CALL. La sentencia CREATE FUNCTION:

```
CREATE FUNCTION ntest1 (SMALLINT)
  RETURNS SMALLINT
  EXTERNAL NAME 'ntest1!nudft1'
  LANGUAGE C    PARAMETER STYLE SQL
  DETERMINISTIC   NOT FENCED   NULL CALL
  NO SQL   NO EXTERNAL ACTION
```

El código del programa:

```
#include "sqlsystm.h"
/* NUDFT1 ES UNA FUNCIÓN ESCALAR DEFINIDA POR EL USUARIO */
/* udft1 acepta como entrada argumentos smallint
y produce como salida argumentos smallint
e implanta la norma:
if (input is null)
set output = null;
else
set output = 2 * input - 4;
*/
void SQL_API_FN nudft1
(short *input,        /* puntero al argumento de entrada */
short *output,        /* puntero al resultado */
short *input_ind,  /* puntero a variable de indicador de entrada */
short *output_ind, /* puntero a variable de indicador de salida */
char sqlstate[6],  /* sqlstate, permite término nulo */
char fname[28],    /* nombre función calificado al completo, termino nulo */
char finst[19],    /* nombre específico función, término nulo */
char msgtext[71])  /* almacenamiento intermedio texto mensaje, término nulo */
{
/* comprobar primero si la entrada es nula */
if (*input_ind == -1)
{
/* si la entrada es nula, hacer nula la salida */
*output_ind = -1;
}
else
{
/* si la entrada no es nula, establecer la salida en 2*input-4 */
*output = 2 * (*input) - 4;
/* establecer el indicador de salida nula en cero */
*output_ind = 0;
}
/* marcar finalización correcta dejando sqlstate sin cambiar */
/* y salir */
return;
}
/* end of UDF: NUDFT1 */
```

Ejemplo 4: Lo siguiente registra una UDF Java que devuelve la posición de la primera volcar de una serie. La UDF está escrita en Java, se debe ejecutar con restricciones de recursos y es el método findvwl de clase javaUDFs.

```
CREATE FUNCTION findv ( CLOB(100K))
  RETURNS INTEGER
  FENCED
  LANGUAGE JAVA
  PARAMETER STYLE JAVA
  EXTERNAL NAME 'javaUDFs.findvwl'
  NO EXTERNAL ACTION
  CALLED ON NULL INPUT
  DETERMINISTIC
  NO SQL
```

Ejemplo 5: Este ejemplo describe un predicado definido por el usuario, WITHIN, que utiliza como entrada dos parámetros, g1 y g2, de tipo SHAPE:

```
CREATE FUNCTION within (g1 SHAPE, g2 SHAPE)
RETURNS INTEGER
LANGUAGE C
PARAMETER STYLE SQL
NOT VARIANT
NOT FENCED
NO SQL
NO EXTERNAL ACTION
```

```
EXTERNAL NAME 'db2sefn!SDESpatilRelations'
PREDICATES
WHEN = 1
FILTER USING mbrOverlap(g1..xmin, g1..ymin, g1..xmax, g1..max,
g2..xmin, g2..ymin, g2..xmax, g2..ymax)
SEARCH BY INDEX EXTENSION gridIndex
WHEN KEY(g1) USE withinExplRule(g2)
WHEN KEY(g2) USE withinExplRule(g1)
```

La descripción de la función WITHIN es similar a la de cualquier función definida por el usuario, pero las adiciones siguientes indican que esta función se puede utilizar en un predicado definido por el usuario.

- **PREDICATES WHEN = 1** indica que cuando esta función aparezca como

  ```
  within(g1, g2) = 1
  ```

  en la cláusula WHERE de una sentencia DML, el predicado debe tratarse como un predicado definido por el usuario y el índice definido por la extensión de índice *gridIndex* se debe utilizar para recuperar las filas que cumplen las condiciones de este predicado. Si se especifica una constante, la constante especificada durante la sentencia DML debe coincidir exactamente con la constante especificada en la sentencia CREATE INDEX. Esta condición se proporciona principalmente para abarcar las expresiones booleanas donde el tipo resultante es un 1 o un 0. En los demás casos, la cláusula EXPRESSION es una elección mejor.
- **FILTER USING mbrOverlap** hace referencia a la función de filtro mbrOverlap, que es una versión más sencilla del predicado WITHIN. En el ejemplo anterior, la función mbrOverlap utiliza como entrada los rectángulos delimitadores mínimos y determina rápidamente si se solapan o no. Si los rectángulos delimitadores mínimos de las dos figuras de entrada no se solapan, significa que g1 no está contenido en g2. Por tanto, el tuplo se puede descartar sin riesgo, evitando la aplicación del costoso predicado WITHIN.
- La cláusula **SEARCH BY INDEX EXTENSION** indica qué combinaciones de extensión de índice y patrón de búsqueda se puede utilizar para este predicado definido por el usuario.

Ejemplo 6: Este ejemplo describe un predicado definido por el usuario, DISTANCE, que utiliza como entrada dos parámetros, P1 y P2, de tipo POINT:

```
CREATE FUNCTION distance (P1 POINT, P2 POINT)
  RETURNS INTEGER
  LANGUAGE C
  PARAMETER STYLE SQL
  NOT VARIANT
  NOT FENCED
  NO SQL
  NO EXTERNAL ACTION
  EXTERNAL NAME 'db2sefn!SDEDistances'
  PREDICATES
  WHEN > EXPRESSION AS distExpr
  SEARCH BY INDEX EXTENSION gridIndex
  WHEN KEY(P1) USE distanceGrRule(P2, distExpr)
  WHEN KEY(P2) USE distanceGrRule(P1, distExpr)
```

La descripción de la función DISTANCE es similar a la de cualquier función definida por el usuario, pero las adiciones siguientes indican que cuando esta función se utiliza en un predicado, éste es un predicado definido por el usuario.

- **PREDICATES WHEN > EXPRESSION AS distExpr** es otra especificación válida del predicado. Cuando se especifica una expresión en la cláusula WHEN, el tipo

resultante de esa expresión se utiliza para determinar si el predicado es un predicado definido por el usuario de la sentencia DML. Por ejemplo:

```
SELECT T1.C1
       FROM T1, T2
       WHERE distance (T1.P1, T2.P1) > T2.C2
```

La especificación de predicado DISTANCE utiliza dos parámetros como entrada y compara los resultados con T2.C2, que es de tipo INTEGER. Debido a que sólo es significativo el tipo de datos de la expresión del lado derecho (a diferencia de cuando se utiliza una constante específica), es mejor elegir la cláusula EXPRESSION en el DDL CREATE FUNCTION para especificar un carácter comodín como valor de comparación.

Como método alternativo, también es válido el siguiente predicado definido por el usuario:

```
SELECT T1.C1
       FROM T1, T2
       WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
```

Existe actualmente la restricción de que sólo el lado derecho se trata como una expresión; el término en el lado izquierdo es la función definida por el usuario correspondiente al predicado definido por el usuario.

- La cláusula **SEARCH BY INDEX EXTENSION** indica qué combinaciones de extensión de índice y patrón de búsqueda se puede utilizar para este predicado definido por el usuario. En el caso de la función DISTANCE, la expresión designada como distExpr es también uno de los argumentos de búsqueda que se pasa a la función productora de rangos (definida como parte de la extensión de índice). El identificador de expresión se utiliza para definir una nombre para la expresión, que se pasa como argumento a la función productora de rangos.

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE TYPE (Estructurado)" en la página 433
- "CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 259
- "Sentencias de SQL que se permiten en rutinas" en la publicación *Consulta de SQL, Volumen 1*
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*
- "Promoción de tipos de datos" en la publicación *Consulta de SQL, Volumen 1*
- "Conversiones entre tipos de datos" en la publicación *Consulta de SQL, Volumen 1*

CREATE FUNCTION (Tabla externa)

Esta sentencia se utiliza para registrar una función escalar externa definida por el usuario en un servidor de aplicaciones.

En la cláusula FROM de una sentencia SELECT, se puede utilizar una *función de tabla* y ésta devuelve una tabla a SELECT de fila en fila.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la función no existe.
  - Privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función.

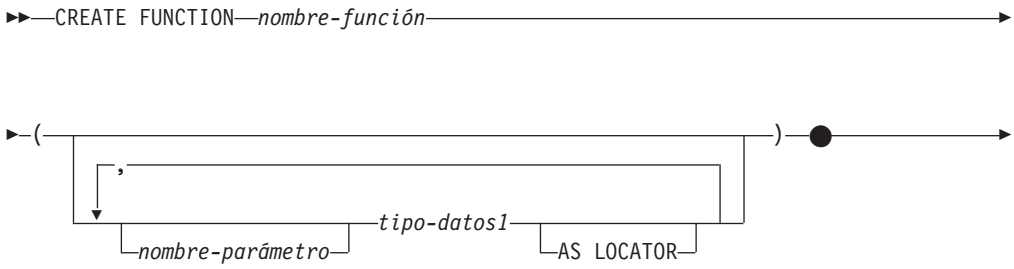
Para crear una función no restringida, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

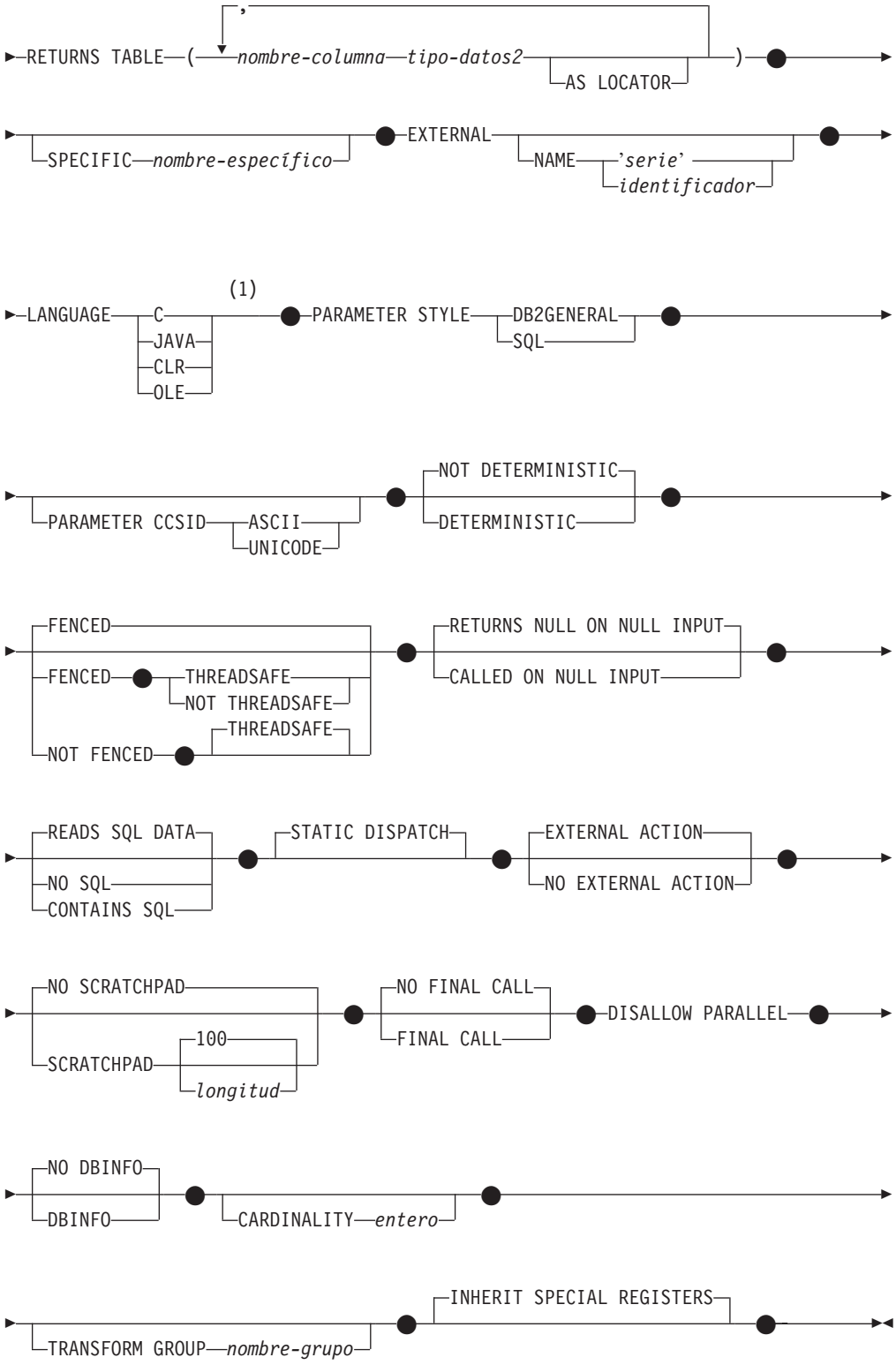
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM.

Para crear una función restringida, no se necesitan autorizaciones ni privilegios adicionales.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:





Notas:

1 Para obtener información acerca de la creación de funciones de tabla externa LANGUAGE OLE DB, consulte “CREATE FUNCTION (Tabla externa OLE

DB)". Para obtener información acerca de la creación de funciones de tabla LANGUAGE SQL, consulte "CREATE FUNCTION (Escalar de SQL, tabla o fila)".

Descripción:

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace especifica implícitamente el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre calificado no debe ser el mismo que el tipo de datos del primer parámetro, si ese parámetro es un tipo estructurado.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro

Especifica un nombre opcional para el parámetro que es diferente de los nombres de todos los demás parámetros de la función.

(tipo-datos1,...)

Identifica el número de parámetros de entrada de la función, al tiempo que especifica el tipo de datos de cada parámetro. En la lista debe especificarse una entrada por cada parámetro que la función espera recibir. No se permiten más de 90 parámetros. En caso de sobrepasarse este límite, se produce un error (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo

tipo, que son DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...

CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

la segunda sentencia y la cuarta fallarían porque se considera que son funciones duplicadas.

tipo-datos1
Especifica el tipo de datos del parámetro.

- Pueden proporcionarse especificaciones y abreviaturas de los tipos de datos SQL que pueden especificarse en la definición *tipo-datos* de una sentencia CREATE TABLE y que tienen una correspondencia en el lenguaje que se esté utilizando para escribir la función.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- REF(*nombre-tipo*) puede especificarse como tipo de datos de un parámetro. Sin embargo, dicho parámetro no debe tener ámbito (SQLSTATE 42997).
- Se pueden especificar tipos estructurados, con la condición de que existan las funciones de transformación apropiadas en el grupo de transformación asociado.

AS LOCATOR
Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB a la UDF en lugar del valor real. Esto reduce considerablemente el número de bytes que se pasan a la UDF y puede también mejorar el rendimiento, especialmente cuando la UDF sólo necesite unos pocos bytes.

El siguiente ejemplo ilustra la utilización de la cláusula AS LOCATOR en las definiciones de parámetros:

```
CREATE FUNCTION foo ( CLOB(10M) AS LOCATOR, IMAGE AS LOCATOR)
  ...
```

lo que supone que IMAGE es un tipo diferenciado basado en uno de los tipos LOB.

Observe también que para fines de promoción de argumentos, la cláusula AS LOCATOR no tiene ningún efecto. En el ejemplo, los tipos se consideran CLOB e IMAGE respectivamente, lo que significa que podría pasarse un argumento CHAR o VARCHAR a la función como el primer argumento. Igualmente, AS LOCATOR no tiene ningún efecto en la signatura de la función, que se utiliza en la comparación de la función (a) cuando se hace referencia en DML, por un proceso llamado ″resolución de función″ y (b) cuando se hace referencia en una

sentencia DDL como, por ejemplo, COMMENT ON o DROP. De hecho, la cláusula puede utilizarse o no en una sentencia COMMENT ON o DROP sin que tenga ningún significado.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS TABLE
Especifica que el resultado de la función es una tabla. El paréntesis que sigue a esta palabra clave delimita una lista de nombres y tipos de las columnas de la tabla, parecido al estilo de una sentencia CREATE TABLE simple que no tenga especificaciones adicionales (restricciones, por ejemplo). No están permitidas más de 255 columnas (SQLSTATE 54011).

nombre-columna
Especifica el nombre de esta columna. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

tipo-datos2
Especifica el tipo de datos de la columna y puede ser cualquier tipo de datos soportado para un parámetro de una UDF escrita en el lenguaje determinado, excepto para tipos estructurados (SQLSTATE 42997).

AS LOCATOR
Cuando *tipo-datos2* es un tipo LOB o un tipo diferenciado basado en un tipo LOB, la utilización de esta opción indica que la función devuelve un localizador para el valor LOB que tiene su instancia en la tabla de resultados.

Los tipos válidos que pueden utilizarse con esta cláusula se explican en "CREATE FUNCTION (Escalar externa)".

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar a otra instancia de la función que ya exista en el servidor de aplicaciones; de lo contrario, se produce un error. (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL
Esta cláusula indica que la sentencia CREATE FUNCTION se emplea para

registrar una nueva función basada en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

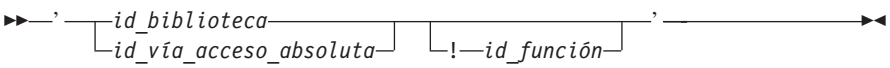
Si no se especifica la cláusula NAME, se da por supuesto "NAME *nombre-función*".

NAME *'serie'*
Esta cláusula identifica al código escrito por el usuario que implanta la función que se está definiendo.

La opción 'serie' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando (CREATE). Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función dentro de la biblioteca). No obstante, cuando se emplea esta función en una sentencia de SQL, la biblioteca y la función de la biblioteca sí deben existir y estar accesibles desde la máquina que actúa como servidor de bases de datos.



Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

id_biblioteca
Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

– En los sistemas basados en UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.

– En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

id_vía_acceso_absoluta
Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, si se especifica '/u/jchui/mylib/myfunc' en sistemas basados en UNIX, el gestor de bases de datos buscará la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id_función
Identifica el nombre del punto de entrada de la función que debe invocarse. El signo ! sirve como delimitador entre el ID de biblioteca

y el ID de función. Si se omite *! id_función*, el gestor de bases de datos utilizará el punto de entrada por omisión que se ha establecido al enlazarse la biblioteca.

Por ejemplo, en un sistema basado en UNIX, 'mimodelo!función8' indicaría al gestor de bases de datos que debe buscar la biblioteca $dir_inst_inicial/sqllib/function/mimodelo y que debe utilizar el punto de entrada función8 que está dentro de esa biblioteca.

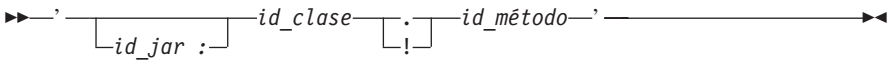
En los Sistemas operativos Windows de 32 bits, 'mimodelo!función8' indicaría al gestor de bases de datos que debe cargar el archivo mimodelo.dll y que debe llamar a la función función8() en la biblioteca de enlace dinámico (DLL).

Si la composición de la serie de caracteres no es correcta, se produce un error (SQLSTATE 42878).

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste deberá existir cuando se ejecute la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos.



  Dentro de los apóstrofos no puede haber ningún espacio en blanco adicional.

  id_jar
  Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'

  id_clase
  Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo 'miPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../miPacks/UserFuncs/' correspondiente a las clases. En los Sistemas operativos Windows de 32 bits, la máquina virtual Java buscará en el directorio '...\miPacks\UserFuncs\'.

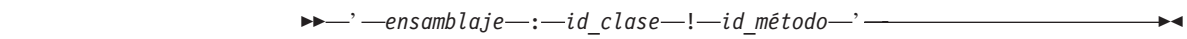
  id_método
  Identifica el nombre de método del objeto Java que se invoca.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se

ejecuta la sentencia CREATE FUNCTION. Sin embargo, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42724).

Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar las decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.



El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

ensamblaje
Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de instalación de DB2 (por ejemplo, c:\sqllib\function). Si el archivo reside en un subdirectorio del directorio function de la instalación, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instalación es c:\sqllib y el archivo de ensamblaje es c:\sqllib\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

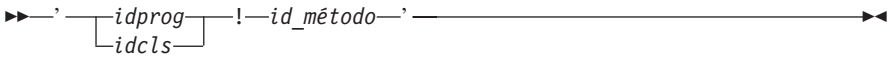
id_clase
Especifica el nombre de la clase del ensamblaje en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id_método
Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La *serie* especificada es el identificador del programa OLE (idprog) o identificador de clase (idcls) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando (CREATE). No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al emitir la sentencia CREATE FUNCTION. No obstante, cuando se utiliza

la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos, de lo contrario se produce un error (SQLSTATE 42724).



Dentro de los apóstrofos no puede haber ningún espacio en blanco adicional.

idprog
Identifica el identificador de programa del objeto OLE.

idprog no se interpreta por el gestor de bases de datos sino que sólo se reenvía a la API OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y dar soporte a un enlace lógico posterior (denominado también enlace lógico basado en IDispatch).

idcls
Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa para especificar *idprog* en el caso de que el objeto OLE no esté registrado con un idprog. El *idcls* tiene el formato:

{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}

donde 'n' es un carácter alfanumérico. *idcls* no se interpreta por el gestor de bases de datos, sólo se reenvía a las API OLE en el momento de la ejecución.

id_método
Identifica el nombre de método del objeto OLE que se debe invocar.

NAME *identificador*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo. El *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre de esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo de la función definida por el usuario.

C Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como si se tratase de una función en C. La función definida por el usuario debe ajustarse al convenio de llamadas y enlaces del lenguaje C, tal y como se define en el prototipo de C estándar de ANSI.

JAVA Esto significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase Java.

CLR Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En este momento, sólo se da soporte a LANGUAGE CLR para funciones definidas por el usuario que se ejecutan en sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

Sólo se da soporte al LANGUAGE OLE para las funciones definidas por el usuario almacenadas en DB2 para Sistemas operativos Windows de 32 bits.

Para obtener información acerca de la creación de funciones de tabla externa LANGUAGE OLE DB, consulte "CREATE FUNCTION (Tabla externa OLE DB)".

PARAMETER STYLE
Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2GENERAL
Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, las cuales están definidas como método en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

SQL
Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo de gráfico ni ningún tipo definido por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode, se pueden crear funciones de tabla con PARAMETER CCSID UNICODE, pero se aplican las normas siguientes:

- Se debe especificar un orden de clasificación alternativo en la configuración de la base de datos antes de crear la función de tabla (SQLSTATE 56031). Las funciones de tabla PARAMETER CCSID UNICODE se clasifican con el orden de clasificación alternativo especificado en la configuración de la base de datos.
- No se pueden utilizar conjuntamente las tablas o las funciones de tablas creadas con CCSID ASCII y las tablas o las funciones de tablas creadas con CCSID UNICODE en una sola sentencia de SQL (SQLSTATE 53090). Esto se aplica a las tablas y a las funciones de tabla a las que se hace referencia directamente en la sentencia, así como a las tablas y las funciones de tabla a las que se hace referencia indirectamente (por ejemplo, mediante restricciones de integridad referencial, activadores, tablas de consultas materializadas y tablas de los cuerpos de las vistas).
- No se puede hacer referencia a las funciones de tabla creadas con PARAMETER CCSID UNICODE en funciones de SQL ni en métodos de SQL (SQLSTATE 560C0).
- Una sentencia de SQL que hace referencia a una función de tabla creada con PARAMETER CCSID UNICODE no puede invocar una función de SQL ni un método de SQL (SQLSTATE 53090).
- Los tipos gráficos y los tipos definidos por el usuario no se pueden utilizar como parámetros para las funciones de tabla PARAMETER CCSID UNICODE (SQLSTATE 560C1).
- Las sentencias que hacen referencia a una función de tabla PARAMETER CCSID UNICODE sólo se pueden invocar desde un cliente DB2 Versión 8.1 o posterior (SQLSTATE 42997).
- Las sentencias de SQL siempre se interpretan en la página de códigos de la base de datos. En particular, significa que cada carácter de los literales, los literales hexadecimales y los identificadores delimitados debe tener una representación en la página de códigos de la base de datos; de lo contrario, el carácter se sustituirá por el carácter de sustitución.

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, las funciones se pueden crear con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. Un ejemplo de una función de tabla NOT DETERMINISTIC podría ser una función que recuperase datos de una fuente de datos como, por ejemplo, un archivo.

FENCED o **NOT FENCED**
Esta cláusula especifica si la función se considera o no “segura” para ejecutarse

en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED).

Si una función se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. La mayoría de las funciones tienen la opción de ejecutarse como FENCED o NOT FENCED. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para funciones no codificadas correctamente puede comprometer la integridad de DB2. DB2 toma algunas precauciones para muchas de las anomalías más habituales que podrían producirse, pero no puede asegurar la integridad completa si se emplean funciones definidas por el usuario como NOT FENCED.

Para una función con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar una función definida por el usuario como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

No se pueden crear funciones LANGUAGE CLR definidas por el usuario al especificar la cláusula NOT FENCED (SQLSTATE 42601).

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

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

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

Para las funciones FENCED, THREADSAFE es el valor por omisión si LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o **CALLED ON NULL INPUT**
Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la apertura de la función de tabla, cualquiera de los argumentos de la función es nulo, no se invoca la función definida por el usuario. El resultado de la función de tabla será una tabla vacía (una tabla sin ninguna fila).

Si se especifica CALLED ON NULL INPUT, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero corresponde a la UDF comprobar si los valores de argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO SQL, CONTAINS SQL, READS SQL DATA
Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL
Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

NO EXTERNAL ACTION o **EXTERNAL ACTION**
Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que la función no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

NO SCRATCHPAD o **SCRATCHPAD** *longitud*
Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Es muy recomendable que las funciones definidas por el usuario sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que la función “guarde el estado” entre una llamada y la siguiente.)

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación de la función definida por el usuario, se asigna memoria para que la función externa utilice una memoria de trabajo. Esta memoria de trabajo tiene las siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.

- El valor de inicialización consta sólo de ceros hexadecimales (X'00').
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para referencia a la función externa en la sentencia de SQL. Por tanto, si la función UDFX de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarían dos memorias de trabajo.

```
SELECT A.C1, B.C2
  FROM TABLE (UDFX(:hv1)) AS A,
       TABLE (UDFX(:hv1)) AS B
    WHERE ...
```

- Su contenido se conserva. Se inicializa al principio de la ejecución de la sentencia y puede ser utilizada por la función de tabla externa para guardar el estado de la memoria de trabajo entre una llamada y la siguiente. Si también se especifica la palabra clave FINAL CALL para la UDF, DB2 no altera NUNCA la memoria de trabajo y, los recursos anclados en ella deben liberarse cuando se emite la llamada especial FINAL.

  Si se especifica NO FINAL CALL o se acepta el valor por omisión, la función de tabla externa debe borrar tales recursos en la llamada CLOSE, pues DB2 reinicializará la memoria de trabajo en cada llamada OPEN. Esta elección entre FINAL CALL o NO FINAL CALL y el comportamiento asociado de la memoria de trabajo puede ser un factor importante, especialmente cuando la función de tabla se utiliza en una subconsulta o unión, pues es cuando pueden producirse varias llamadas OPEN durante la ejecución de una sentencia.
- Puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir la función externa. La función podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

  (Como se ha indicado anteriormente, la palabra clave FINAL CALL/NO FINAL CALL se utiliza para controlar la reinicialización de la memoria de trabajo y también determina cuándo la función de tabla externa debe liberar los recursos anclados en la memoria de trabajo.)

Si se especifica SCRATCHPAD, en cada invocación de la función definida por el usuario se pasa un argumento adicional a la función externa que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo a la función externa.

FINAL CALL o **NO FINAL CALL**

Esta cláusula opcional especifica si debe realizarse una llamada final (y una primera llamada aparte) a una función externa. También controla cuándo debe volver a inicializarse la memoria de trabajo. Si se especifica NO FINAL CALL, DB2 sólo puede realizar tres tipos de llamadas a la función de tabla: open (apertura), fetch (lectura) y close (cierre). En cambio, si se especifica FINAL CALL, además de las llamadas de apertura, lectura y cierre, pueden efectuarse una primera llamada y una llamada final a la función de tabla.

Para las funciones de tabla externa, el argumento tipo-llamada SIEMPRE está presente, sea cual sea la opción que se elija.

Si la llamada final está realizándose porque se ha producido una interrupción o un fin de transacción, puede que la UDF no emita ninguna sentencia de SQL, a excepción del cursor CLOSE (SQLSTATE 38505). Para estas situaciones especiales de la llamada final, se pasa un valor especial en el argumento del ″tipo de llamada″.

DISALLOW PARALLEL
Esta cláusula especifica que, para una única referencia a la función, la invocación de la función no puede ser en paralelo. Las funciones de tabla siempre se ejecutan en una sola partición.

NO DBINFO o **DBINFO**
Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 a la UDF como un argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613).

Si se especifica DBINFO, entonces se pasa una estructura a la UDF que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, independientemente de las UDF anidadas existentes entre esta UDF y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - no se puede aplicar a funciones de tabla externas.
- Nombre de tabla - no se puede aplicar a funciones de tabla externas.
- Nombre de columna - no se puede aplicar a funciones de tabla externas.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca la UDF.
- Plataforma - contiene el tipo de plataforma del servidor.
- Número de columna del resultado de la función de tabla - una matriz de los números de las columnas del resultado de la función de tabla realmente necesarios para la sentencia en particular que hace referencia a la función. Sólo se proporciona para las funciones de tabla, permite que la UDF optimice devolviendo sólo los valores de columna necesarios en lugar de todos los valores de columna.

CARDINALITY *entero*
Esta cláusula opcional proporciona una estimación del número esperado de filas que debe devolver la función con fines de optimización. Los valores válidos para *entero* están comprendidos dentro del rango 0 a 9 223 372 036 854 775 807, ambos inclusive.

Si no se especifica la cláusula CARDINALITY para una función de tabla, DB2 asumirá un valor finito como omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

Aviso: Si una función tiene, de hecho, una cardinalidad infinita — es decir, devuelve una fila cada vez que se llama para ello y nunca devuelve una condición de "fin de tabla" — las consultas que requieran una condición de fin de tabla para funcionar correctamente serán infinitas y se deberán interrumpir. Las consultas que contienen una cláusula GROUP BY o ORDER BY son ejemplos de esta clase de consultas. No se recomienda escribir estas UDF.

TRANSFORM GROUP *nombre-grupo*
Indica el grupo de transformación que se debe utilizar, cuando se invoca la función, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación si la definición de la función incluye un tipo estructurado definido por el usuario, en calidad de tipo de datos de

parámetro. Si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, DB2_FUNCTION. Si el *nombre-grupo* especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Si una función de transformación necesaria FROM SQL no está definida para el nombre de grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

Notas:

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas para la promoción que afectarán a sus valores de entrada. Por ejemplo, una constante que puede utilizarse como un valor de entrada podría tener un tipo de datos incorporado distinto del que se espera y, todavía más importante, podría no promoverse al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - INTEGER en lugar de SMALLINT
  - DOUBLE en lugar de REAL
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- Para garantizar la portabilidad de las UDF entre distintas plataformas, se recomienda utilizar los tipos de datos siguientes:
  - DOUBLE o REAL en lugar de FLOAT
  - DECIMAL en lugar de NUMERIC
  - CLOB (o BLOB) en lugar de LONG VARCHAR
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- En un entorno de bases de datos particionadas, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Si la función permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.

- *Restricciones de acceso a las tablas*

  Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.
- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 para z/OS y OS/390:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - ASUTIME NO LIMIT
      - NO COLLID
      - PROGRAM TYPE SUB
      - STAY RESIDENT NO
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL
    - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
    - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
    - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
    - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
    - DB2GENRL puede especificarse en lugar de DB2GENERAL.
- *Privilegios*
  - El usuario que define una función siempre recibe el privilegio EXECUTE y WITH GRANT para la función, así como el derecho para poder eliminar la función.
  - Cuando la función se utiliza en una sentencia de SQL, el usuario que define la función debe disponer de privilegio EXECUTE para cualquiera de los paquetes que la función utiliza.

Ejemplos:

Ejemplo 1: Lo siguiente registra una función de tabla escrita para devolver una fila que consta de una columna identificadora de un solo documento para cada documento conocido en un sistema de gestión de texto. El primer parámetro coincide con un área de temas determinada y el segundo parámetro contiene una serie determinada.

Dentro del contexto de una sola sesión, la UDF siempre devolverá la misma tabla y, por lo tanto, se define como DETERMINISTIC. Observe que la cláusula RETURNS que define la salida de DOCMATCH. Debe especificarse FINAL CALL para cada función de tabla. Además, se añade la palabra clave DISALLOW PARALLEL porque las funciones de tabla no pueden funcionar en paralelo. Aunque el tamaño de la salida para DOCMATCH es muy variable, el valor CARDINALITY 20 es representativo y se especifica para ayudar al optimizador DB2.

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
  LANGUAGE C
  PARAMETER STYLE SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

Ejemplo 2: Lo siguiente registra una función de tabla OLE que se utiliza para recuperar información de cabecera de mensajes y el texto parcial de los mensajes en Microsoft Exchange.

```
CREATE FUNCTION MAIL()
  RETURNS TABLE (TIMERECEIVED DATE,
                 SUBJECT VARCHAR(15),
                 SIZE INTEGER,
                 TEXT VARCHAR(30))
  EXTERNAL NAME 'tfmail.header!list'
  LANGUAGE OLE
  PARAMETER STYLE SQL
  NOT DETERMINISTIC
  FENCED
  CALLED ON NULL INPUT
  SCRATCHPAD
  FINAL CALL
  NO SQL
  EXTERNAL ACTION
  DISALLOW PARALLEL
```

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE FUNCTION (Tabla externa OLE DB)" en la página 241
- "CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 259
- "CREATE FUNCTION (Escalar externa)" en la página 198
- "Sentencias de SQL que se permiten en rutinas" en la publicación *Consulta de SQL, Volumen 1*
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*
- "Promoción de tipos de datos" en la publicación *Consulta de SQL, Volumen 1*

CREATE FUNCTION (Tabla externa OLE DB)

Esta sentencia se utiliza para registrar una función de tabla externa OLE DB para acceder a los datos de un proveedor OLE DB.

Puede utilizarse una *función de tabla* en la cláusula FROM de SELECT.

Invocación:

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

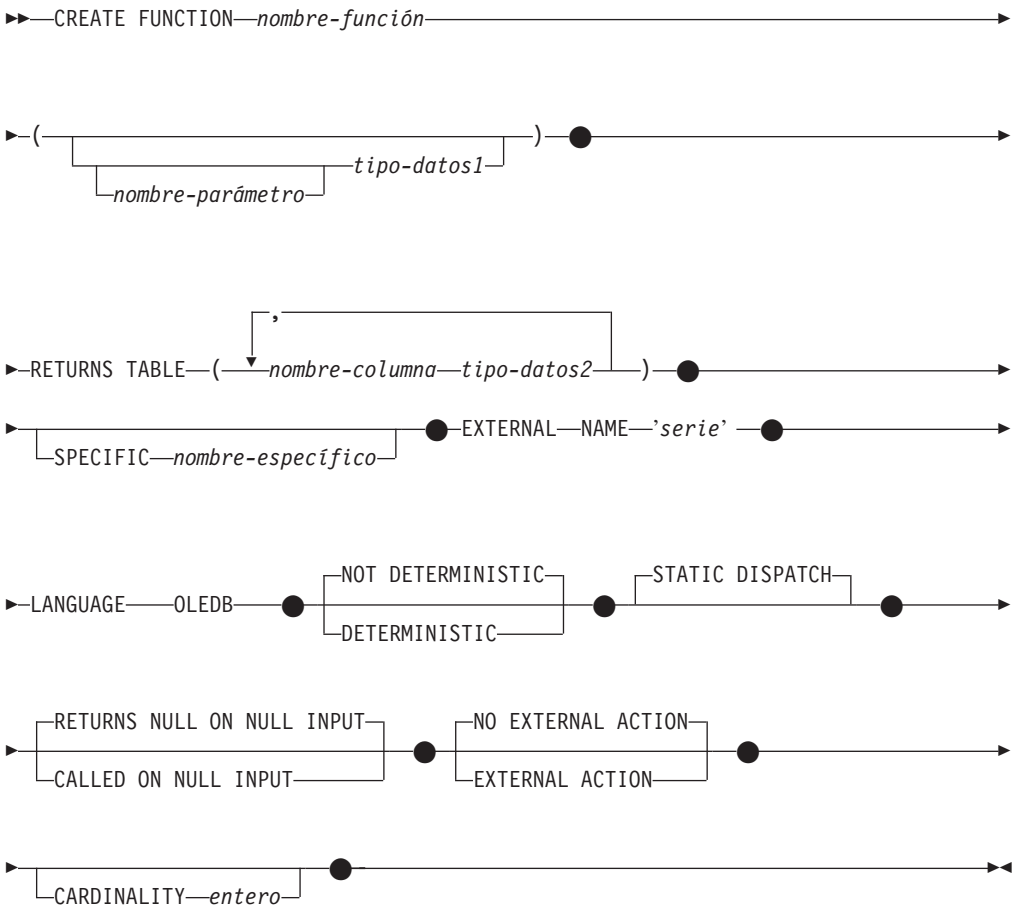
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la función no existe.
  - Privilegio CREATEIN para el esquema, si existe el nombre de esquema de la función.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:



Descripción:

nombre-función
Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace especifica implícitamente el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT,

AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro
Especifica un nombre opcional para el parámetro.

tipo-datos1
Identifica el parámetro de entrada de la función y el tipo de datos del parámetro. Si no se especifica ningún parámetro de entrada, los datos se recuperan de la fuente externa posiblemente subestablecida a través de la optimización de consulta. El parámetro de entrada puede ser cualquier tipo de datos de caracteres o de serie gráfica y pasa el texto del mandato al proveedor OLE DB.

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. No se tiene en cuenta la longitud en este tipo de comparación. Por lo tanto, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

RETURNS TABLE
Especifica que el resultado de la función es una tabla. El paréntesis que sigue a esta palabra clave delimita una lista de nombres y tipos de las columnas de la tabla, parecido al estilo de una sentencia CREATE TABLE simple que no tenga especificaciones adicionales (restricciones, por ejemplo).

nombre-columna
Especifica el nombre de la columna que debe ser igual que el nombre de columna del conjunto de filas correspondiente. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

tipo-datos2
Especifica el tipo de datos de la columna.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar a otra instancia de la función que ya exista en el servidor de aplicaciones; de lo contrario, se genera un error. (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

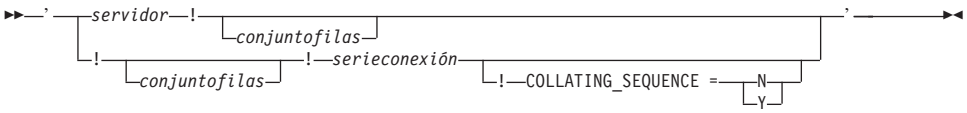
Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-función*, de lo contrario se genera un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL NAME *'serie'*
Esta cláusula identifica la tabla externa y el proveedor OLE DB.

La opción 'serie' es una constante de serie con un máximo de 254 caracteres.

La serie especificada se utiliza para establecer una conexión y una sesión con un proveedor OLE DB y recuperar los datos de un conjunto de filas. No es necesario que existan el proveedor OLE DB ni la fuente de datos cuando se ejecuta la CREATE FUNCTION.



servidor
Identifica el nombre local de una fuente de datos tal como "CREATE SERVER" lo ha definido.

conjuntofilas
Identifica el conjunto de filas (tabla) expuesto por el proveedor OLE DB. Deben proporcionarse nombres de tabla completamente calificados para los proveedores OLE DB que soportan nombres de catálogo o de esquema.

serieconexión
Versión de la serie de las propiedades de inicialización necesaria para conectarse a la fuente de datos. El formato básico de una serie de conexión se basa en la serie de conexión ODBC. La serie contiene una serie de pares de palabra clave/valor separados por puntos y comas. El signo igual (=) separa cada palabra clave y su valor. Las palabras clave son las descripciones de las propiedades de inicialización de OLE DB (conjunto de propiedades DBPROPSET_DBINIT) o palabras clave específicas del proveedor.

COLLATING_SEQUENCE
Especifica si la fuente de datos utiliza el mismo orden de clasificación que DB2 Universal Database. Para obtener detalles, consulte "CREATE SERVER". Los valores válidos son los siguientes:

Y = El mismo orden de clasificación

N = Diferente orden de clasificación

Si no se especifica COLLATING_SEQUENCE, se da por supuesto que la fuente de datos tiene un orden de clasificación distinto de DB2 Universal Database.

Si se proporciona el *servidor*, *serie-conexión* o COLLATING_SEQUENCE no están permitidos en el nombre externo. Se definen como las opciones de servidor CONNECTSTRING y COLLATING_SEQUENCE. Si no se proporciona ningún *servidor*, debe proporcionarse una *serieconexión*. Si no se proporciona

conjuntofilas, la función de tabla debe tener un parámetro de entrada para realizar el paso a través del texto del mandato al proveedor OLE DB.

LANGUAGE OLEDB
Esto significa que el gestor de bases de datos desplegará un cliente OLE DB genérico incorporado para recuperar los datos de OLE DB. El desarrollado no necesita ninguna implantación de función de tabla.

Las funciones de tabla LANGUAGE OLEDB pueden crearse en cualquier plataforma, pero sólo se ejecutan en plataformas soportadas por Microsoft OLE DB.

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

RETURNS NULL ON NULL INPUT o **CALLED ON NULL INPUT**
Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener parámetros, esta condición de argumento nulo no puede producirse.

Si se especifica RETURNS NULL ON NULL INPUT y si, en tiempo de ejecución, alguno de los argumentos de la función es nulo, no se llama a la función definida por el usuario y el resultado es la tabla vacía; es decir, una tabla sin filas.

Si se especifica CALLED ON NULL INPUT, entonces durante la ejecución sin tener en cuenta si los argumentos son o no nulos, se invoca la función definida por el usuario. Puede devolver una tabla vacía o no, dependiendo de su lógica. Pero corresponde a la UDF comprobar si los valores de los argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO EXTERNAL ACTION o **EXTERNAL ACTION**
Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar que se realicen optimizaciones en las que se da por supuesto que las funciones no tendrán ningún efecto externo, especifique EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

CARDINALITY *entero*
Esta cláusula opcional proporciona una estimación del número esperado de filas que debe devolver la función con fines de optimización. Los valores válidos de *entero* están en el rango de 0 a 2 147 483 647 inclusive.

Si no se especifica la cláusula CARDINALITY para una función de tabla, DB2 asumirá un valor finito como omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

Aviso: Si una función tiene, de hecho, una cardinalidad infinita — es decir, devuelve una fila cada vez que se llama para ello y nunca devuelve una condición de ″fin de tabla″ — las consultas que requieran una condición de fin de tabla para funcionar correctamente serán infinitas y se deberán interrumpir. Las consultas que contienen una cláusula GROUP BY o ORDER BY son ejemplos de esta clase de consultas. No se recomienda escribir estas UDF.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
    - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
    - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
    - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
- FENCED, FINAL CALL, SCRATCHPAD, PARAMETER STYLE SQL, DISALLOW PARALLEL, NO DBINFO, NOT THREADSAFE y NO SQL están implícitas en la sentencia y pueden especificarse.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas para la promoción que afectarán a sus valores de entrada. Por ejemplo, una constante que puede utilizarse como un valor de entrada podría tener un tipo de datos incorporado distinto del que se espera y, todavía más importante, podría no promoverse al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- Para garantizar la portabilidad de las UDF entre distintas plataformas, se recomienda utilizar los tipos de datos siguientes:
  - DOUBLE o REAL en lugar de FLOAT
  - DECIMAL en lugar de NUMERIC
  - CLOB (o BLOB) en lugar de LONG VARCHAR
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- ***Privilegios***

  El usuario que define una función siempre recibe el privilegio EXECUTE y WITH GRANT para la función, así como el derecho para poder eliminar la función.

Ejemplos:

Ejemplo 1: Lo siguiente registra una función de tabla OLE DB, que recupera información de clasificación de una base de datos Microsoft Access. La serie de conexión se define en el nombre externo.

```
CREATE FUNCTION orders ()
  RETURNS TABLE (orderid INTEGER,
                 customerid CHAR(5),
                 employeeid INTEGER,
                 orderdate TIMESTAMP,
                 requireddate TIMESTAMP,
                 shippeddate TIMESTAMP,
                 shipvia INTEGER,
                 freight dec(19,4))
  LANGUAGE OLEDB
  EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
                        Data Source=c:\sqllib\samples\oledb\nwind.mdb
   !COLLATING_SEQUENCE=Y';
```

Ejemplo 2: Lo siguiente registra una función de tabla OLE DB, que recupera información de cliente de una base de datos Oracle. La serie de conexión se proporciona a través de una definición de servidor. El nombre de tabla está completamente calificado en el nombre externo. El usuario local john se correlaciona con el usuario remoto dave. Los demás usuarios utilizarán el ID de usuario de invitado en la serie de conexión.

```
CREATE SERVER spirit
 WRAPPER OLEDB
 OPTIONS (CONNECTSTRING 'Provider=MSDAORA;Persist Security Info=False;
                         User ID=guest;password=pwd;Locale Identifier=1033;
                         OLE DB Services=CLIENTCURSOR;Data Source=spirit');

CREATE USER MAPPING FOR john
  SERVER spirit
   OPTIONS (REMOTE_AUTHID 'dave', REMOTE_PASSWORD 'mypwd');

CREATE FUNCTION customers ()
   RETURNS TABLE (customer_id INTEGER,
                  name  VARCHAR(20),
                  address VARCHAR(20),
                  city VARCHAR(20),
                  state VARCHAR(5),
                   zip_code INTEGER)
  LANGUAGE OLEDB
   EXTERNAL NAME 'spirit!demo.customer';
```

Ejemplo 3: Lo siguiente registra una función de tabla OLE DB, que recupera información sobre tiendas a partir de la base de datos MS SQL Server 7.0.La serie de conexión se proporciona en el nombre externo. La función de tabla tiene un parámetro de entrada para el paso a través del texto del mandato al proveedor OLE DB. No es necesario especificar el nombre del conjunto de filas en el nombre externo. La consulta del ejemplo pasa un texto de sentencia de SQL para recuperar las tres tiendas con mayor nivel de ventas.

```
CREATE FUNCTION favorites (varchar(600))
  RETURNS TABLE (store_id CHAR (4),
                 name  VARCHAR (41),
                 sales INTEGER)
  SPECIFIC favorites
  LANGUAGE OLEDB
  EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
                 User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;
                 Locale Identifier=1033;Use Procedure for Prepare=1;
                 Auto Translate=False;Packet Size=4096;Workstation ID=WALTZ;
                 OLE DB Services=CLIENTCURSOR;';

SELECT *
    FROM TABLE (favorites
                (' select top 3 sales.stor_id as store_id, ' ||
                    ' stores.stor_name as name, ' ||
```

```
    ' sum(sales. qty) as sales ' ||
' from sales, stores ' ||
' where sales.stor_id = stores.stor_id ' ||
' group by sales.stor_id, stores.stor_name ' ||
' order by sum(sales.qty) desc ')) as f;
```

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE SERVER" en la página 337
- "CREATE USER MAPPING" en la página 464
- "CREATE WRAPPER" en la página 481
- "CREATE FUNCTION (Tabla externa)" en la página 223
- "Promoción de tipos de datos" en la publicación *Consulta de SQL, Volumen 1*

CREATE FUNCTION (Con origen o plantilla)

Esta sentencia se utiliza para:

- Registrar una función definida por el usuario, basada en otra función escalar o de columna existente, en un servidor de aplicaciones.
- Registrar una plantilla de función en un servidor de aplicaciones que está designado como servidor federado. Una *plantilla de función* es una función parcial que no contiene código ejecutable. El usuario lo crea con la finalidad de correlacionarlo con una función fuente de datos. Después de crear la correlación, el usuario puede especificar la plantilla de función en consultas que se envían al servidor federado. Cuando se procesa una consulta como esta, el servidor federado invocará la función fuente de datos a la que está asignada la plantilla y devolverá los valores cuyos tipos de datos correspondan con los de la opción RETURNS de la definición de la plantilla.

Invocación:

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

Autorización:

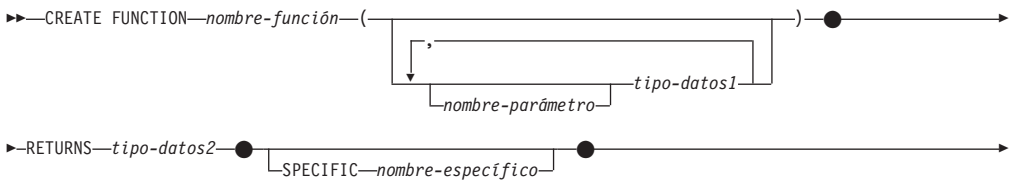
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

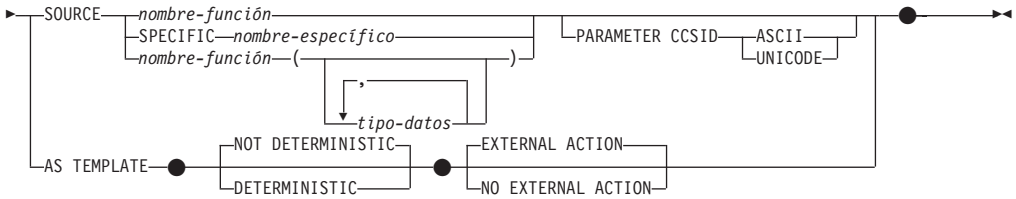
- El privilegio IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
- El privilegio CREATEIN en el esquema, si existe el nombre de esquema de la función
- Autorización SYSADM o DBADM

Los privilegios del ID de autorización de la sentencia también deben incluir el privilegio EXECUTE en la función fuente si el ID de autorización de la sentencia no tiene la autorización SYSADM o DBADM y se especifica la cláusula SOURCE.

Si el ID de autorización no tiene suficiente autorización para realizar la operación, se devuelve un error (SQLSTATE 42502).

Sintaxis:





Descripción:

nombre-función
Identifica la función o plantilla de función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función ni una plantilla de función descritos en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Al nombrar una función definida por el usuario que tenga su origen en una función ya existente con el fin de dar soporte a la misma función con un tipo diferenciado definido por el usuario, puede utilizarse el mismo nombre que el de la función con origen. De esta forma, los usuarios pueden utilizar la misma función con un tipo diferenciado definido por el usuario sin darse cuenta de que era necesaria una definición adicional. En general, puede utilizarse el mismo nombre para más de una función si existe alguna diferencia en la signatura de las funciones.

(tipo-datos, ...)
Identifica el número de parámetros de entrada de la función o plantilla de función y especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que la función o plantilla de función espera recibir. No se permiten más de 90 parámetros. Si se excede este límite, se produce un error (SQLSTATE 54023).

Se puede registrar una función o plantilla de función que no tenga ningún parámetro. En este caso, sigue siendo necesario escribir los paréntesis, sin ningún tipo de datos interpuesto. Por ejemplo,

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones o plantillas de función con idéntico nombre tengan exactamente el mismo tipo para todos los parámetros correspondientes. (Esta restricción también es aplicable a una función y plantilla de función de un esquema que tengan el mismo nombre). Las longitudes, precisiones y escalas no se tienen en cuenta en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, que son DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE FUNCTION PART (INT, CHAR(15)) ...
CREATE FUNCTION PART (INTEGER, CHAR(40)) ...

CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

la segunda y la cuarta sentencia fallarían porque se considera que son funciones duplicadas.

nombre-parámetro
Especifica un nombre opcional para el parámetro que es diferente de los nombres de todos los demás parámetros de la función.

tipo-datos1
Especifica el tipo de datos del parámetro.

Con una función escalar con origen, se puede utilizar cualquier tipo de datos SQL válido, siempre que se pueda convertir al tipo del parámetro correspondiente de la función identificada en la cláusula SOURCE. Un tipo de datos REF(*nombre-tipo*) no puede especificarse como tipo de datos de un parámetro (SQLSTATE 42997).

Puesto que la función es con origen, no es necesario (pero se permite) especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Pueden utilizarse paréntesis vacíos en su lugar; por ejemplo, CHAR(). Un *tipo de datos con parámetros* es cualquiera de los tipos de datos que se pueden definir con una longitud, escala o precisión específicas. Los tipos de datos con parámetros son los tipos de datos de serie de caracteres y los tipos de datos decimales.

También se pueden utilizar paréntesis vacíos con una plantilla de función en lugar de especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Se recomienda utilizar paréntesis vacíos para los tipos de datos con parámetros. Si utiliza paréntesis vacíos, la longitud, la precisión o la escala son iguales que las de la función remota, que se determina cuando se correlaciona la plantilla de función con una función remota mediante la creación de una correlación de función. Si se omiten los paréntesis, se utilizará la longitud por omisión para el tipo de datos (consulte la descripción de la sentencia CREATE TABLE).

RETURNS
Esta cláusula obligatoria identifica el resultado de la función o plantilla de función.

tipo-datos2
Especifica el tipo de datos de la salida.

Con una función escalar con origen, se acepta cualquier tipo de datos SQL válido, como un tipo diferenciado, siempre que se pueda convertir desde el tipo del resultado de la función de origen.

El parámetro de un tipo con parámetros no es necesario especificarlo, tal como se describió anteriormente para los parámetros de una función con origen. En su lugar, pueden utilizarse paréntesis vacíos; por ejemplo, VARCHAR().

Para obtener información acerca de las consideraciones y normas adicionales que se aplican a la especificación del tipo de datos en la cláusula RETURNS cuando la función tiene su origen en otra función, consulte el apartado "Normas" de esta sentencia.

Con una plantilla de función, los paréntesis vacíos no están permitidos (SQLSTATE 42611). Se debe especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Se recomienda especificar la misma longitud, precisión o escala que las de la función remota.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre, incluido el calificador implícito o explícito, no debe identificar otra instancia de función que exista en el servidor de aplicaciones; de lo contrario, se devuelve un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, debe ser el mismo calificador explícito o implícito del *nombre-función* o se devuelve un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

SOURCE
Especifica que la función que se esté creando debe implantarse por otra función (la función fuente), que el gestor de bases de datos ya conoce. La función de origen puede ser una función incorporada (excepto para COALESCE, DBPARTITIONNUM, NULLIF, HASHEDVALUE, TYPE_ID, TYPE_NAME, TYPE_SCHEMA o VALUE) o una función escalar definida por el usuario creada anteriormente.

Sólo puede especificarse la cláusula SOURCE para funciones escalares o de columna; no se puede especificar para funciones de tabla.

La cláusula SOURCE proporciona la identidad de la otra función.

nombre-función
Identifica la función en particular que va a utilizarse como el origen y sólo es válida si existe exactamente una función específica en el esquema con este *nombre-función* para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, para localizar la función se utiliza la vía de acceso de SQL actual (el valor del registro especial CURRENT PATH). Se selecciona el primer esquema de la vía de acceso de función que tiene una función con este nombre y para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función con este nombre en el esquema nombrado o si el esquema no está calificado y no hay ninguna función con este nombre en la vía de acceso de la función, se devuelve un error (SQLSTATE 42704). Si hay más de una instancia específica autorizada de la función en el esquema nombrado o localizado, se devuelve un error (SQLSTATE 42725). Si existe una función con este nombre y el ID de autorización de la sentencia no tiene el privilegio EXECUTE en esta función, se devuelve un error (SQLSTATE 42501).

SPECIFIC *nombre-específico*
Identifica una función determinada definida por el usuario que debe servir como fuente, por el *nombre-específico* ya sea especificado o tomado por omisión en el tiempo de creación de la función. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, se utiliza la vía de acceso SQL actual para localizar la función. Se selecciona el primer esquema de la vía de acceso de función que tiene una función con este nombre específico para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función para este *nombre-específico* en el esquema nombrado o si el nombre no está calificado y no hay ninguna función con este *nombre-específico* en la vía de acceso SQL, se devuelve un error (SQLSTATE 42704). Si existe una función con este *nombre-específico* y el ID de autorización de la sentencia no dispone de privilegio EXECUTE para esta función, se devuelve un error (SQLSTATE 42501).

nombre-función (tipo-datos,...)
Proporciona la signatura de la función, que identifica de manera exclusiva la función fuente. Esta es la única variante de sintaxis válida para una función fuente que es una función incorporada.

Las normas para la resolución de funciones se aplicarán con el fin de seleccionar una función entre las funciones que tienen el mismo nombre de función, según los tipos de datos que se han especificado en la cláusula SOURCE. Sin embargo, el tipo de datos de cada parámetro de la función seleccionada debe tener exactamente el tipo de datos correspondiente que se ha especificado en la función fuente.

nombre-función
Constituye el nombre de la función fuente. Si se proporciona un nombre no calificado, entonces se tienen en cuenta los esquemas de la vía de acceso de SQL del usuario.

tipo-datos
Debe coincidir con el tipo de datos que se haya especificado en la sentencia CREATE FUNCTION en la posición correspondiente (separado por comas).

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos.

Por ejemplo, DECIMAL() coincidirá con un parámetro cuyo tipo de datos esté definido como DECIMAL(7,2)).

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

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION. Puede ser útil para asegurarse de que se va a utilizar la función que desea. Tenga en cuenta que los sinónimos de los tipos de datos también se considerarán una coincidencia (por ejemplo, DEC y NUMERIC coincidirán).

No es necesario que un tipo FLOAT(n) coincida con el valor definido para n, porque 0<n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ninguna función con la firma especificada en el esquema nombrado o implícito, se devuelve un error (SQLSTATE 42883).

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

La cláusula PARAMETER CCSID debe especificar el mismo esquema de codificación que la función de origen (SQLSTATE 53090).

AS TEMPLATE
Indica que esta sentencia se utilizará para crear una plantilla de función y no una función con código ejecutable.

NOT DETERMINISTIC o **DETERMINISTIC**
Especifica si la función devuelve los mismos resultados para argumentos de entrada idénticos. El valor por omisión es NOT DETERMINISTIC.

NOT DETERMINISTIC
Especifica que es posible que la función no devuelva el mismo resultado cada vez que la función se invoque con los mismos argumentos de entrada. La función depende de algunos valores de estado que afectan a los resultados. El gestor de bases de datos utiliza

esta información durante la optimización de las sentencias de SQL. Un ejemplo de una función que no es determinante es la que genera números aleatorios.

Una función que no es determinante puede recibir resultados incorrectos si se ejecuta por tareas paralelas.

DETERMINISTIC
Especifica que la función siempre devuelve el mismo resultado cada vez que se invoca con los mismos argumentos de entrada. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL. Un ejemplo de función determinante es la que calcula la raíz cuadrada del argumento de entrada.

El cuerpo de la rutina de SQL debe ser coherente con la especificación implícita o explícita de DETERMINISTIC o NOT DETERMINISTIC. Una función que se ha definido como DETERMINISTIC no debe invocar otra función que se haya definido como NOT DETERMINISTIC, ni puede hacer referencia a un registro especial (SQLSTATE 428C2). Por ejemplo, una función de SQL que invoca una función incorporada RAND en su sentencia RETURN se debe haber creado como función no determinante.

EXTERNAL ACTION o **NO EXTERNAL ACTION**
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION
Especifica que la función puede provocar un cambio en el estado de un objeto que el gestor de bases de datos no gestiona.

Una función con acciones externas puede recibir resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si una función envía una nota para cada llamada inicial a esa función, se envía una nota para cada tarea paralela en lugar de una nota para la función.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

EXTERNAL ACTION se debe especificar implícita o explícitamente si el cuerpo de la rutina de SQL invoca una función que está definida con EXTERNAL ACTION (SQLSTATE 428C2).

Normas:

- En esta sección llamaremos CF a la función que se está creando y FS a la función identificada en la cláusula SOURCE, no importa cual de las tres sintaxis permitidas se haya utilizado para identificar FS.
  - El nombre no calificado de la función CF y el nombre no calificado de la SF pueden ser distintos.
  - Una función nombrada como la fuente de otra función puede, a su vez, utilizar otra función como su fuente. Se deben extremar las precauciones al utilizar este recurso ya que podría ser muy difícil depurar una aplicación si una función invocada indirectamente devuelve un error.

  - Si se especifican con la cláusula SOURCE, ninguna de las cláusulas siguientes es válida (porque la CF heredará estos atributos de la SF):
    - CAST FROM ...,
    - EXTERNAL ...,
    - LANGUAGE ...,
    - PARAMETER STYLE ...,
    - DETERMINISTIC / NOT DETERMINISTIC,
    - FENCED / NOT FENCED,
    - RETURNS NULL ON NULL INPUT / CALLED ON NULL INPUT
    - EXTERNAL ACTION / NO EXTERNAL ACTION
    - NO SQL / CONTAINS SQL / READS SQL DATA
    - SCRATCHPAD / NO SCRATCHPAD
    - FINAL CALL / NO FINAL CALL
    - RETURNS TABLE (...)
    - CARDINALITY ...
    - ALLOW PARALLEL / DISALLOW PARALLEL
    - DBINFO / NO DBINFO
    - THREADSAFE / NOT THREADSAFE
    - INHERIT SPECIAL REGISTERS

  Si se incumplen estas normas se produce un error (SQLSTATE 42613).
- El número de parámetros de entrada en CF debe ser igual a los de SF; de lo contrario, se devuelve un error (SQLSTATE 42624).
- La CF no tendrá que especificar necesariamente la longitud, precisión ni escala para un tipo de datos con parámetros en los casos siguientes:
  - Los parámetros de entrada de la función.
  - Su parámetro RETURNS

  En su lugar, se pueden especificar parámetros vacíos como parte del tipo de datos (por ejemplo: VARCHAR()), para mostrar que la longitud/precisión/escala serán las mismas que las de la función fuente o las determinadas por la conversión del tipo de datos.

  No obstante, si se especifica la longitud, la precisión o la escala, el valor de la CF se comprueba comparándolo con el valor correspondiente de la SF, tal y como se explica más abajo para los parámetros de entrada, y se devuelve un valor.
- La especificación de los parámetros de entrada de la CF se comparan con los de la SF. El tipo de datos de cada parámetro de CF debe ser el mismo que el tipo de datos del parámetro correspondiente de la SF o debe ser *convertible* a él. Si algún parámetro no es del mismo tipo o no se puede convertir, se devuelve un error (SQLSTATE 42879).

  Tenga en cuenta que esta norma no proporciona ninguna garantía frente a los errores producidos al utilizar la CF. Un argumento que coincide con el tipo de datos y los atributos de longitud o de precisión de un parámetro de la CF, tal vez no sea asignable si el parámetro correspondiente de la SF tiene una longitud inferior o una precisión menor. Generalmente, los parámetros de la CF no deben tener atributos de longitud o de precisión superiores a los de los parámetros correspondientes de la SF.
- Las especificaciones del tipo de datos RETURNS de la CF se comparan con las de la SF. El tipo de datos final RETURNS de la SF, tras la conversión que sea, debe ser el mismo o bien ser convertible al tipo de datos RETURNS de la CF. De lo contrario, se devuelve un error (SQLSTATE 42866).

Tenga en cuenta que esta norma no proporciona ninguna garantía frente a los errores producidos al utilizar la CF. Un valor de resultado que coincide con el tipo de datos y los atributos de longitud y de precisión del tipo de datos RETURNS de la SF, tal vez no sea asignable si el tipo de datos RETURNS de la CF tiene una longitud inferior o una precisión menor. Debe tenerse precaución al especificar el tipo de datos RETURNS de la CF con atributos de precisión o longitud inferiores a los atributos del tipo de datos RETURNS de la SF.

Notas:

- La determinación de si un tipo de datos es convertible a otro no tiene en cuenta la longitud, precisión ni escala de los tipos de datos con parámetros, como son CHAR y DECIMAL. Por esta razón, pueden producirse errores al utilizar una función como resultado del intento de conversión de un valor del tipo de datos fuente a un valor del tipo de datos de destino. Por ejemplo, VARCHAR es convertible a DATE, pero si el tipo de fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.
- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, considere las normas de promoción que afectarán a sus valores de entrada (consulte el apartado “Promoción de tipos de datos”). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - INTEGER en lugar de SMALLINT
  - DOUBLE en lugar de REAL
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Para que un servidor federado reconozca una función fuente de datos, la función debe estar correlacionada con una función complementaria en la base de datos federada. Si la base de datos no contiene ninguna función complementaria, el usuario debe crearla y luego definir la correlación.

  La función complementaria puede ser una función (escalar o fuente) o una plantilla de función. Si el usuario crea una función y la correlación necesaria, entonces cada vez que se procese una consulta que especifique la función, DB2 (1) compara las estrategias para invocarla con estrategias para invocar la función fuente de datos e (2) invoca la función que se prevé que exigirá menos recursos adicionales.

  Si el usuario crea una plantilla de función y la correlación, cada vez que se procese una consulta que especifique la plantilla, DB2 invocará la función de origen de datos con la que ésta se correlaciona, siempre que exista un plan de acceso para la invocación de esta función.
- ***Privilegios***

  El usuario que define una función siempre recibe el privilegio EXECUTE para la función, así como el derecho para poder eliminar la función. El usuario que define la función también recibe WITH GRANT OPTION si alguna de las condiciones siguientes es verdadera:
  - La función de origen es una función incorporada.

- El usuario que define la función dispone de EXECUTE WITH GRANT OPTION para la función de origen.
- La función es una plantilla.

Ejemplos:

Ejemplo 1: Algún tiempo después de la creación de la función escalar externa CENTRE de Pellow, otro usuario desea crear una función basada en ella, excepto que esta función está pensada para aceptar solamente argumentos enteros.

```
CREATE FUNCTION MYCENTRE (INTEGER, INTEGER)
  RETURNS FLOAT
  SOURCE PELLOW.CENTRE (INTEGER, FLOAT)
```

Ejemplo 2: Se ha creado un tipo diferenciado, HATSIZE, basándose en el tipo de datos INTEGER incorporado. Sería útil tener una función AVG para calcular el tamaño medio (hatsize) de los distintos departamentos. Esto se lleva a cabo fácilmente de la manera siguiente:

```
CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE
  SOURCE SYSIBM.AVG (INTEGER)
```

La creación del tipo diferenciado ha generado la función de conversión necesaria, permitiendo la conversión de HATSIZE en INTEGER para el argumento y de INTEGER a HATSIZE para el resultado de la función.

Ejemplo 3: En un sistema federado, un usuario desea invocar una UDF Oracle que devuelva estadísticas de tabla en forma de valores con comas flotantes de precisión doble. El servidor federado sólo puede reconocer esta función si existe una correlación entre la función y una función complementaria situada en una base de datos federada. Pero esta función complementaria no existe. El usuario decide suministrar una función complementaria en forma de plantilla de función y asignar esta plantilla a un esquema llamado NOVA. El usuario utiliza el código siguiente para registrar la plantilla en el servidor federado.

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)
  RETURNS DOUBLE
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

Ejemplo 4: En un sistema federado, un usuario desea invocar una UDF de Oracle que devuelve los importes en dólares que los empleados de una organización determinada ganan en forma de bonos. El servidor federado sólo puede reconocer esta función si existe una correlación entre la función y una función complementaria situada en una base de datos federada. No existe esta función complementaria; por lo tanto, el usuario crea una en forma de plantilla de función. El usuario usa el código siguiente para registrar esta plantilla en el servidor federado.

```
CREATE FUNCTION BONUS ()
  RETURNS DECIMAL (8,2)
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

Información relacionada:

- "Funciones" en la publicación *Consulta de SQL, Volumen 1*
- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE FUNCTION MAPPING" en la página 268
- "Promoción de tipos de datos" en la publicación *Consulta de SQL, Volumen 1*
- "Conversiones entre tipos de datos" en la publicación *Consulta de SQL, Volumen 1*

CREATE FUNCTION (escalar de SQL, tabla o fila)

Esta sentencia se utiliza para definir una función SQL escalar, de tabla o de fila, definida por el usuario. Una *función escalar* devuelve un solo valor cada vez que se invoca y en general es válida cuando una expresión SQL es válida. Se puede utilizar una *función de tabla* en una cláusula FROM y devuelve una tabla. Se puede utilizar una *función de fila* como función de transformación y devuelve una fila.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Para cada tabla, vista o apodo identificado en cualquier selección completa:
  - Privilegio CONTROL para esa tabla, vista o apodo o
  - Privilegio SELECT para esa tabla, vista o apodo

  y como mínimo uno de los elementos siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la función no existe.
  - Privilegio CREATEIN en el esquema, si el nombre de esquema de la función hace referencia a un esquema existente

Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE FUNCTION.

Los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo se aplican al invocarse la función. El ID de autorización de la conexión se puede correlacionar con un ID de autorización remoto diferente.

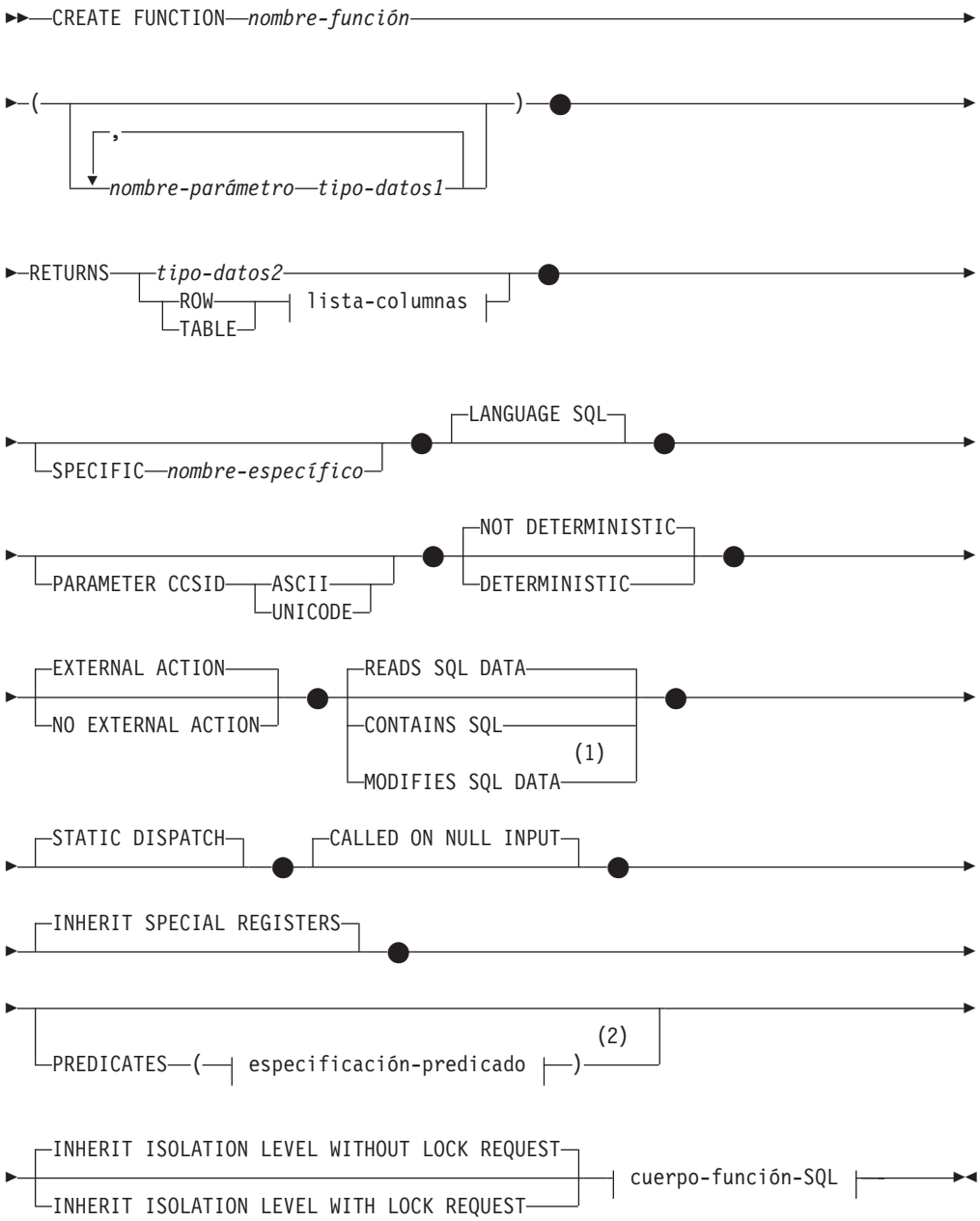
Si un usuario que define una función sólo puede crear la función porque dicho usuario dispone de autorización SYSADM, se le otorga autorización DBADM implícita para poder crear la función.

Si el ID de autorización de la sentencia no tiene la autorización SYSADM o DBADM, los privilegios que el ID de autorización de la sentencia tiene deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL que se especifican en el cuerpo de la función.

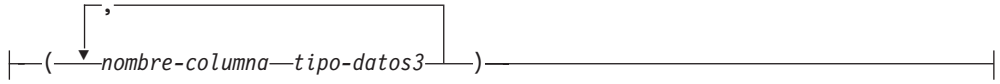
Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:

CREATE FUNCTION (escalar de SQL, tabla o fila)



lista-columnas:



cuerpo-función-SQL:

Notas:

1 Sólo es válido si RETURNS especifica una tabla (TABLE *lista-columnas*)

2 Sólo es válido si RETURNS especifica un resultado escalar (*tipo-datos2*)

Descripción:

nombre-función
Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada del *nombre-función* es un identificador SQL (cuya longitud máxima es de 18). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

nombre-parámetro
Especifica un nombre que es diferente de los nombres de todos los demás parámetros de la función.

tipo-datos1
Especifica el tipo de datos del parámetro:

- Pueden proporcionarse especificaciones y abreviaturas de tipo de datos SQL que puedan especificarse en la definición de *tipo-datos1* de la sentencia CREATE TABLE.
- Se puede especificar REF, pero no tiene un ámbito definido. El sistema no intenta deducir el ámbito del parámetro o resultado. Dentro del cuerpo de la función, se puede utilizar un tipo de referencia en una operación de eliminación de referencia sólo si primero se convierte para que tenga un ámbito. Similarmente, una referencia devuelta por una función SQL se puede utilizar en una operación de eliminación de referencia sólo si primero se convierte para que tenga un ámbito.
- Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC no se pueden utilizar (SQLSTATE 42815).

RETURNS
Esta cláusula obligatoria identifica el tipo de salida de la función.

tipo-datos2
Especifica el tipo de datos de la salida.

En esta sentencia, son válidas exactamente las mismas consideraciones que se describieron anteriormente en *tipo-datos1* para parámetros de funciones SQL.

ROW *lista-columnas*
Especifica que la salida de la función es una fila individual. Si la función devuelve más de una fila, se produce un error (SQLSTATE 21505). La *lista-columnas* debe incluir dos columnas como mínimo (SQLSTATE 428F0).

Una función de fila sólo se puede utilizar como función de transformación para un tipo estructurado (utilizando un tipo estructurado como parámetro y devolviendo sólo tipos base).

TABLE *lista-columnas*
Especifica que el resultado de la función es una tabla.

lista-columnas
Es la lista de nombres de columnas y tipos de datos devueltos para una función de fila o de tabla.

nombre-columna
Especifica el nombre de esta columna. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la fila.

tipo-datos3
Especifica el tipo de datos de la columna y puede ser cualquier tipo de datos soportado por un parámetro de la función SQL.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al eliminar la función o bien al comentarla. Nunca puede emplearse para invocar a la función. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe designar otra instancia de función que exista en el servidor de aplicaciones; de lo contrario se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-función* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-función*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-función*; de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

LANGUAGE SQL
Especifica que la función está escrita en SQL.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula

PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE
Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

Se debe especificar NOT DETERMINISTIC, de forma explícita o implícita, si el cuerpo de la función accede a un registro especial, o invoca otra función no determinista (SQLSTATE 428C2).

NO EXTERNAL ACTION o **EXTERNAL ACTION**
Esta cláusula opcional especifica si la función emprende o no alguna acción que cambie el estado de un objeto no gestionado por el gestor de bases de datos. Si se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones basadas en que la función no produce ningún efecto externo.

Se debe especificar EXTERNAL ACTION, de forma explícita o implícita, si el cuerpo de la función invoca otra función que tiene una acción externa (SQLSTATE 428C2).

CONTAINS SQL, READS SQL DATA, o **MODIFIES SQL DATA**
Indica qué tipo de sentencias de SQL se pueden ejecutar.

CONTAINS SQL
Indica que la función puede ejecutar sentencias de SQL que no lean ni modifiquen datos SQL (SQLSTATE 42985).

READS SQL DATA
Indica que la función puede ejecutar sentencias de SQL que no modifiquen datos SQL (SQLSTATE 42985).

MODIFIES SQL DATA
Indica que la función puede ejecutar todas las sentencias de SQL soportadas en *sentencia-compuesta-dinámica*.

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

CALLED ON NULL INPUT
Esta cláusula indica que se invoca la función con independencia de si cualquiera de sus argumentos es nulo. Puede devolver un valor nulo o un

valor no nulo. En este caso, la función definida por el usuario debe comprobar si los valores de los argumentos son nulos.

Puede especificarse NULL CALL en lugar de CALLED ON NULL INPUT.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional indica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no de la definición del objeto).

Al proceso que llama a la función no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Algunos registros especiales como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, nunca heredan valores del proceso que realiza la llamada.

PREDICATES
Para los predicados que hacen uso de esta función, esta cláusula indica quiénes pueden explotar las extensiones de índice y pueden utilizar la cláusula opcional SELECTIVITY para la condición de búsqueda del predicado. Si se especifica la cláusula PREDICATES, la función debe definirse como DETERMINISTIC con NO EXTERNAL ACTION (SQLSTATE 42613). Si se especifica la cláusula PREDICATES y la base de datos no es Unicode, no se debe especificar PARAMETER CCSID UNICODE (SQLSTATE 42613).

especificación-predicado
Para obtener información detallada acerca de la especificación del predicado, consulte “CREATE FUNCTION (escalar externa)”.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST o **INHERIT ISOLATION LEVEL WITH LOCK REQUEST**
Especifica si una petición de bloqueo puede asociarse o no a la cláusula de aislamiento de la sentencia cuando la función hereda el nivel de aislamiento de la sentencia que invoca la función. El valor por omisión es INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST
Especifica que, como la función hereda el nivel de aislamiento de la sentencia que invoca, no se puede invocar en el contexto de una sentencia de SQL que incluya una cláusula de petición de bloqueo como parte de la cláusula de aislamiento especificada (SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST
Especifica que, como la función hereda el nivel de aislamiento de la sentencia que invoca, también hereda la cláusula de petición de bloqueo especificada.

cuerpo-función-SQL
Especifica el cuerpo de la función. En el cuerpo-función-SQL se puede hacer referencia a nombres de parámetros. Los nombres de parámetros pueden calificarse con el nombre de función para evitar referencias ambiguas.

Si el cuerpo-función-SQL es una sentencia compuesta dinámica, deberá contener, como mínimo, una sentencia RETURN y deberá ejecutarse una sentencia RETURN cuando se llame a la función (SQLSTATE 42632). Si la

función es una función de tabla o fila, sólo podrá contener una sentencia RETURN, que deberá ser la última sentencia de la sentencia compuesta dinámica (SQLSTATE 429BD).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
- La resolución de las llamadas de función dentro del cuerpo de la función se realiza de acuerdo con la vía de función que está vigente para la sentencia CREATE FUNCTION y no cambia una vez creada la función.
- Si una función SQL contiene varias referencias a cualquiera de los registros especiales de fecha u hora, todas las referencias devuelven el mismo valor, y será el mismo valor devuelto por la invocación de registro en la sentencia donde se invocó la función.
- El cuerpo de una función SQL no puede contener una llamada recursiva a sí misma o a otra función o método que la llame, dado que una función de este tipo no puede existir para llamarla.
- Todas las sentencias que crean funciones o métodos aplican las normas siguientes:
  - Una función no puede tener la misma signatura que un método (cuando se compara el primer *tipo-parámetro* de la función con el *tipo-sujeto* del método).
  - Una función y un método no pueden estar en una relación de alteración temporalmente. Esto significa que si la función fuera un método con su primer parámetro como sujeto, no debe alterar a ni ser alterado temporalmente por otro método. Para obtener más información acerca de los métodos de alteración temporal, consulte la sentencia “CREATE TYPE (Estructurado)”.
  - Puesto que la alteración temporal no se aplica a las funciones, pueden existir dos funciones de tal forma que, si fueran métodos, una alteraría temporalmente a la otra.

  Por lo que respecta a la comparación de tipos de parámetros en las normas anteriores:
  - Los nombres de parámetros, longitudes, AS LOCATOR y FOR BIT DATA no se tienen en cuenta.
  - Un subtipo se considera que es diferente que su supertipo.
- ***Restricciones de acceso a las tablas***

  Si se define una función como READS SQL DATA, ninguna sentencia de la función puede acceder a una tabla que se esté modificando por la sentencia que ha invocado la función (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.

Si una función definida con MODIFIES SQL DATA contiene sentencias CALL anidadas, no se permite el acceso de lectura a las tablas que la función está modificando (por la definición de función o la sentencia que ha invocado la función) (SQLSTATE 57053).

- ***Privilegios***

  El usuario que define una función siempre recibe el privilegio EXECUTE para la función, así como el derecho para poder eliminar la función. El usuario que define la función también recibe WITH GRANT OPTION para la función si dicho usuario dispone de WITH GRANT OPTION para todos los privilegios necesarios para definir la función o si el usuario que define la función dispone de autorización SYSADM o DBADM.

  El usuario que define una función sólo adquiere privilegios si, en el momento de crearse la función, existen los privilegios de los que estos privilegios se obtienen. El usuario que define el método debe disponer de estos privilegios directamente o bien porque PUBLIC tiene los privilegios. Los privilegios que tienen los grupos de los que es miembro el usuario que define la función no se consideran. Cuando se utiliza la función, el ID de autorización del usuario conectado debe disponer de los privilegios válidos para la tabla o vista a la que hace referencia el apodo en la fuente de datos.

Ejemplos:

Ejemplo 1: Definición de una función escalar que devuelve la tangente de un valor utilizando las funciones de seno y coseno existentes.

```
CREATE FUNCTION TAN (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

Ejemplo 2: Definición de una función de transformación para el tipo estructurado PERSON.

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

Ejemplo 3: Definición de una tabla de función que muestra los empleados que trabajan en un número de departamento especificado.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
    SELECT EMPNO, LASTNAME, FIRSTNME
      FROM EMPLOYEE
      WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

Ejemplo 4: Defina una función escalar que invierta una serie de caracteres.

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(4000))
  RETURNS VARCHAR(4000)
  DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
  BEGIN ATOMIC
  DECLARE REVSTR, RESTSTR VARCHAR(4000) DEFAULT '';
  DECLARE LEN INT;
  IF INSTR IS NULL THEN
  RETURN NULL;
  END IF;
  SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
  WHILE LEN > 0 DO
  SET (REVSTR, RESTSTR, LEN)
    = (SUBSTR(RESTSTR, 1, 1) || REVSTR,
    SUBSTR(RESTSTR, 2, LEN - 1),
    LEN - 1);
  END WHILE;
  RETURN REVSTR;
END
```

Ejemplo 4: Defina la función de tabla del Ejemplo 4 con auditoría.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  MODIFIES SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  BEGIN ATOMIC
    INSERT INTO AUDIT
    VALUES (USER,
            'Table: EMPLOYEE Prd: DEPTNO = ' || DEPTNO);
    RETURN
      SELECT EMPNO, LASTNAME, FIRSTNME
        FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
  END
```

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE TYPE (Estructurado)" en la página 433
- "RETURN" en la página 656
- "SQL compuesto (Dinámico)" en la página 130
- "CREATE FUNCTION (Escalar externa)" en la página 198
- "Sentencias de SQL que se permiten en rutinas" en la publicación *Consulta de SQL, Volumen 1*
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*

CREATE FUNCTION MAPPING

La sentencia CREATE FUNCTION MAPPING se utiliza para:

- Crear una correlación entre una función o una plantilla de función, situadas en una base de datos federada, y una función fuente de datos. La correlación puede asociar la función o la plantilla de base de datos federada con una función de:
  - Una fuente de datos especificada
  - Un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular
- Inhabilitar una correlación por omisión entre una función de base de datos federada y una función de fuente de datos.

Si se pueden aplicar múltiples correlaciones de función a una función, se aplica la más reciente.

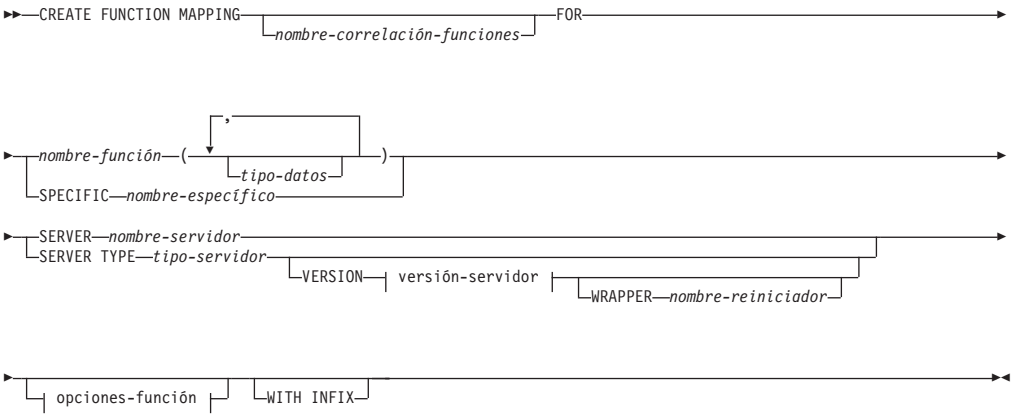
Invocación:

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

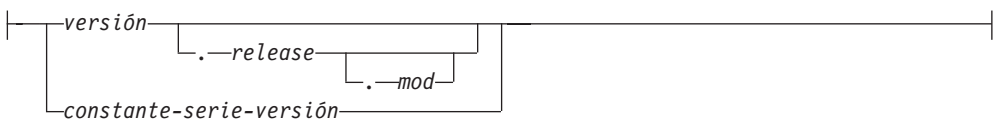
Autorización:

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

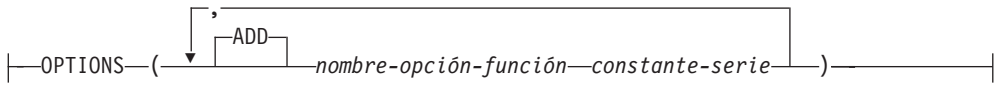
Sintaxis:



versión-servidor:



opciones-función:



Descripción:

nombre-correlación-funciones
Nombra la correlación de funciones. El nombre no debe identificar ninguna correlación de funciones que ya esté descrita en el catálogo (SQLSTATE 42710).

Si se omite *nombre-correlación-funciones*, se asigna un nombre exclusivo generado por el sistema.

nombre-función
Especifica el nombre calificado o no calificado de la función o de la plantilla de función de base de datos federada desde la cual se debe realizar la correlación.

tipo-datos
Para una función o una plantilla de función que tiene parámetros de entrada, *tipo-datos* especifica el tipo de datos de cada parámetro. El *tipo-datos* no puede ser LONG VARCHAR, LONG VARGRAPHIC, DATALINK o un tipo definido por el usuario.

Se pueden utilizar paréntesis vacíos en lugar de especificar la longitud, la precisión o la escala para tipos de datos con parámetros. Se recomienda utilizar los paréntesis vacíos para tipos de datos con parámetros; por ejemplo, CHAR(). Un tipo de datos con parámetros es cualquiera de los tipos de datos que se puede definir con una longitud, una escala o una precisión específica. Los tipos de datos con parámetros son los tipos de datos de serie y los tipos de datos decimales. Si especifica la longitud, la precisión o la escala, deben ser las mismas que las de la plantilla de función. Si omite los paréntesis, se utilizará la longitud por omisión para el tipo de datos (consulte la descripción de la sentencia CREATE TABLE).

SPECIFIC *nombre-específico*
Identifica la función o la plantilla de función desde la cual se debe realizar la correlación. Especifique *nombre-específico* si la función o plantilla de función no tiene un *nombre-función* exclusivo en la base de datos federada.

SERVER *nombre-servidor*
Nombra la fuente de datos que contiene la función que se está correlacionando.

TYPE *tipo-servidor*
Identifica el tipo de fuente de datos que contiene la función que se está correlacionando.

VERSION
Identifica la versión de la fuente de datos indicada por *tipo-servidor*.

versión
Especifica el número de versión. El valor debe ser un entero.

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

mod
Especifica el número de la modificación del release indicado por *release*. El valor debe ser un entero.

constante-serie-versión
Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i') o puede estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*
Especifica el nombre del reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*.

OPTIONS
Indica las opciones de correlación de funciones que se deben habilitar.

ADD
Habilita una o varias opciones de correlación de funciones.

nombre-opción-función
Nombra una opción de correlación de funciones que se aplica a la correlación de funciones o a la función de fuente de datos incluida en la correlación.

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

WITH INFIX
Especifica que la función de fuente de datos se debe generar en formato infijo. El sistema de base de datos federado convierte la notación de prefijo en la notación infijo que utiliza la fuente de datos remota.

Notas:

- Una función o plantilla de función de base de datos federada puede correlacionarse con una función de fuente de datos si:
  - La función o plantilla de la base de datos federada tiene el mismo número de parámetros de entrada que la función de fuente de datos.
  - Los tipos de datos que se definen para la función o la plantilla federadas son compatibles con los tipos de datos correspondientes definidos para la función de la fuente de datos.
- Si una petición distribuida hace referencia a una función DB2 que se correlaciona con una función de fuente de datos, el optimizador desarrolla estrategias para invocar cualquiera de las dos funciones cuando se procesa la petición. Se invoca la función DB2 si para ello se necesita menos actividad general que para invocar la función de fuente de datos. De lo contrario, si la invocación de la función DB2 requiere más actividad general, se invocará a la función de fuente de datos.
- Si una petición distribuida hace referencia a una plantilla de función DB2 que se correlaciona con una función de fuente de datos, sólo se puede invocar la función de fuente de datos cuando se procesa la petición. La plantilla no puede invocarse porque no tiene ningún código ejecutable.
- Las correlaciones de funciones por omisión pueden pasar a ser no operativas inhabilitándolas (no se pueden desactivar). Para inhabilitar la correlación de funciones, codifique la sentencia CREATE FUNCTION MAPPING de modo que especifique el nombre de la función DB2 en la correlación y establezca la opción DISABLE en 'Y'.
- Las funciones del esquema SYSIBM no tienen ningún nombre específico. Para alterar temporalmente la correlación de funciones por omisión para una función

del esquema SYSIBM, especifique *nombre-función* con el calificador SYSIBM y un nombre de función (por ejemplo, LENGTH).

- Una sentencia CREATE FUNCTION MAPPING de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos

Ejemplos:

Ejemplo 1: Correlacione una plantilla de función con una UDF a la que pueden acceder todas las fuentes de datos Oracle. La plantilla se llama STATS y pertenece a un esquema llamado NOVA. La UDF de Oracle se llama STATISTICS y pertenece a un esquema llamado STAR.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
  FOR NOVA.STATS (DOUBLE, DOUBLE)
  SERVER TYPE ORACLE
  OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

Ejemplo 2: Correlacione una plantilla de función llamado BONUS con una UDF, que también se llama BONUS y que se utiliza en una fuente de datos Oracle llamada ORACLE1.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2
  FOR BONUS()
  SERVER ORACLE1
  OPTIONS (REMOTE_NAME 'BONUS')
```

Ejemplo 3: Suponga que existe una correlación de funciones por omisión entre la función del sistema WEEK que está definida en la base de datos federada y una función similar que está definida en las fuentes de datos Oracle. Cuando se procesa una consulta que pide datos Oracle y que hace referencia a WEEK, se invocará WEEK o su equivalente de Oracle, dependiendo de cuál estime el optimizador que necesita menos actividad general. El DBA desea averiguar cómo afectaría al rendimiento si sólo se invocase WEEK para dichas consultas. Para asegurarse de que se invoca WEEK cada vez, el DBA debe inhabilitar la correlación.

```
CREATE FUNCTION MAPPING
  FOR SYSFUN.WEEK(INT)
  TYPE ORACLE
  OPTIONS (DISABLE 'Y')
```

Ejemplo 4: Correlacione la función local UCASE(CHAR) con una UDF que se utiliza en una fuente de datos Oracle denominada ORACLE2. Incluya el número estimado de instrucciones por invocación de la UDF Oracle.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4
  FOR SYSFUN.UCASE(CHAR)
  SERVER ORACLE2
  OPTIONS
    (REMOTE_NAME 'UPPERCASE',
    INSTS_PER_INVOC '1000')
```

Conceptos relacionados:

- "Function mappings in a federated system" en la publicación *Federated Systems Guide*
- "How function mappings work in a federated system" en la publicación *Federated Systems Guide*
- "Function templates" en la publicación *Federated Systems Guide*
- "How to create function mappings" en la publicación *Federated Systems Guide*

Tareas relacionadas:

- "Disabling a default function mapping" en la publicación *Federated Systems Guide*

Información relacionada:

- "Function mapping options for federated systems" en la publicación *Federated Systems Guide*

CREATE INDEX

La sentencia CREATE INDEX se utiliza para:

- Crear un índice en una tabla de DB2
- Crear una especificación de índice (metadatos que indican al optimizador que una tabla de fuente de datos tiene un índice)

Invocación:

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

Autorización:

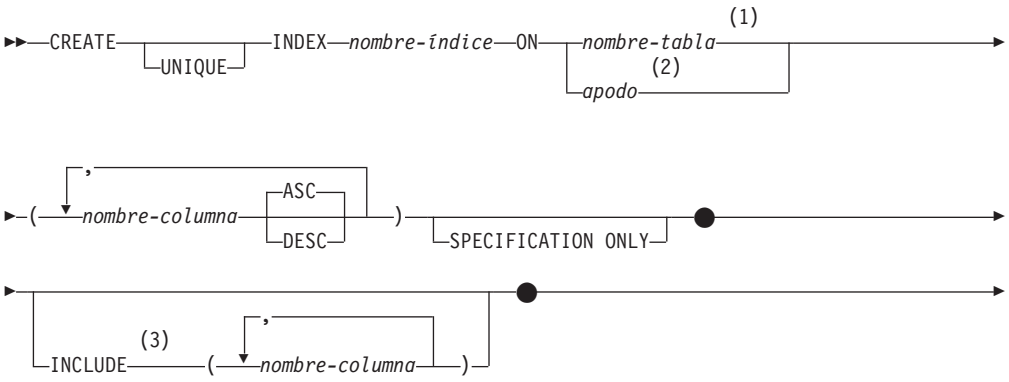
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

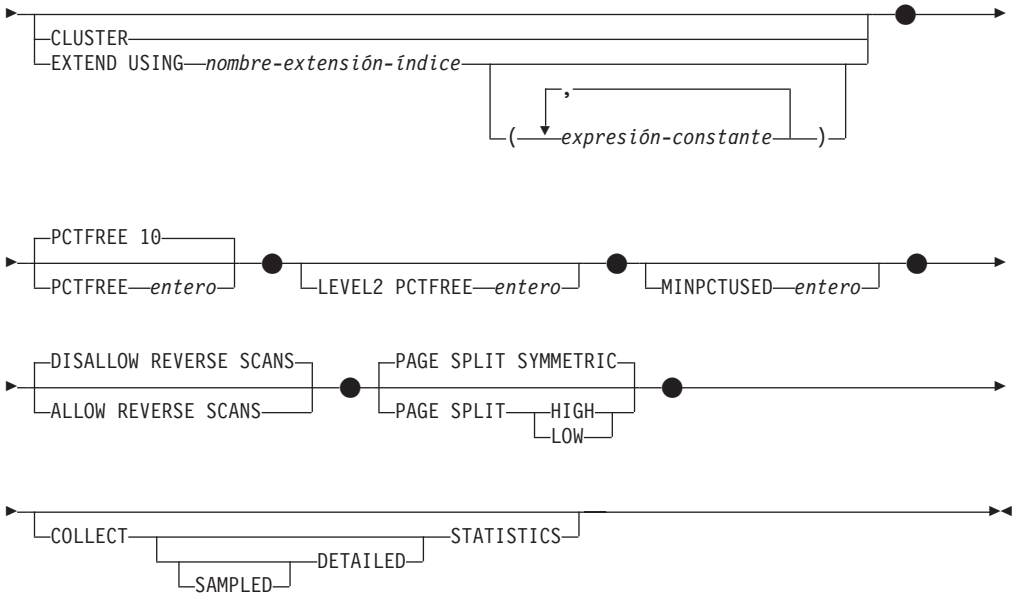
- Autorización SYSADM o DBADM.
- Uno de los siguientes:
  - El privilegio CONTROL en la tabla o el apodo en el que está definido el índice
  - El privilegio INDEX en la tabla o el apodo en el que está definido el índice

  y uno de estos:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del índice no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema del índice hace referencia a un esquema existente.

No se necesita ningún privilegio explícito para crear un índice en una tabla temporal declarada.

Sintaxis:





Notas:

1 En un sistema federado, *nombre-tabla* debe identificar una tabla en la base de datos federada. No puede identificar una tabla de fuente de datos.

2 Si se especifica *apodo*, la sentencia CREATE INDEX crea una especificación de índice. En este caso, no se pueden especificar INCLUDE, CLUSTER, EXTEND USING, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, ALLOW REVERSE SCANS, PAGE SPLIT ni COLLECT STATISTICS.

3 La cláusula INCLUDE sólo se puede especificar si se especifica UNIQUE.

Descripción:

UNIQUE

Si se especifica ON *nombre-tabla*, UNIQUE evita que la tabla contenga dos o más filas con el mismo valor de la clave de índice. La unicidad de valores se aplica al final de la sentencia de SQL que actualiza filas o inserta nuevas filas.

La exclusividad también se comprueba durante la ejecución de la sentencia CREATE INDEX. Si la tabla ya contiene filas con valores de clave duplicados, no se crea el índice.

Cuando se utiliza UNIQUE, los valores nulos se tratan como cualquier otro valor. Por ejemplo, si la clave es una sola columna que puede contener valores nulos, esa columna no puede contener más de un valor nulo.

Si se especifica la opción UNIQUE y la tabla tiene una clave de particionamiento, las columnas de la clave de índice deben ser un superconjunto de la clave de partición. Es decir, las columnas especificadas para una clave de índice de unicidad deben incluir todas las columnas de la clave de particionamiento (SQLSTATE 42997).

Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).

Si se especifica ON *apodo*, sólo se debe especificar UNIQUE si los datos para la clave de índice contienen valores exclusivos para cada fila de la tabla de fuente de datos. No se comprobará la exclusividad.

INDEX *nombre-índice*
Nombra el índice o la especificación de índice. El nombre, (incluido el calificador implícito o explícito) no debe designar un índice o una especificación de índice que se describa en el catálogo o un índice existente de una tabla temporal declarada (SQLSTATE 42704). El calificador no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

El calificador implícito o explícito de los índices de las tablas temporales globales declaradas debe ser SESSION (SQLSTATE 428EK).

ON *nombre-tabla* **o** *apodo*
El *nombre-tabla* identifica una tabla en la que va a crearse un índice. La tabla debe ser una tabla base (no una vista), una tabla de consultas materializadas descrita en el catálogo o una tabla temporal declarada. El nombre de una tabla temporal declarada debe calificarse con SESSION. El *nombre-tabla* no debe identificar una tabla de catálogo (SQLSTATE 42832). Si se especifica UNIQUE y *nombre-tabla* es una tabla con tipo, no debe ser una subtabla (SQLSTATE 429B3).

apodo es el apodo en el que una especificación de índice se creará. El *apodo* hace referencia a la tabla de fuente de datos cuyo índice se describe mediante la especificación de índice o la vista de fuente de datos que se basa en esa tabla. El *apodo* debe aparecer en la lista del catálogo.

nombre-columna
Para un índice, *nombre-columna* identifica una columna que debe formar parte de la clave de índice. Para una especificación de índice, *nombre-columna* es el nombre que el servidor federado utiliza para hacer referencia a una columna de una tabla de fuente de datos.

Cada *nombre-columna* debe ser un nombre no calificado que identifique a una columna de la tabla. Pueden especificarse hasta 16 columnas. Si *nombre-tabla* es una tabla con tipo, pueden especificarse hasta 15 columnas. Si *nombre-tabla* es una subtabla, como mínimo deberá introducirse un *nombre-columna* en la subtabla; es decir, no deberá heredarse de una supertabla (SQLSTATE 428DS). No puede repetirse ningún *nombre-columna* (SQLSTATE 42711).

La suma de los atributos de longitud de las columnas especificadas no debe ser mayor que 1024. Si *nombre-tabla* es una tabla con tipo, la longitud de la clave de índice debe ser todavía 4 bytes menos.

Tenga en cuenta que la actividad general del sistema puede reducir esta longitud, que varía en función del tipo de datos de la columna y en función de si puede contener nulos. Para obtener más información acerca de la forma en que la actividad general puede afectar a este límite, consulte “Cuentas de bytes” en “CREATE TABLE”.

No puede utilizarse ninguna columna LOB, columna DATALINK o columna de tipo diferenciado basada en LOB o DATALINK como parte de un índice, aunque el atributo de longitud de la columna sea lo suficientemente pequeño como para estar comprendido dentro del límite de 1024 bytes (SQLSTATE 54008). Las columnas de tipo estructurado sólo se pueden especificar si también se especifica la cláusula EXTEND USING (SQLSTATE 42962). Si se especifica la cláusula EXTEND USING, sólo puede especificarse una columna, y el tipo de la columna debe ser un tipo estructurado o un tipo diferenciado no basado en LOB, DATALINK, LONG VARCHAR o LONG VARGRAPHIC (SQLSTATE 42997).

ASC
Especifica que las entradas de índice se deben mantener en el orden

ascendente de los valores de columna; este es el valor por omisión. ASC no se puede especificar para índices que están definidos con EXTEND USING (SQLSTATE 42601).

DESC
Especifica que las entradas de índice se deben mantener en el orden descendente de los valores de columna. DESC no se puede especificar para índices que están definidos con EXTEND USING (SQLSTATE 42601).

SPECIFICATION ONLY
Indica que esta sentencia será utilizada para crear una especificación de índice que se suscribe a la tabla de fuente de datos a la que se hace referencia mediante *apodo*. Se debe especificar SPECIFICATION ONLY si se especifica *apodo* (SQLSTATE 42601). No se puede especificar si se especifica *nombre-tabla* (SQLSTATE 42601).

Esta cláusula no puede utilizarse al crear un índice en una tabla temporal declarada (SQLSTATE 42995).

INCLUDE
Esta palabra clave introduce una cláusula que especifica columnas adicionales a añadir al conjunto de columnas de la clave de índice. Las columnas incluidas con esta cláusula no se utilizan para imponer la exclusividad. Estas columnas incluidas pueden mejorar el rendimiento de algunas consultas mediante el acceso de sólo índice. Las columnas deben ser diferentes de las columnas utilizadas para imponer la exclusividad (SQLSTATE 42711). Los límites para el número de columnas y la suma de los atributos de longitud se aplican a todas las columnas del índice y la clave de unicidad.

Esta cláusula no puede utilizarse con tablas temporales declaradas (SQLSTATE 42995).

nombre-columna
Identifica una columna que se incluye en el índice, pero que no forma parte de la clave de índice de unicidad. Se aplican las mismas normas que se han definido para las columnas de la clave de índice de unicidad. Pueden especificarse las palabras clave ASC o DESC después del nombre-columna, pero no tienen efecto sobre el orden.

INCLUDE no se puede especificar para índices que están definidos con EXTEND USING, o si está especificado *apodo* (SQLSTATE 42601).

CLUSTER
Especifica que el índice es el índice de agrupación de la tabla. El factor de agrupación de un índice de agrupación se mantiene o se mejora dinámicamente cuando los datos se insertan en la tabla asociada al intentar insertar filas nuevas físicamente cerca de las filas para las que los valores de clave de este índice están en el mismo rango. Sólo puede existir un índice de agrupación para una tabla, por lo que no puede especificarse CLUSTER si se utiliza en la definición de cualquier índice existente en la tabla (SQLSTATE 55012). No puede crearse un índice de agrupación en una tabla que esté definida para utilizar la modalidad APPEND (SQLSTATE 428D8).

CLUSTER no está permitido si se ha especificado *apodo* (SQLSTATE 42601). Esta cláusula no se puede utilizar con tablas temporales declaradas (SQLSTATE 42995) ni con tablas de clúster de rango SQLSTATE 429BG).

EXTEND USING *nombre-extensión-índice*
Designa la *extensión-índice* utilizada para gestionar el índice. Si se especifica esta cláusula, debe haber un solo *nombre-columna* especificado y esa columna

debe ser un tipo estructurado o un tipo diferenciado (SQLSTATE 42997). El *nombre-extensión-índice* debe ser una extensión de índice descrita en el catálogo (SQLSTATE 42704). Para un tipo diferenciado, la columna debe coincidir exactamente con el tipo del correspondiente parámetro de clave fuente de la extensión de índice. Para una columna de tipo estructurado, el tipo del correspondiente parámetro de clave fuente debe ser el mismo tipo o un supertipo del tipo de columna (SQLSTATE 428E0).

Esta cláusula no puede utilizarse con tablas temporales declaradas (SQLSTATE 42995).

expresión-constante
Designa valores para los argumentos necesarios de la extensión de índice. Cada expresión debe ser un valor constante cuyo tipo de datos coincide exactamente con el tipo de datos definido de los correspondientes parámetros de la extensión de índice, incluidas la longitud o precisión, y la escala (SQLSTATE 428E0). Esta cláusula no debe exceder de 32.768 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

PCTFREE *entero*
Especifica qué porcentaje de cada página de índice se va a dejar como espacio libre cuando se cree el índice. La primera entrada de una página se añade sin restricciones. Cuando se colocan entradas adicionales en una página de índice, en cada página se deja, como mínimo, un *entero* como porcentaje de espacio libre. El valor de *entero* entra en un rango que va de 0 a 99. No obstante, si se especifica un valor superior a 10, sólo se dejará un 10 por ciento de espacio libre en las páginas sin hoja. El valor por omisión es 10.

PCTFREE no está permitido si *apodo* está especificado (SQLSTATE 42601). Esta cláusula no puede utilizarse con tablas temporales declaradas (SQLSTATE 42995).

LEVEL2 PCTFREE *entero*
Especifica qué porcentaje de cada página de nivel 2 de índice se va a dejar como espacio libre cuando se cree el índice. El valor de *entero* entra en un rango que va de 0 a 99. Si LEVEL2 PCTFREE no está establecido, se deja un mínimo de 10 o PCTFREE por ciento de espacio libre en todas las páginas que no son hojas. Si LEVEL2 PCTFREE está establecido, se deja *entero* por ciento de espacio libre en páginas intermedias de nivel 2 y se deja un mínimo de 10 o *entero* por ciento de espacio libre en páginas intermedias de nivel 3 y superiores.

LEVEL2 PCTFREE no está permitido si se especifica *apodo* (SQLSTATE 42601). Esta cláusula no puede utilizarse con tablas temporales declaradas (SQLSTATE 42995).

MINPCTUSED *entero*
Indica si las páginas de hoja de índice deben fusionarse en línea y el umbral del porcentaje mínimo de espacio utilizado en una página de hoja de índice. Si, después de eliminarse una clave de una página de hoja de índice, el porcentaje de espacio utilizado en la página es el porcentaje de *entero* o está por debajo de éste, se intentarán fusionar las claves restantes de esta página con las de una página vecina. Si hay espacio suficiente en una de estas páginas, se realiza la fusión y se elimina una de las páginas. El valor de *entero* puede ir de 0 a 99. Sin embargo, se recomienda un valor de 50 o inferior por motivos de rendimiento. La especificación de esta opción influirá en el rendimiento de la actualización y de la supresión. Para los índices de tipo 2, la fusión sólo tiene lugar durante las operaciones de actualización y supresión cuando existe un bloqueo de tabla exclusivo. Si no existe un bloqueo de tabla exclusivo, las

claves se marcan como claves a las que se ha aplicado una supresión falsa durante las operaciones de actualización y supresión y no se realiza ninguna fusión. Considere la utilización de la opción CLEANUP ONLY ALL de REORG INDEXES para fusionar páginas de hoja en lugar de utilizar la opción MINPCTUSED de CREATE INDEX.

MINPCTUSED no está permitido si *apodo* está especificado (42601). Esta cláusula no puede utilizarse con tablas temporales declaradas (SQLSTATE 42995).

DISALLOW REVERSE SCANS
Especifica que un índice sólo permite búsquedas hacia adelante o en el orden definido al crear el índice. Este es el valor por omisión.

DISALLOW REVERSE SCANS no está permitido si *apodo* está especificado (42601).

ALLOW REVERSE SCANS
Especifica que un índice permite búsquedas hacia delante y hacia atrás; es decir, en el orden definido al crear el índice y en el orden opuesto.

ALLOW REVERSE SCANS no está permitido si *apodo* está especificado (42601).

PAGE SPLIT
Especifica un comportamiento de división de índice. El valor por omisión es SYMMETRIC.

SYMMETRIC
Especifica que las páginas se deben dividir por la mitad.

HIGH
Especifica un comportamiento de división de páginas de índice que utiliza espacio de páginas de índice de forma eficiente cuando los valores de las claves de índice que se insertan siguen un determinado patrón. La clave de índice debe contener más de una columna. Para un subconjunto de valores de claves de índices, la columna o columnas que hay más a la izquierda del índice deben contener el mismo valor y la columna o columnas que hay más a la derecha del índice deben contener valores que aumentan con cada inserción. Para ver detalles, consulte "Opciones de la sentencia CREATE INDEX".

LOW
Especifica un comportamiento de división de páginas de índice que utiliza espacio de páginas de índice de forma eficiente cuando los valores de las claves de índice que se insertan siguen un determinado patrón. La clave de índice debe contener más de una columna. Para un subconjunto de valores de claves de índices, la columna o columnas que hay más a la izquierda del índice deben contener el mismo valor y la columna o columnas que hay más a la derecha del índice deben contener valores que disminuyen con cada inserción. Para ver detalles, consulte "Opciones de la sentencia CREATE INDEX".

COLLECT STATISTICS
Especifica que van a recopilarse estadísticas de índice básicas durante la creación del índice.

DETAILED
Especifica que también van a recopilarse estadísticas de índice ampliadas (CLUSTERFACTOR y PAGE_FETCH_PAIRS) durante la creación del índice.

SAMPLED
Especifica que puede utilizarse el muestreo durante la compilación de estadísticas de índice ampliadas.

Normas:

- La sentencia CREATE INDEX fallará (SQLSTATE 01550) si se intenta crear un índice que coincida con un índice ya existente. Dos descripciones de índice se consideran duplicadas si:
  – el conjunto de columnas (las columnas de la clave y de inclusión) con su orden en el índices el mismo que el de un índice existente Y
  – los atributos de orden son los mismos Y
  – tanto el índice preexistente como el nuevo no son exclusivos O BIEN el índice que existía anteriormente es exclusivo Y
  – si tanto el índice preexistente como el nuevo son exclusivos, las columnas de clave del índice que se crean son las mismas que las de la clave del índice que existía anteriormente o son un superconjunto de éstas.

Notas:

- *Compatibilidades*
  – Para mantener la compatibilidad con DB2 para OS/390:
    - Se tolera y se pasa por alto la sintaxis siguiente:
      • CLOSE
      • DEFINE
      • FREEPAGE
      • GBPCACHE
      • PIECESIZE
      • TYPE 2
      • utilizando-bloque
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      • COPY NO
      • DEFER NO
- Durante la creación de un índice, está permitido el acceso de lectura/grabación concurrente a la tabla. Una vez creado el índice, los cambios que se han realizado en la tabla durante la creación del índice se reenvían e incorporan al nuevo índice. El acceso de grabación a la tabla queda bloqueado brevemente mientras se completa la creación del índice, tras lo cual, el nuevo índice estará disponible.

  Para eludir este comportamiento por omisión, utilice la sentencia LOCK TABLE para bloquear explícitamente la tabla antes de emitir una sentencia CREATE INDEX. (La tabla puede bloquearse en modalidad SHARE o EXCLUSIVE, en función de si va a estar permitido el acceso de lectura.)
- Si la tabla nombrada ya contiene datos, CREATE INDEX crea las entradas de índice de la misma. Si la tabla todavía no contiene datos, CREATE INDEX crea una descripción del índice; las entradas del índice se crean al insertar los datos en la tabla.
- Una vez se ha creado el índice y se han cargado los datos en la tabla, es aconsejable emitir el mandato RUNSTATS. El mandato RUNSTATS actualiza las estadísticas que se reúnen en las tablas de bases de datos, las columnas y los índices. Estas estadísticas sirven para determinar la mejor vía de acceso a las tablas. Mediante la emisión del mandato RUNSTATS, el gestor de bases de datos puede determinar las características del nuevo índice. Si se habían cargado datos

antes de la emisión de la sentencia CREATE INDEX, se recomienda utilizar la opción COLLECT STATISTICS de la sentencia CREATE INDEX como alternativa a la utilización del mandato RUNSTATS.

- La creación de un índice con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- El optimizador puede recomendar índices antes de crear el índice actual.
- Si una especificación de índice se está definiendo para una tabla de fuente de datos, el nombre de la especificación de índice no tiene que coincidir con el nombre del índice.
- El optimizador utiliza las especificaciones de índice para mejorar el acceso a las tablas de fuente de datos a las que las especificaciones se suscriben.
- Las opciones COLLECT STATISTICS no reciben soporte con las tablas temporales declaradas (SQLSTATE 42995).
- Las opciones COLLECT STATISTICS no reciben soporte si se ha especificado un apodo (SQLSTATE 42601).
- Al crear un índice exclusivo en una tabla de consultas materializadas (MQT), tenga en cuenta las implicaciones que esta restricción de exclusividad puede tener para otros procesos. Por ejemplo, si el índice exclusivo no coincide con los atributos de exclusividad de la consulta materializada de renovación inmediata de una MQT mantenida por el sistema, será el índice, y no la MQT, el que obtendrá las violaciones de exclusividad durante las operaciones de inserción o actualización de la tabla subyacente. En un caso similar, con una renovación diferida de una MQT mantenida por el sistema, la sentencia REFRESH TABLE fallará. En general, un índice exclusivo en una MQT debe coincidir con las restricciones de exclusividad que ya existen para los datos basados en la tabla subyacente o que pueden inferirse de la consulta asociada a la MQT.

Ejemplos:

Ejemplo 1: Cree un índice denominado UNIQUE_NAM en la tabla PROJECT. La finalidad de este índice consiste en garantizar que en la misma tabla no habrá dos entradas que tengan el mismo valor para el nombre del proyecto (PROJNAME). Las entradas de índice deben estar en orden ascendente.

```
CREATE UNIQUE INDEX UNIQUE_NAM
   ON PROJECT(PROJNAME)
```

Ejemplo 2: Cree un índice denominado JOB_BY_DPT en la tabla EMPLOYEE. Disponga las entradas de índice en orden ascendente por el título de los trabajos (JOB) dentro de cada departamento.

```
CREATE INDEX JOB_BY_DPT
   ON EMPLOYEE (WORKDEPT, JOB)
```

Ejemplo 3: El apodo EMPLOYEE hace referencia a una tabla de fuente de datos denominada CURRENT_EMP. Tras la creación de este apodo, se ha definido un índice en CURRENT_EMP. Las columnas escogidas para la clave de índice fueron WORKDEBT y JOB. Cree una especificación de índice que describa a este índice. Mediante esta especificación, el optimizador sabrá que el índice existe y cuál es su clave. Con esta información, el optimizador puede mejorar su estrategia de acceso a la tabla.

```
CREATE UNIQUE INDEX JOB_BY_DEPT
ON EMPLOYEE (WORKDEPT, JOB)
   SPECIFICATION ONLY
```

Ejemplo 4: Cree un tipo de índice ampliado denominado SPATIAL_INDEX en una ubicación de columna de tipo estructurado. La descripción de la extensión de índice GRID_EXTENSION se utiliza para mantener SPATIAL_INDEX. El literal se proporciona a GRID_EXTENSION para crear el tamaño de cuadrícula de índice.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
  EXTEND USING (GRID_EXTENSION (x'000100100010001000400010'))
```

Ejemplo 5: Cree un índice denominado IDX1 en una tabla denominada TAB1 y recopile estadísticas de índice básicas en el índice IDX1.

```
CREATE INDEX IDX1 ON TAB1 (col1) COLLECT STATISTICS
```

Ejemplo 6: Cree un índice denominado IDX2 en una tabla denominada TAB1 y recopile estadísticas de índice detalladas en el índice IDX2.

```
CREATE INDEX IDX2 ON TAB1 (col2) COLLECT DETAILED STATISTICS
```

Ejemplo 7: Cree un índice denominado IDX3 en una tabla denominada TAB1 y recopile estadísticas de índice detalladas en el índice IDX3 utilizando el muestreo.

```
CREATE INDEX IDX3 ON TAB1 (col3) COLLECT SAMPLED DETAILED STATISTICS
```

Conceptos relacionados:

- "Options on the CREATE INDEX statement" en la publicación *Administration Guide: Implementation*
- "Index specifications in a federated system" en la publicación *Federated Systems Guide*
- "Index specifications" en la publicación *Federated Systems Guide*

Información relacionada:

- "CREATE TABLE" en la página 341
- "Interacción de los activadores con las restricciones" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE INDEX EXTENSION" en la página 282

Ejemplos relacionados:

- "dbstat.sqb -- Reorganize table and run statistics (MF COBOL)"
- "TbGenCol.java -- How to use generated columns (JDBC)"

CREATE INDEX EXTENSION

La sentencia CREATE INDEX EXTENSION crea un objeto de extensión para utilizar con índices de tablas que tienen columnas de tipo estructurado o de tipo diferenciado.

Invocación:

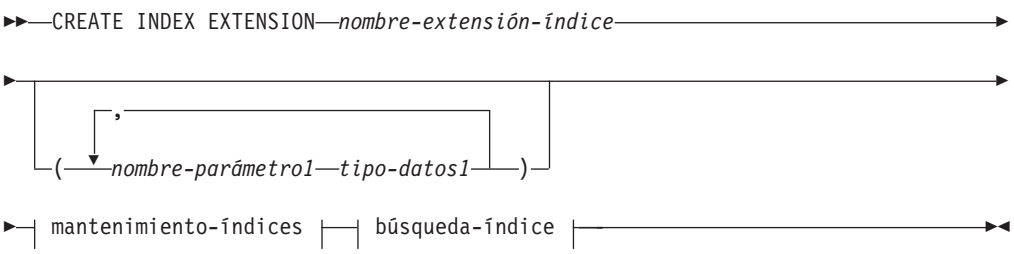
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

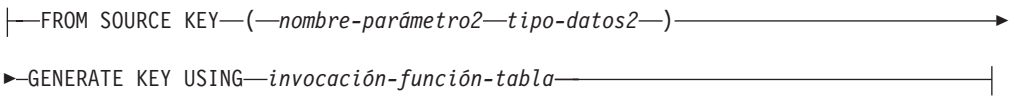
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos (si el nombre de esquema de la extensión de índice no hace referencia a un esquema existente)
- Privilegio CREATEIN para el esquema (si el nombre de esquema de la extensión de índice hace referencia a un esquema existente)

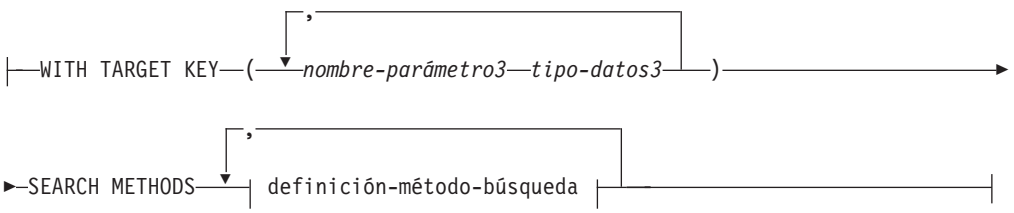
Sintaxis:



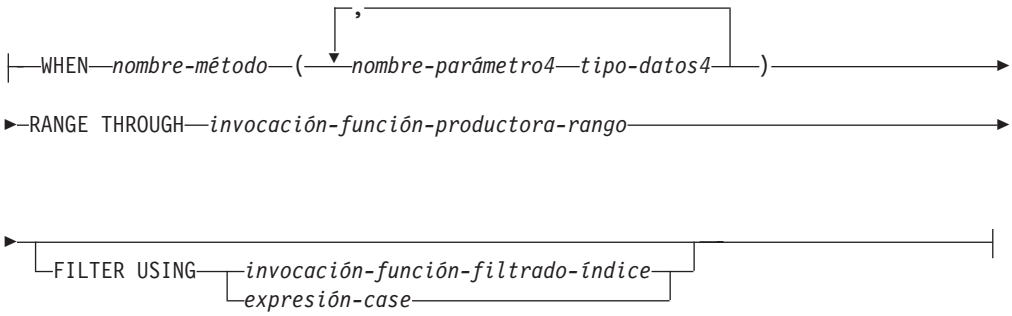
mantenimiento-índices:



búsqueda-índice:



definición-método-búsqueda:



Descripción:

nombre-extensión-índice
Designa la extensión de índice. El nombre (incluido el calificador implícito o explícito) no debe designar una extensión de índice descrita en el catálogo. Si se especifica un *nombre-extensión-índice* compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS'; de lo contrario se devuelve un error (SQLSTATE 42939).

nombre-parámetro1
Identifica un parámetro que se pasa a la extensión de índice, al ejecutar CREATE INDEX, para definir el comportamiento de la extensión de índice. El parámetro que se pasa a la extensión de índice se llama *parámetro de instancia*, pues ese valor define una nueva instancia de una extensión de índice.

nombre-parámetro1 debe ser exclusivo dentro de la definición de la extensión de índice. No se permiten más de 90 parámetros. Si se excede este límite, se produce un error (SQLSTATE 54023).

tipo-datos1
Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que la extensión de índice espera recibir. Los únicos tipos de datos SQL que se pueden especificar son los que se pueden utilizar como constantes, tales como VARCHAR, INTEGER, DECIMAL, DOUBLE o VARGRAPHIC (SQLSTATE 429B5). El valor de parámetro que la extensión de índice recibe al ejecutarse CREATE INDEX debe coincidir exactamente con *tipo-datos1*, incluida la longitud, precisión y escala (SQLSTATE 428E0).

mantenimiento-índices
Especifica cómo se realiza el mantenimiento de las claves de índice de una columna de tipo estructurado o de tipo diferenciado. El mantenimiento de índices es el proceso de transformar la columna fuente en una clave destino. El proceso de transformación se define utilizando una función de tabla que previamente se ha definido en la base de datos.

FROM SOURCE KEY (*nombre-parámetro2 tipo-datos2***)**
Especifica un tipo de datos estructurado o tipo diferenciado para la columna de la clave fuente que está soportada por esta extensión de índice.

nombre-parámetro2
Identifica el parámetro que está asociado a la columna de la clave

fuente. Una columna de clave fuente es la columna de clave de índice (definida en la sentencia CREATE INDEX) que tiene el mismo tipo de datos que *tipo-datos2*.

tipo-datos2
Especifica el tipo de datos de *nombre-parámetro2*. *tipo-datos2* debe ser un tipo estructurado definido por el usuario o un tipo diferenciado cuyo origen no sea LOB, DATALINK, LONG VARCHAR ni LONG VARGRAPHIC (SQLSTATE 42997). Cuando la extensión de índice se asocia con el índice al ejecutar CREATE INDEX, el tipo de datos de la columna de clave de índice debe:

- coincidir exactamente con *tipo-datos2* si es un tipo diferenciado; o bien
- ser el mismo tipo o un subtipo de *tipo-datos2* si es un tipo estructurado

De lo contrario se produce un error (SQLSTATE 428E0).

GENERATE KEY USING *invocación-función-tabla*
Especifica cómo se genera la clave de índice utilizando una función de tabla definida por el usuario. Se pueden crear varias entradas de índice para una clave fuente individual. No se puede duplicar una entrada de índice a partir de una clave fuente individual (SQLSTATE 22526). La función puede utilizar *nombre-parámetro1*, *nombre-parámetro2* o una constante como argumentos. Si el tipo de datos de *nombre-parámetro2* es un tipo de datos estructurado, sólo pueden utilizarse en sus argumentos los métodos observadores de ese tipo estructurado (SQLSTATE 428E3). La salida de la función GENERATE KEY se debe especificar en la especificación TARGET KEY. La salida de la función también se puede utilizar como entrada de la función de filtrado de índice que se especifica en la cláusula FILTER USING.

La función que se utiliza en *invocación-función-tabla* debe cumplir estas condiciones:

- Su resolución debe producir una función de tabla (SQLSTATE 428E4)
- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
- No debe estar definida como NOT DETERMINISTIC (SQLSTATE 428E4) ni como EXTERNAL ACTION (SQLSTATE 428E4)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).
- No debe tener un tipo de datos estructurado, LOB, DATALINK, LONG VARCHAR ni LONG VARGRAPHIC (SQLSTATE 428E3) en el tipo de datos de los parámetros, con la excepción de los métodos observadores generados por el sistema.
- No debe incluir una subconsulta (SQLSTATE 428E3).
- Debe devolver columnas cuyos tipos de datos siguen las restricciones para los tipos de datos de columnas de un índice definido sin la cláusula EXTEND USING.

Si un argumento invoca otra operación o rutina, debe ser un método observador (SQLSTATE 428E3).

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

búsqueda-índice
Especifica cómo se realiza la búsqueda proporcionando una correlación de los argumentos de búsqueda con rangos de búsqueda.

WITH TARGET KEY
Especifica los parámetros de clave destino que son la salida de la función generadora de índices especificada en la cláusula GENERATE KEY USING.

nombre-parámetro3
Identifica el parámetro asociado a una clave destino determinada. *nombre-parámetro3* corresponde a las columnas de la tabla RETURNS tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING. El número de parámetros especificados debe coincidir con el número de columnas devueltas por esa función de tabla (SQLSTATE 428E2).

tipo-datos3
Especifica el tipo de datos de cada *nombre-parámetro3* correspondiente. *tipo-datos3* debe coincidir exactamente con el tipo de datos de las columnas de salida correspondientes de la tabla RETURNS, tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING (SQLSTATE 428E2), incluida la longitud, precisión y tipo.

SEARCH METHODS
Presenta los métodos de búsqueda que están definidos para el índice.

definición-método-búsqueda
Especifica las características del método de la búsqueda por índice. Consta de un nombre de método, los argumentos de búsqueda, una función productora de rangos y una función opcional de filtrado de índice.

WHEN *nombre-método*
Es el nombre de un método de búsqueda. Es un identificador SQL que está asociado con el nombre de método especificado en la norma de explotación de índice (situada en la cláusula PREDICATES de una función definida por el usuario). El *nombre-método-búsqueda* puede estar referenciado por una sola cláusula WHEN en la definición del método de búsqueda (SQLSTATE 42713).

nombre-parámetro4
Identifica el parámetro de un argumento de búsqueda. Estos nombres se utilizan en las cláusulas RANGE THROUGH y FILTER USING.

tipo-datos4
Es el tipo de datos asociado a un parámetro de búsqueda.

RANGE THROUGH *invocación-función-productora-rangos*
Especifica una función de tabla externa que produce rangos de búsqueda. Esta función utiliza *nombre-parámetro1*, *nombre-parámetro4* o una constante como argumentos y devuelve un conjunto de rangos de búsqueda.

La función de tabla que se utiliza en *invocación-función-producción-rango* debe cumplir estas condiciones:

- Su resolución debe producir una función de tabla (SQLSTATE 428E4)
- No debe incluir una subconsulta (SQLSTATE 428E3) ni una función SQL (SQLSTATE 428E4) en sus argumentos
- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)

- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 428E4)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).
- El número y los tipos de los resultados de esta función deben estar relacionados con los resultados de la función de tabla especificada en la cláusula GENERATE KEY USING de la manera siguiente (SQLSTATE 428E1):
  - Devuelven un número máximo de columnas igual al doble de las devueltas por la función de transformación de claves
  - Tienen un número par de columnas, donde la primera mitad de las columnas devueltas definen el inicio del rango (valores de clave de inicio) y la segunda mitad de las columnas devueltas definen el final del rango (valores de clave de parada)
  - Cada columna de clave de inicio tiene el mismo tipo que la correspondiente columna de clave de parada
  - Cada columna de clave de inicio tiene el mismo tipo que la correspondiente columna de la función de transformación de claves.

Más concretamente, sean $a_1:t_1$, ..., $a_n:t_n$ las columnas resultantes y los tipos de datos de la función de transformación. Las columnas resultantes de la *invocación-función-productora-rangos* deben ser $b_1:t_1$, ..., $b_m:t_m$, $c_1:t_1$, ..., $c_m:t_m$, donde m <= n y las columnas "b" son las columnas de clave de inicio y las columnas "c" son las columnas de clave de parada.

Cuando la *invocación-función-productora-rangos* devuelve un valor nulo como valor de clave de inicio o de parada, la semántica no está definida.

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

FILTER USING

Permite especificar una función externa o expresión CASE para filtrar las entradas de índice resultantes de aplicar la función productora de rangos.

invocación-función-filtrado-índice

Especifica una función externa para el filtrado de entradas de índice. Esta función utiliza el *nombre-parámetro1*, *nombre-parámetro3*, *nombre-parámetro4* o una constante como argumentos (SQLSTATE 42703) y devuelve un entero (SQLSTATE 428E4). Si el valor devuelto es 1, la fila correspondiente a la entrada de índice se recupera de la tabla. En otro caso, la entrada de índice no es objeto de más proceso.

Si no se especifica esta opción, no se realiza el filtrado de índice.

La función que se utiliza en la *invocación-función-filtro-índice* debe cumplir las condiciones siguientes:

- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 42845)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).
- No debe tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- No debe incluir una subconsulta (SQLSTATE 428E3)

Si un argumento invoca otra función u otro método, esas cuatro normas también se aplican a la función o el método anidados. Sin embargo, los métodos observadores generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o método utilizado como argumento), siempre que el resultado de evaluar el argumento sea un tipo de datos incorporado.

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

expresión-case
Especifica una expresión CASE para el filtrado de entradas de índice. Se puede utilizar *nombre-parámetro1*, *nombre-parámetro3*, *nombre-parámetro4* o una constante (SQLSTATE 42703) en la *cláusula-when-búsqueda* y en la *cláusula-when-simple*. Como *expresión-resultante* se puede utilizar una función externa con las normas especificadas en FILTER USING *invocación-función-filtrado-índice*. Cualquier función referenciada en la *expresión-case* debe también seguir las cuatro normas descritas para *invocación-función-filtrado-índice*. Además, no se pueden utilizar subconsultas en ningún lugar de la *expresión-case* (SQLSTATE 428E4). La expresión case debe devolver un valor entero (SQLSTATE 428E4). Si la *expresión-resultante* devuelve el valor 1, significa que se conserva la entrada de índice; en otro caso se descarta.

Notas:

- La creación de una extensión de índice con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema, siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

Ejemplos:

Ejemplo 1: El ejemplo siguiente crea una extensión de índice denominada *grid_extension* que utiliza una columna SHAPE de tipo estructurado en una función de tabla denominada *gridEntry* para generar siete claves de destino de índice. Esta extensión de índice proporciona también dos métodos de búsqueda por índice para producir rangos de búsqueda para un argumento de búsqueda determinado.

```
CREATE INDEX EXTENSION GRID_EXTENSION (LEVELS VARCHAR(20) FOR BIT DATA)
  FROM SOURCE KEY (SHAPECOL SHAPE)
  GENERATE KEY USING GRIDENTRY(SHAPECOL..MBR..XMIN,
                               SHAPECOL..MBR..YMIN,
                               SHAPECOL..MBR..XMAX,
                               SHAPECOL..MBR..YMAX,
                               LEVELS)
  WITH TARGET KEY (LEVEL INT, GX INT, GY INT,
                   XMIN INT, YMIN INT, XMAX INT, YMAX INT)
  SEARCH METHODS
  WHEN SEARCHFIRSTBYSECOND (SEARCHARG SHAPE)
  RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                               SEARCHARG..MBR..YMIN,
                               SEARCHARG..MBR..XMAX,
                               SEARCHARG..MBR..YMAX,
                               LEVELS)
  FILTER USING
    CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
       SEARCHARG..MBR..YMAX < YMIN) THEN 0
    ELSE CHECKDUPLICATE(LEVEL, GX, GY,
                               XMIN, YMIN, XMAX, YMAX,
                               SEARCHARG..MBR..XMIN,
```

```
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX,
                                LEVELS)
  END
WHEN SEARCHSECONDBYFIRST (SEARCHARG SHAPE)
RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX,
                                LEVELS)
FILTER USING
  CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
     SEARCHARG..MBR..YMAX < YMIN) THEN 0
  ELSE MBROVERLAP(XMIN, YMIN, XMAX, YMAX,
                                SEARCHARG..MBR..XMIN,
                                SEARCHARG..MBR..YMIN,
                                SEARCHARG..MBR..XMAX,
                                SEARCHARG..MBR..YMAX)
  END
```

Información relacionada:

- "Constantes" en la publicación *Consulta de SQL, Volumen 1*

CREATE METHOD

Esta sentencia asocia un cuerpo de método con una especificación de método que ya forma parte de la definición de un tipo estructurado definido por el usuario.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CREATEIN para el esquema del tipo estructurado al que se hace referencia en la sentencia CREATE METHOD
- Definidor del tipo estructurado referenciado en la sentencia CREATE METHOD.

Para asociar un cuerpo de método externo a su especificación de método, el ID de autorización de la sentencia también debe incluir, como mínimo, una de las siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos.

Al crear un método de SQL, los privilegios del ID de autorización de la sentencia también deben incluir, para cada tabla, vista o apodo que se identifique en cualquier selección completa:

- Privilegio CONTROL para esa tabla, vista o apodo o
- Privilegio SELECT para esa tabla, vista o apodo

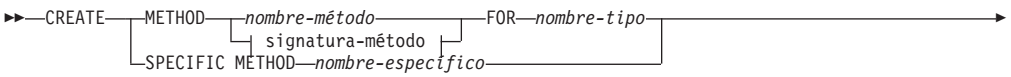
Si el usuario que define un método de SQL sólo puede crear el método porque dicho usuario dispone de autorización SYSADM, se le otorgará autorización DBADM implícita para que pueda crear el método.

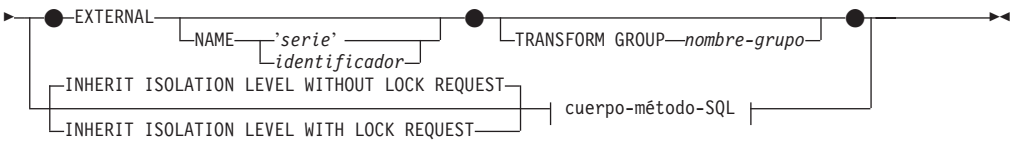
Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE METHOD.

Los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo se aplican al invocarse el método. El ID de autorización de la conexión puede correlacionarse con un ID de autorización remoto distinto.

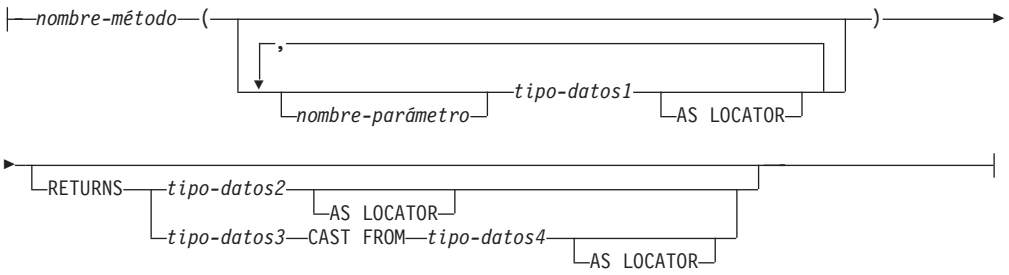
Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:





signatura-método:



cuerpo-método-SQL:

Descripción:

METHOD
Identifica una especificación de método existente que está asociada a un tipo estructurado definido por el usuario. La especificación-método se puede identificar de varias maneras:

nombre-método
Designa la especificación de método para la que se define un cuerpo de método. El esquema implícito es el esquema del tipo indicado (*nombre-tipo*). Debe existir una sola especificación de método para *nombre-tipo* que tenga este *nombre-método* (SQLSTATE 42725).

signatura-método
Proporciona la signatura del método, que identifica de manera exclusiva el método que se debe definir. La signatura de método debe coincidir con la especificación proporcionada en la sentencia CREATE TYPE o ALTER TYPE (SQLSTATE 42883).

nombre-método
Designa la especificación de método para la que se define un cuerpo de método. El esquema implícito es el esquema del tipo indicado (*nombre-tipo*).

nombre-parámetro
Designa el nombre del parámetro. Si la signatura de método proporciona nombres de parámetros, deben coincidir exactamente con las partes correspondientes de la especificación de método asociada. Esta sentencia da soporte a los nombres de parámetros sólo con fines de documentación.

tipo-datos1
Especifica el tipo de datos de cada parámetro.

AS LOCATOR
Para los tipos LOB o los tipos diferenciados que se basan en un tipo LOB, se puede añadir la cláusula AS LOCATOR.

RETURNS
Esta cláusula identifica la salida del método. Si la signatura de método proporciona una cláusula RETURNS, ésta debe coincidir exactamente con la parte correspondiente de la especificación de método asociada existente en CREATE TYPE. Esta sentencia da soporte a la cláusula RETURNS sólo con fines de documentación.

tipo-datos2
Especifica el tipo de datos de la salida.

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que el método debe devolver un localizador de LOB, en lugar del valor real.

tipo-datos3 **CAST FROM** *tipo-datos4*
Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función.

AS LOCATOR
Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede utilizar la cláusula AS LOCATOR para indicar que el método debe devolver un localizador de LOB, en lugar del valor real.

FOR *nombre-tipo*
Designa el tipo para el cual se debe asociar el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

SPECIFIC METHOD *nombre-específico*
Identifica el método concreto, utilizando el nombre específico que se especificó o se tomó por omisión al ejecutar CREATE TYPE. El nombre-específico debe identificar una especificación de método del esquema nombrado o implícito; de lo contrario se produce un error (SQLSTATE 42704).

EXTERNAL
Esta cláusula indica que se utiliza la sentencia CREATE METHOD para registrar un método, basándose en el código escrito en un lenguaje de programación externo y de acuerdo con los convenios documentados para enlaces e interfaces. La especificación-método asociada de CREATE TYPE debe especificar un valor distinto de SQL para LANGUAGE. Cuando se invoca el método, el sujeto del método se pasa a la implementación como primer parámetro implícito.

Si no se especifica la cláusula NAME, se supone "NAME *nombre-método*".

NAME
Esta cláusula identifica el nombre del código escrito por el usuario que aplica el método que se está definiendo.

'serie'
La opción 'serie' es una constante de tipo serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada. Para obtener más información acerca de los convenios específicos de cada lenguaje, consulte "CREATE FUNCTION (escalar externa)".

identificador
Este identificador especificado es un identificador SQL. El identificador SQL se utiliza como id-biblioteca en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre de esquema. Esta modalidad de NAME sólo puede utilizarse con LANGUAGE C (tal como está definido en la especificación-método en CREATE TYPE).

TRANSFORM GROUP *nombre-grupo*
Indica el grupo de transformación que se utiliza, cuando se invoca el método, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación, pues la definición de método incluye un tipo estructurado definido por el usuario.

Se recomienda especialmente especificar un nombre de grupo de transformación; si no se especifica esta cláusula, el nombre de grupo por omisión que se utiliza es DB2_FUNCTION. Si el nombre-grupo especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Similarmente, si una función de transformación necesaria FROM SQL o TO SQL no está definida para el nombre-grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST o **INHERIT ISOLATION LEVEL WITH LOCK REQUEST**
Especifica si una petición de bloqueo puede asociarse o no a una cláusula de aislamiento de la sentencia cuando el método hereda el nivel de aislamiento de la sentencia que invoca el método. El valor por omisión es INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST
Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, no se puede invocar en el contexto de una sentencia de SQL que incluya una cláusula de petición de bloqueo como parte de una cláusula de aislamiento especificada (SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST
Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, también hereda la cláusula de petición de bloqueo especificada.

cuerpo-método-SQL
El cuerpo-método-SQL define cómo se implementa el método si la especificación de método en CREATE TYPE es LANGUAGE SQL.

El cuerpo-método-SQL debe adaptarse a las siguientes partes de la especificación de método:

- DETERMINISTIC o NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION o NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL o READS SQL DATA (SQLSTATE 42985)

En el cuerpo-método-SQL se puede hacer referencia a nombres de parámetros. El sujeto del método se pasa a la implementación de método como un primer parámetro implícito llamado SELF.

Para obtener información adicional, consulte “SQL compuesto (dinámico)” y la “Sentencia RETURN”.

Normas:

- Para poder utilizar CREATE METHOD, se debe definir previamente la especificación de método utilizando la sentencia CREATE TYPE o ALTER TYPE (SQLSTATE 42723).
- Si el método que está creándose es un método de alteración temporal, se invalidan los paquetes que dependen de los métodos siguientes:
  - El método original
  - Otros métodos de alteración temporal cuyo sujeto es un supertipo del método que está creándose

Notas:

- Si el método permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- ***Privilegios***

  El usuario que define un método siempre recibe el privilegio EXECUTE para el método, así como el derecho para poder eliminar el método.

  Si se crea un método EXTERNAL, el usuario que define el método siempre recibe el privilegio EXECUTE y WITH GRANT OPTION.

  Si se crea un método de SQL, el usuario que define el método sólo recibirá el privilegio EXECUTE y WITH GRANT OPTION para el método cuando dicho usuario disponga de WITH GRANT OPTION para todos los privilegios necesarios para definir el método o si dicho usuario dispone de autorización SYSADM o DBADM. El usuario que define un método de SQL sólo adquiere privilegios si, en el momento de crearse el método, existen los privilegios de los que estos privilegios se obtienen. El usuario que define el método debe disponer de estos privilegios directamente o bien porque PUBLIC tiene los privilegios. Los privilegios que tienen los grupos de los que es miembro el usuario que define el método no se consideran. Cuando se utiliza el método, el ID de autorización del usuario conectado debe disponer de privilegios válidos para la tabla o vista a la que hace referencia el apodo en la fuente de datos.
- ***Restricciones de acceso a las tablas***

  Si un método se ha definido como READS SQL DATA, ninguna sentencia del método podrá acceder a una tabla que la sentencia que ha invocado el método está modificando (SQLSTATE 57053).

Ejemplos:

Ejemplo 1:

```
CREATE METHOD BONUS (RATE DOUBLE)
  FOR EMP
  RETURN SELF..SALARY * RATE
```

Ejemplo 2:

```
CREATE METHOD SAMEZIP (addr address_t)
  RETURNS INTEGER
  FOR address_t
  RETURN
```

```
(CASE
  WHEN (self..zip = addr..zip)
    THEN 1
  ELSE 0
END)
```

Ejemplo 3:

```
CREATE METHOD DISTANCE (address_t)
  FOR address_t
  EXTERNAL NAME 'addresslib!distance'
  TRANSFORM GROUP func_group
```

Información relacionada:

- "RETURN" en la página 656
- "SQL compuesto (Dinámico)" en la página 130
- "CREATE FUNCTION (Escalar externa)" en la página 198

CREATE NICKNAME

La sentencia CREATE NICKNAME crea un apodo para un objeto de fuente de datos.

Invocación:

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

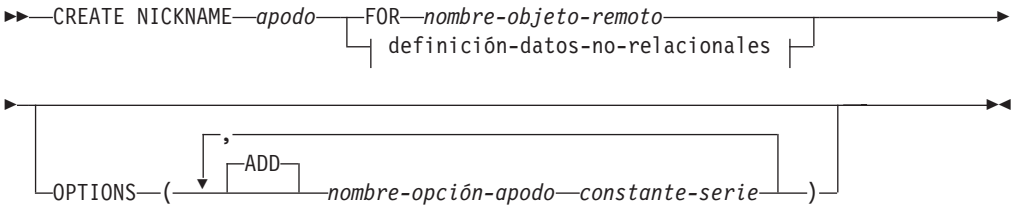
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

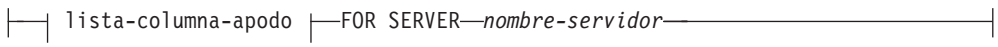
- Autorización IMPLICIT_SCHEMA para la base de datos federada, si el nombre de esquema implícito o explícito del apodo no existe
- Privilegio CREATEIN para el esquema, si existe el nombre de esquema del apodo
- Autorización SYSADM o DBADM

Para las fuentes de datos que requieren una correlación de usuarios, los privilegios contenidos en el ID de autorización en la fuente de datos deben incluir el privilegio para seleccionar datos del objeto que el apodo representa.

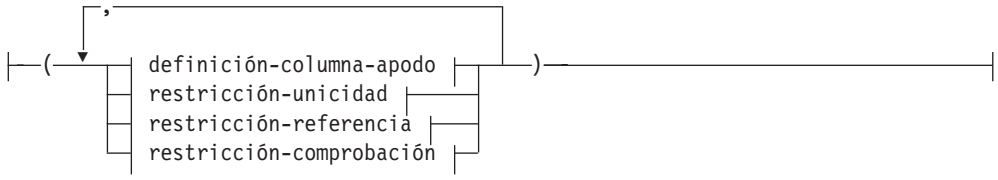
Sintaxis:



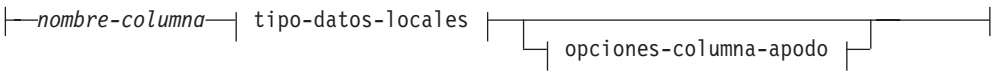
definición-datos-no-relacionales:



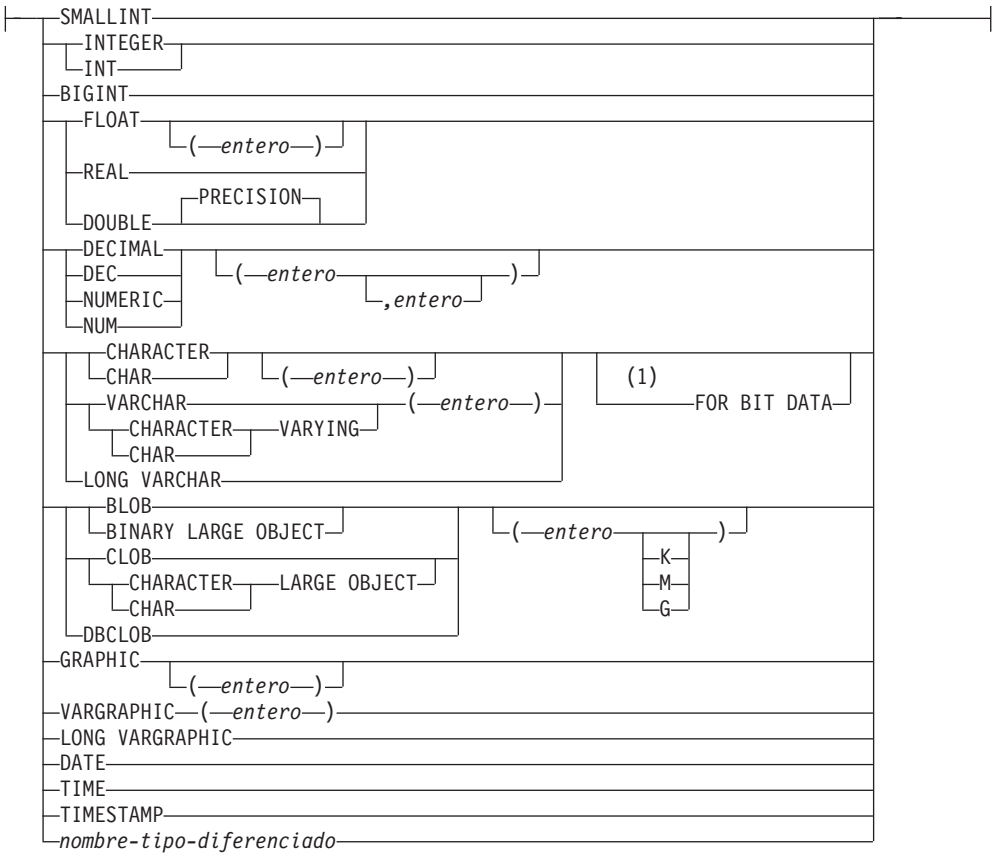
lista-columna-apodo:



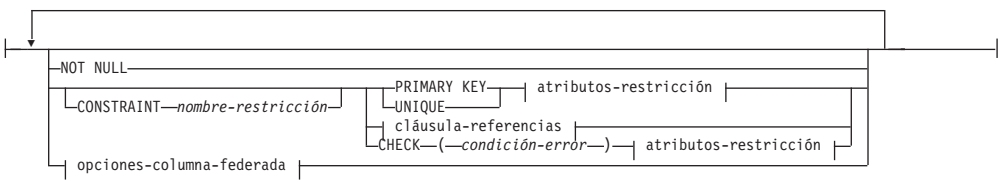
definición-columna-apodo:



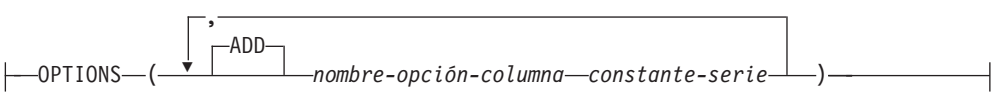
tipo-datos-locales:



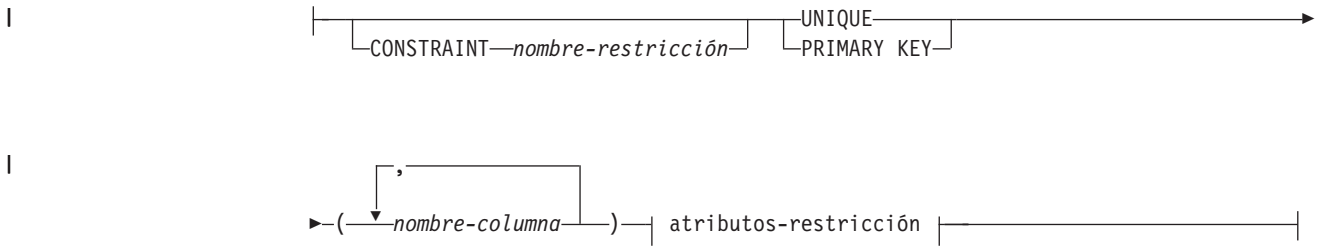
opciones-columna-apodo:



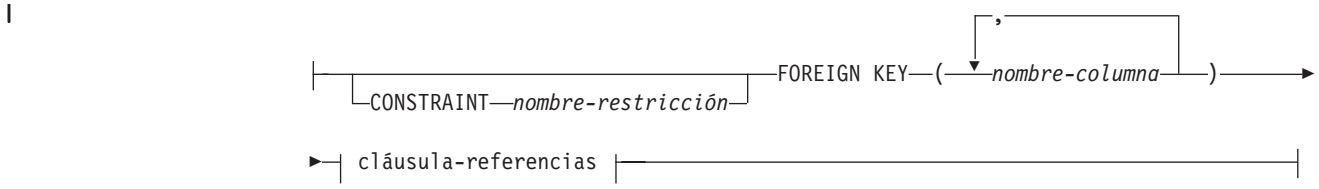
opciones-columna-federada:



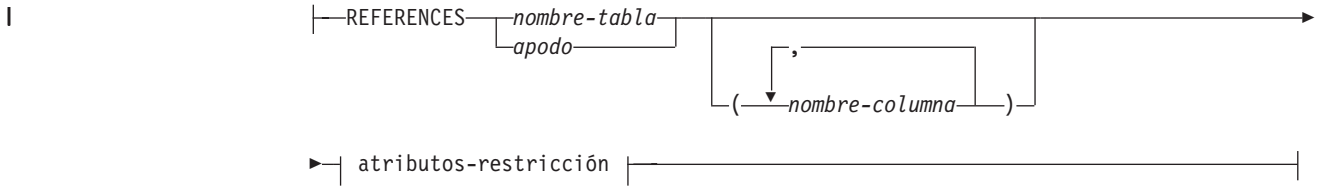
restricción-unicidad:



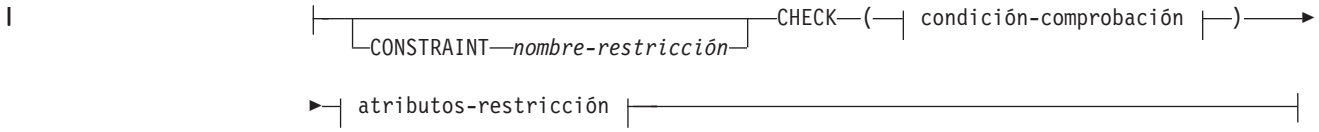
restricción-referencia:



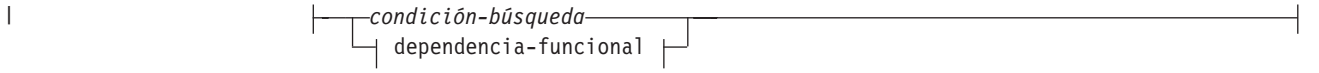
cláusula-referencias:



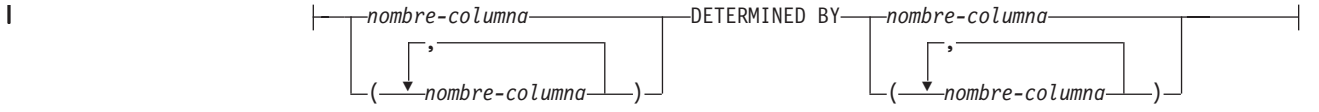
restricción-comprobación:



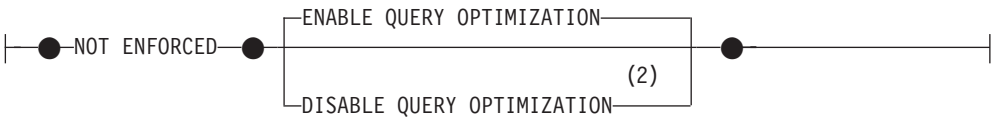
condición-comprobación:



dependencia-funcional:



atributos-restricción:



Notas:

1 La cláusula FOR BIT DATA se puede especificar en cualquier orden con respecto a las restricciones de columna siguientes.

2 No se da soporte a DISABLE QUERY OPTIMIZATION para una restricción de clave primaria o exclusiva.

Descripción:

apodo
Especifica un apodo, el identificador utilizado por el servidor federado para el objeto de fuente de datos. El apodo, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo ni alias descrito en el catálogo. El objeto de fuente de datos no puede ser un alias de DB2. El nombre de esquema no debe empezar por 'SYS' (SQLSTATE 42939).

FOR *nombre-objeto-remoto*
Especifica un identificador. Para fuentes de datos que dan soporte a nombres de esquema, se trata de un identificador compuesto de tres partes con el formato *nombre-fuente-datos.nombre-esquema-remoto.nombre-tabla-remota*. Para las fuentes de datos que no dan soporte a nombres de esquema, se trata de un identificador compuesto de dos partes con el formato *nombre-fuente-datos.nombre-tabla-remota*.

nombre-fuente-datos
Nombra la fuente de datos que contiene la tabla o la vista para la cual se está creando el apodo. El *nombre-fuente-datos* es el mismo nombre que se ha asignado al *nombre-servidor* en la sentencia CREATE SERVER.

nombre-esquema-remoto
Nombra el esquema al que pertenece la tabla o la vista. Si el nombre de esquema remoto contiene caracteres especiales o caracteres en minúsculas, debe especificarse entre comillas dobles.

nombre-tabla-remota
Especifica el nombre del objeto de fuente de datos específico (como, por ejemplo, una tabla o una vista) para el que se crea el apodo. La tabla no puede ser una tabla temporal declarada (SQLSTATE 42995). Si el nombre de tabla remota contiene caracteres especiales o caracteres en minúsculas, debe especificarse entre comillas dobles.

definición-datos-no-relacionales
Define los datos a los que se debe acceder a través de un reiniciador no relacional.

definición-columna-apodo
Define los atributos locales de la columna para el apodo. Algunos reiniciadores necesitan que se especifiquen estos atributos, mientras que otros reiniciadores admiten que los atributos se determinen a partir de la fuente de datos.

nombre-columna
Especifica el nombre local de la columna. El nombre puede ser diferente del de la columna correspondiente del *nombre-objeto-remoto*.

tipo-datos-locales
Especifica el tipo de datos locales para la columna. Algunos reiniciadores sólo dan soporte a un subconjunto de tipos de datos SQL. Para ver descripciones de tipos de datos específicos, consulte la descripción de la sentencia "CREATE TABLE".

opciones-columna-apodo
Especifica opciones adicionales relacionadas con columnas del apodo.

NOT NULL
Especifica que la columna no admite valores nulos.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar una restricción que ya se haya especificado en la misma sentencia CREATE NICKNAME (SQLSTATE 42710).

Si se omite esta cláusula, el sistema genera un identificador de 18 caracteres que es exclusivo entre los identificadores de las restricciones existentes que se han definido en el apodo. (El identificador se compone de la palabra 'SQL' seguida de una secuencia de 15 caracteres numéricos generados por una función basada en la indicación de la hora.)

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, se puede utilizar el *nombre-restricción* como nombre de una especificación de índice que se crea para dar soporte a la restricción.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. De este modo, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada.

Consulte el apartado PRIMARY KEY en la descripción de la *restricción-unicidad* más abajo.

UNIQUE
Proporciona un método abreviado de definir una clave de unicidad compuesta de una sola columna. Por lo tanto, si se especifica UNIQUE en la definición de la columna C, el efecto es el mismo que si se especificase la cláusula UNIQUE(C) como una cláusula separada.

Consulte el apartado UNIQUE en la descripción de la *restricción-unicidad* más abajo.

cláusula-referencias
Proporciona un método abreviado de definir una clave externa compuesta de una sola columna. Así, si se especifica una cláusula-referencias en la definición de la columna C, el efecto es el mismo que si se especificase esa cláusula-referencias como parte de una cláusula FOREIGN KEY en la que C fuera la única columna identificada.

Consulte *cláusula-referencias* en *restricción-referencia*, que encontrará más adelante.

CHECK (*condición-error***)**
Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Vea CHECK (*condición-error*) más adelante.

OPTIONS
Indica las opciones de columna que se añaden cuando se crea el apodo. Algunos reiniciadores requieren que se especifiquen algunas opciones de columna.

ADD
Añade una opción de columna.

nombre-opción-columna
Especifica el nombre de la opción.

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

restricción-unicidad
Define una restricción de clave primaria o de unicidad.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de clave primaria o de unicidad.

UNIQUE (*nombre-columna***,...)**
Define una clave de unicidad compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado “Cuentas de bytes” en la página 393 para obtener información acerca de las longitudes almacenadas). No se puede utilizar LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado basado en uno de estos tipos ni un tipo estructurado como parte de una clave exclusiva, incluso aunque el atributo de longitud de la columna sea lo suficientemente pequeño para caber dentro del límite de 1024 bytes (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

La descripción del apodo, tal como se ha registrado en el catálogo, incluye la clave exclusiva y su especificación de índice. Se creará automáticamente una especificación de índice para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre de la especificación de índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice o una especificación de índice existente en el esquema donde se crea el apodo. Si el nombre de la especificación de índice entra en conflicto, el nombre será la palabra ’SQL’ seguida de los caracteres de la indicación de la hora (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

PRIMARY KEY (*nombre-columna***,...)**
Define una clave primaria formada por las columnas identificadas. La cláusula no debe especificarse más de una vez y las columnas identificadas deben definirse como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 393 para obtener información acerca de las longitudes almacenadas). No se puede utilizar LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado basado en uno de estos tipos ni un tipo estructurado como parte de una clave primaria, incluso aunque el atributo de longitud de la columna sea lo suficientemente pequeño para caber dentro del límite de 1024 bytes (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

Sólo se puede definir una única clave primaria en un apodo.

La descripción del apodo, tal como está registrada en el catálogo, incluye la clave primaria y su especificación de índice. Se creará automáticamente una especificación de índice para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre de la especificación de índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice o una especificación de índice existente en el esquema donde se crea el apodo. Si el nombre de la especificación de índice entra en conflicto, el nombre será la palabra 'SQL', seguida por los caracteres de la indicación de la hora (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

restricción-referencia
Define una restricción de referencia.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de referencia.

FOREIGN KEY (*nombre-columna***,...)**
Define una restricción de referencia con el *nombre-restricción* especificado.

Deje que N1 indique el apodo de objeto de la sentencia. La clave foránea de la restricción de referencia se compone de las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de N1 y la misma columna no se debe identificar más de una vez. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 393 para obtener información acerca de las longitudes almacenadas). Las claves foráneas pueden definirse en columnas de longitud variable cuya longitud sea superior a 255 bytes. No se puede utilizar una columna LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, de tipo diferenciado basado en uno de estos tipos ni de tipo estructurado como parte de una clave foránea (SQLSTATE 42962). Debe haber el mismo número de columnas de clave foránea que hay en la

clave padre y los tipos de datos de las columnas correspondientes deben ser compatibles (SQLSTATE 42830). Dos descripciones de columna son compatibles si contienen tipos de datos compatibles (ambas columnas son numéricas, de serie de caracteres, gráficas, de fecha y hora o tienen el mismo tipo diferenciado).

cláusula-referencias
Especifica la tabla padre o el apodo padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla* o *apodo*
La tabla o el apodo especificados en una cláusula REFERENCES debe identificar una tabla base o un apodo que se describa en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción de referencia está duplicada si su clave foránea, clave padre y tabla padre o apodo padre son iguales a la clave foránea, clave padre y tabla padre o apodo padre de una restricción de referencia especificada previamente. Las restricciones de referencia duplicadas se pasan por alto y se devuelve un aviso (SQLSTATE 01543).

En la siguiente explicación, N2 indicará la tabla padre identificada o el apodo padre y N1 indicará el apodo que se está creando (o modificando). N1 y N2 pueden ser el mismo apodo.

La clave foránea especificada debe tener el mismo número de columnas que la clave padre de N2, y la descripción de la columna número *n* de la clave foránea debe ser comparable a la descripción de la columna número *n* de esa clave padre. Las columnas de indicación de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta norma.

La restricción de referencia especificada por una cláusula FOREIGN KEY define una relación en la que N2 es el padre y N1 es dependiente.

(*nombre-columna***,...)**
La clave padre de la restricción de referencia se compone de las columnas identificadas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de N2. La misma columna no se puede identificar más de una vez.

La lista de nombres de columnas debe coincidir con el conjunto de columnas (en cualquier orden) de la clave primaria o una restricción exclusiva que exista en N2 (SQLSTATE 42890). Si no se especifica una lista de nombres de columna, N2 debe tener una clave primaria (SQLSTATE 42888). La omisión de la lista de nombres de columna es una especificación implícita de las columnas de dicha clave primaria en la secuencia especificada originalmente.

atributos-restricción
Define los atributos que se asocian a las restricciones de comprobación o de integridad referencial.

NOT ENFORCED
El gestor de bases de datos no aplica la restricción durante la realización de las operaciones normales como, por ejemplo, la inserción, la actualización o la supresión.

ENABLE QUERY OPTIMIZATION
Se supone que la restricción es verdadera y se puede utilizar para la optimización de la consulta bajo las circunstancias adecuadas.

DISABLE QUERY OPTIMIZATION
La restricción no puede utilizarse para la optimización de la consulta.

restricción-comprobación
Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o que define una dependencia funcional entre columnas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de comprobación.

CHECK (*condición-error***)**
Define una restricción de comprobación. La *condición-comprobación* debe ser verdadera o desconocida para cada fila del apodo.

condición-búsqueda
La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia a columna debe ser a una columna del apodo que se crea.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- No puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
  - Subconsultas
  - Operaciones de eliminación de referencia o funciones DEREF donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
  - Especificaciones CAST con una cláusula SCOPE
  - Funciones de columna
  - Funciones que no sean deterministas
  - Funciones definidas para que exista una acción externa
  - Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
  - Variables del lenguaje principal
  - Marcadores de parámetro
  - Registros especiales
  - Referencias a columnas generadas que no correspondan a la columna de identidad

dependencia-funcional
Define una dependencia funcional entre columnas.

El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la

columna no debe ser un tipo de datos LOB, un tipo diferenciado basado en el tipo de datos LOB ni un tipo estructurado (SQLSTATE 42962). Ninguna columna del conjunto de columnas hijo puede ser una columna anulable (SQLSTATE 42621).

Si se especifica una restricción de comprobación como parte de una *definición-columna*, sólo puede establecerse una referencia de columna que haga referencia a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de apodo pueden contener referencias a columna que identifiquen columnas definidas previamente en la sentencia CREATE NICKNAME. No se comprueba si hay incoherencias, condiciones duplicadas ni condiciones equivalentes en las restricciones de comprobación. Por lo tanto, se pueden definir restricciones de comprobación contradictorias o redundantes, lo que podría dar lugar a posibles errores en tiempo de ejecución.

FOR SERVER *nombre-servidor*
Especifica el servidor que se ha registrado utilizando la sentencia CREATE SERVER. Este servidor se utilizará para acceder a los datos para el apodo.

OPTIONS
Indica las opciones de apodo que se habilitan cuando se crea el apodo.

ADD
Añade una opción de apodo.

nombre-opción-apodo
Especifica el nombre de la opción.

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

Notas:

- Las tablas y las vistas son ejemplos de objetos de fuente de datos relacionales. Los objetos de Documentum o tablas registradas, los archivos de texto (.txt), los objetos en los que puede ejecutar una búsqueda BLAST y los archivos Microsoft Excel (.xls) son ejemplos de objetos de datos no relacionales.
- El objeto de fuente de datos al que el apodo hace referencia ya debe existir en la fuente de datos indicada por el primer calificador en el *nombre-objeto-remoto*.
- La lista de los tipos de datos de fuente de datos soportados varía según el reiniciador. Los iniciadores no dan soporte a los tipos de datos de fuente de datos que corresponden a los tipos de datos DB2 siguientes: DATALINK, tipos estructurados y tipos REF. Cuando la sentencia CREATE NICKNAME especifica un *nombre-objeto-remoto* que tiene columnas con tipos de datos no soportados, se devuelve un error.

  Los tipos de datos de fuente de datos LONG VARCHAR y LONG VARGRAPHIC se correlacionan con los tipos de datos CLOB y DBCLOB, respectivamente. LONG VARCHAR FOR BIT DATA se correlaciona con BLOB.
- La longitud máxima permitida de los nombres de índice de DB2 es de 18 caracteres. Si se está creando un apodo para una tabla relacional que tiene un índice cuyo nombre sobrepasa esta longitud, no se cataloga todo el nombre. En su lugar, DB2 lo trunca a 18 caracteres. Si la serie formada por estos caracteres no es exclusiva en el esquema al que pertenece el índice, DB2 intenta convertirla en exclusiva sustituyendo el último carácter por un 0. Si el resultado continúa sin ser exclusivo, DB2 cambia el último carácter por un 1. DB2 repite este

proceso con los números 2 a 9 y, si es necesario, con los números 0 a 9 para el decimoséptimo carácter del nombre, para el decimosexto y así sucesivamente hasta que se genera un número exclusivo. Como ilustración: El índice de la tabla de fuente de datos se llama ABCDEFGHIJKLMNOPQRSTUVWXYZ. Los nombres ABCDEFGHIJKLMNOPQR y ABCDEFGHIJKLMNOPQ0 ya existen en el esquema al que pertenece el índice. El nuevo nombre tiene más de 18 caracteres; por lo tanto, DB2 lo trunca a ABCDEFGHIJKLMNOPQR. Puesto que este nombre ya existe en el esquema, DB2 cambia la versión truncada por ABCDEFGHIJKLMNOPQ0. Y puesto que este nombre también existe, DB2 cambia la versión truncada por ABCDEFGHIJKLMNOPQ1. Este nombre aún no existe en el esquema, por lo que DB2 lo acepta como el nuevo nombre.

- Cuando se crea un apodo para un objeto de fuente de datos, DB2 almacena los nombres de las columnas de apodo en el catálogo. Cuando el objeto de fuente de datos es una tabla o una vista, DB2 hace que los nombres de las columnas de apodo sean los mismos que los nombres de las columnas de tabla o de vista. Si un nombre excede la longitud máxima permitida para los nombres de columna de DB2, DB2 trunca el nombre en función de esa longitud. Si la versión truncada no es exclusiva entre los demás nombres de las columnas de la tabla o vista, DB2 hace que sea exclusivo siguiendo el procedimiento que se describe en el párrafo anterior.
- Si se cambia la definición de un objeto de fuente de datos remota (por ejemplo, se suprime una columna o se cambia el tipo de datos), se debe descartar y volver a crear el apodo; de lo contrario, se pueden producir errores cuando se utilice el apodo en una sentencia de SQL.

Ejemplos:

*Ejemplo 1:*Cree un apodo para una vista, DEPARTMENT, que esté en un esquema llamado HEDGES. Esta vista se almacena en una fuente de datos DB2 UDB para z/OS y OS/390 denominada OS390A.

```
CREATE NICKNAME DEPT
  FOR OS390A.HEDGES.DEPARTMENT
```

Ejemplo 2: Seleccione todos los registros de la vista para la cual se ha creado un apodo en el ejemplo 1. Se debe hacer referencia a la vista por su apodo. Se puede hacer referencia a la vista remota utilizando el nombre por el cual se conoce en la fuente de datos únicamente en sesiones de paso a través.

```
SELECT * FROM DEPT                          Válido después de crear el apodo DEPT

SELECT * FROM OS390A.HEDGES.DEPARTMENT      No válido
```

Ejemplo 3: Cree un apodo para la tabla remota JAPAN que está en un esquema denominado salesdata. Puesto que el nombre de esquema y el nombre de tabla de la fuente de datos se almacenan en minúsculas, especifique el nombre de esquema remoto y el nombre de tabla entre comillas dobles:

```
CREATE NICKNAME JPSALES
  FOR asia."salesdata"."japan"
```

Ejemplo 4: Cree un apodo para el archivo estructurado de tabla DRUGDATA1.TXT. Incluya las opciones de apodo FILE_PATH, COLUMN DELIMITER, KEY_COLUMN y VALIDATE_DATA_FILE en la sentencia.

```
CREATE NICKNAME DRUGDATA1
  (Dcode         INTEGER,
  DRUG           CHAR(20),
  MANUFACTURER   CHAR(20))
  FOR SERVER biochem_lab
```

```
OPTIONS
 (FILE_PATH '/usr/pat/DRUGDATA1.TXT',
  COLUMN_DELIMITER ',',
  KEY_COLUMN 'DCODE',
  SORTED 'Y',
  VALIDATE_DATA_FILE 'Y')
```

Ejemplo 5: Cree el apodo padre CUSTOMERS en múltiples archivos XML bajo la vía de acceso de directorio especificada /home/db2user. Incluya las opciones siguientes:

- Opciones de columna:
  - La opción de columna XPATH para la columna VARCHAR(5) denominada ID, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna XPATH para la columna VARCHAR(16) denominada NAME, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna XPATH para la columna VARCHAR(30) denominada ADDRESS, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna PRIMARY_KEY para la columna VARCHAR(16) denominada CID, que identifica el apodo de los clientes como apodo padre en una jerarquía de apodos
- Opciones de apodo:
  - La opción de apodo DIRECTORY_PATH para indicar la ubicación de los archivos XML que proporcionan los datos
  - La opción de apodo XPATH para indicar el elemento de los archivos XML donde empiezan los datos
  - La opción de apodo STREAMING para indicar que los datos fuente XML se separan y procesan elemento por elemento. En este ejemplo, el elemento es un registro de cliente.

```
CREATE NICKNAME clientes
  (id      VARCHAR(5)   OPTIONS(XPATH './@id'),
   name    VARCHAR(16)  OPTIONS(XPATH './/name'),
   address VARCHAR(30)  OPTIONS(XPATH './/address/@street'),
   cid     VARCHAR(16)  OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS
     (DIRECTORY_PATH '/home/db2user',
      XPATH '//customer',
      STREAMING 'YES')
```

Información relacionada:

- "CREATE TABLE" en la página 341
- "ALTER NICKNAME" en la página 27
- "CREATE SERVER" en la página 337
- "Nickname column options for federated systems" en la publicación *Federated Systems Guide*
- "Valid data source objects" en la publicación *Federated Systems Guide*

CREATE PROCEDURE

La sentencia CREATE PROCEDURE define un procedimiento con un servidor de aplicaciones.

Existen dos tipos distintos de procedimientos que pueden crearse mediante la utilización de esta sentencia. Cada una de ellas se describe por separado.

- Externo. El cuerpo del procedimiento se escribe en un lenguaje de programación. Al ejecutable externo hace referencia un procedimiento que se ha definido con un servidor de aplicaciones, junto con diversos atributos del procedimiento.
- SQL. El cuerpo del procedimiento se escribe en SQL. El cuerpo del procedimiento se define con un servidor de aplicaciones junto con diversos atributos del procedimiento.

Información relacionada:

- "CREATE PROCEDURE (Externo)" en la página 308
- "CREATE PROCEDURE (SQL)" en la página 322

CREATE PROCEDURE (Externo)

La sentencia CREATE PROCEDURE (Externo) se utiliza para definir un procedimiento externo con un servidor de aplicaciones.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del procedimiento no hace referencia a un esquema existente
  - Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente

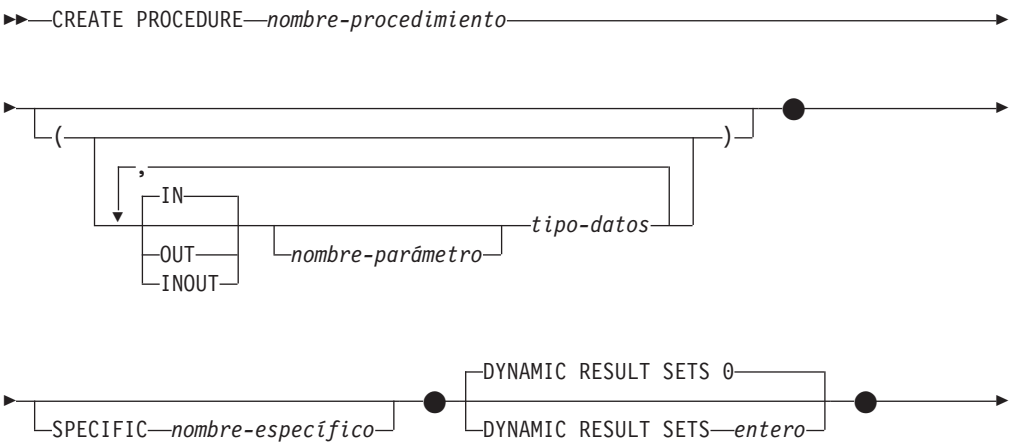
Para crear un procedimiento almacenado no delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, una de las siguientes:

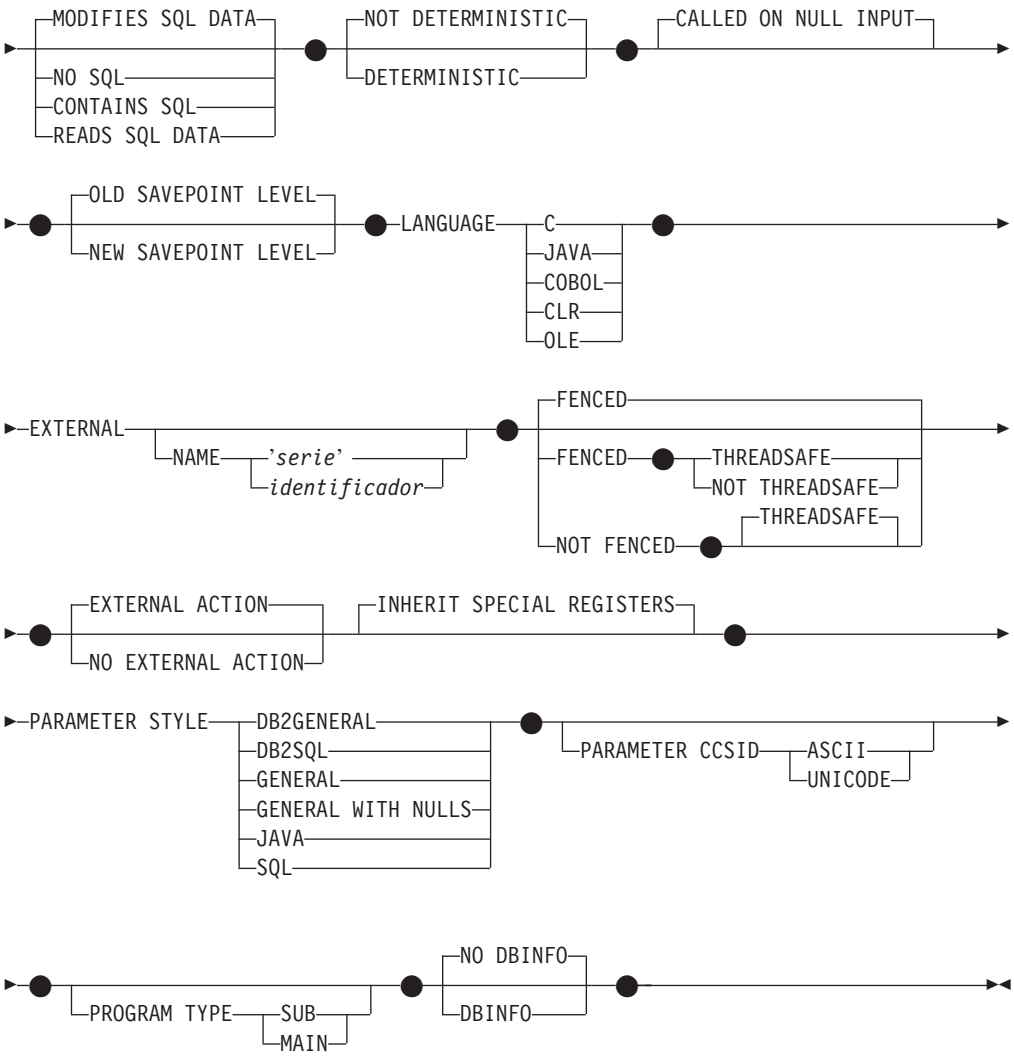
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM.

Para crear un procedimiento almacenado restringido, no se necesitan ni autorizaciones ni privilegios adicionales.

Si el ID de autorización no tiene la autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:





Descripción:

nombre-procedimiento
Indica el nombre del procedimiento que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. La forma no calificada de *nombre-procedimiento* es un identificador SQL (con una longitud máxima de 128). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica explícitamente el calificador para los nombres de objetos no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros no debe designar un procedimiento descrito en el catálogo (SQLSTATE 42723). No es necesario que el nombre no calificado, junto con el número de parámetros, sea exclusivo en los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

*(***IN** | **OUT** | **INOUT** *nombre-parámetro tipo-datos,...)*
Identifica los parámetros del procedimiento y especifica la modalidad, el tipo

de datos y el nombre opcional de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el procedimiento va a esperar.

En un esquema, no se permite que dos procedimientos que se denominen igual tengan exactamente el mismo número de parámetros. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

la segunda sentencia no se ejecutará satisfactoriamente porque el número de parámetros del procedimiento es el mismo, aunque los tipos de datos no lo sean.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT
Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT
Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro
Opcionalmente, especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos
Especifica el tipo de datos del parámetro.

- Pueden especificarse abreviaturas y especificaciones de tipo de datos de SQL, que pueden indicarse en la definición de *tipo-datos* de una sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje que está utilizándose para escribir el procedimiento.
- No se da soporte a los tipos de datos definidos por el usuario (SQLSTATE 42601).
- No se da soporte a los tipos de datos LONG VARCHAR y LONG VARGRAPHIC como tipos de parámetros para procedimientos externos.
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia del procedimiento que se está definiendo. El nombre específico puede utilizarse al eliminar el procedimiento o realizar un comentario en el procedimiento. No puede utilizarse nunca para invocar el procedimiento. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe identificar otra instancia de rutina que exista en el servidor de aplicaciones; de lo contrario, se genera un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-procedimiento* ya existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, éste debe ser el mismo que el calificador explícito o implícito del *nombre-procedimiento*; de lo contrario, se genera un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de la hora en caracteres SQLaammddhhmmsshhn.

DYNAMIC RESULT SETS *entero*
Indica el enlace lógico superior estimado de los conjuntos del resultado devueltos para el procedimiento almacenado.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA
Indica si el procedimiento almacenado emite sentencias de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que el procedimiento almacenado no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL
Indica que el procedimiento almacenado puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004). Las sentencias que no reciben soporte en ningún procedimiento almacenado devuelven un error distinto (SQLSTATE 38003).

READS SQL DATA
Indica que en el procedimiento almacenado pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ningún procedimiento almacenado devuelven un error distinto (SQLSTATE 38003).

MODIFIES SQL DATA
Indica que el procedimiento almacenado puede ejecutar cualquier sentencia de SQL a excepción de las sentencias que no reciben soporte en los procedimientos almacenados (SQLSTATE 38003).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula especifica si el procedimiento devuelve siempre los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si el procedimiento depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un procedimiento DETERMINISTIC debe siempre devolver el mismo resultado ante invocaciones sucesivas con entradas de datos idénticas.

Actualmente esta cláusula no afecta al proceso del procedimiento almacenado.

CALLED ON NULL INPUT
>CALLED ON NULL INPUT siempre se aplica a procedimientos almacenados. Esto significa que el procedimiento almacenado se llamará, con independencia de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento almacenado comprobar si hay valores argumento nulos.

OLD SAVEPOINT LEVEL o **NEW SAVEPOINT LEVEL**
Especifica si este procedimiento almacenado establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información acerca de los niveles de punto de salvaguarda, consulte la sección "Normas" de la descripción de la sentencia SAVEPOINT.

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que se está escrito el cuerpo del procedimiento almacenado.

C Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuese un procedimiento C. El procedimiento almacenado debe ajustarse al convenio de llamada y enlace del lenguaje C, tal como se define por el prototipo C ANSI estándar.

JAVA
Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuera un método de una clase Java.

COBOL
Esto significa que el gestor de bases de datos llamará al procedimiento como si fuera un procedimiento COBOL.

CLR
Significa que el gestor de bases de datos llamará al procedimiento almacenado como un método en una clase .NET. En este momento, sólo se da soporte a LANGUAGE CLR para procedimientos almacenados que se ejecutan en sistemas operativos Windows. NOT FENCED no se puede especificar para una rutina CLR (SQLSTATE 42601).

OLE
Esto significa que el gestor de bases de datos llamará al procedimiento almacenado como si fuera un método expuesto por un objeto de automatización OLE. El procedimiento almacenado debe ajustarse a los tipos de datos de automatización y al mecanismo de invocación de OLE. Además, el objeto de automatización OLE necesita implementarse como servidor en proceso (DLL). Estas restricciones se describen en la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para los procedimientos almacenados que se han almacenado en DB2 para sistemas operativos Windows. THREADSAFE no puede especificarse para los procedimientos que se han definido con LANGUAGE OLE (SQLSTATE 42613).

EXTERNAL
Esta cláusula indica que la sentencia CREATE PROCEDURE se emplea para registrar un nuevo procedimiento basado en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

Si no se especifica la cláusula NAME, se supone "NAME *nombre-procedimiento*". Si la cláusula NAME no está formateada correctamente, se devuelve un error (SQLSTATE 42878).

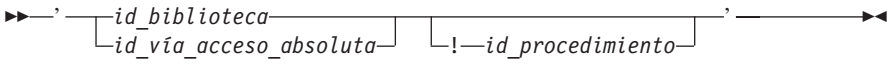
NAME *'serie'*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta el procedimiento que se está definiendo.

La opción 'serie' es una constante de serie con un máximo de 254 caracteres. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y el procedimiento de la biblioteca que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado que se está creando (CREATE). Al ejecutar la sentencia CREATE PROCEDURE, no es preciso

que exista la biblioteca (ni el procedimiento dentro de la biblioteca). Sin embargo, cuando se llama el procedimiento, deben existir la biblioteca y el procedimiento en la biblioteca y deben ser accesibles desde la máquina servidora de la base de datos.



El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

id_biblioteca
Identifica el nombre de la biblioteca donde reside el procedimiento. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

- En sistemas basados en UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará el procedimiento en la biblioteca /u/production/sqllib/function/myproc si se ha especificado FENCED o en /u/production/sqllib/function/unfenced/myproc si se ha especificado NOT FENCED.
- En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

Los procedimientos almacenados en cualquiera de estos directorios no utilizan ninguno de los atributos registrados.

id_vía_acceso_absoluta
Identifica el nombre completo de vía de acceso del procedimiento.

En sistemas basados en UNIX, por ejemplo, '/u/jchui/mylib/myproc' haría que el gestor de bases de datos buscara el procedimiento myproc en /u/jchui/mylib.

En sistemas operativos Windows, 'd:\mylib\myproc.dll' haría que el gestor de bases de datos cargara el archivo myproc.dll desde el directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id_procedimiento
Identifica el nombre del punto de entrada del procedimiento que se debe invocar. El signo de admiración (!) sirve de delimitador entre el ID de biblioteca y el ID de procedimiento. Si se omite *! id_procedimiento*, el gestor de bases de datos utilizará el punto de entrada por omisión establecido cuando se ha enlazado la biblioteca.

Por ejemplo, '!proc8' indicaría al gestor de bases de datos que buscara el ID de procedimiento especificado basándose en las normas del sistema y que utilizara el punto de entrada proc8 de esa biblioteca.

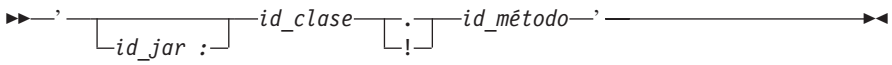
De forma similar, '!proc8' indicaría al gestor de bases de datos que buscara la biblioteca en la ubicación especificada por el *id_vía_acceso_absoluta* y que utilizara el punto de entrada proc8 de esa biblioteca.

Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

El cuerpo de cada procedimiento almacenado debe encontrarse en un directorio que se haya montado y que esté disponible en todas las particiones de la base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado que se está creando. No es preciso que existan el identificador de clase y el identificador de método cuando se ejecuta la sentencia CREATE PROCEDURE. Si se especifica un *id_jar*, éste deberá existir cuando se ejecute la sentencia CREATE PROCEDURE. Sin embargo, cuando se llama al procedimiento, el identificador de clase y el identificador de método deben existir y poderse acceder desde la máquina del servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42884).



  El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

  id_jar
  Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'.

  id_clase
  Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo de paquete completo, por ejemplo, 'misPaqs.StoredProcs'. La máquina virtual Java buscará en el directorio '../miPaqs/UserFuncs/' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '..\misPaqs\StoredProcs\'.

  id_método
  Identifica el nombre de método de la clase Java que se debe invocar.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método dentro de la clase que el gestor de bases de datos invoca para ejecutar el procedimiento que se está creando. No es necesario que existan el módulo, la clase y el método cuando se ejecute la sentencia CREATE PROCEDURE. Sin embargo, cuando se llama al procedimiento, el módulo, la clase y el método deben existir y poderse acceder desde la máquina de servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42284).

  Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que la infraestructura .NET se está utilizando en un procedimiento almacenado para tomar las decisiones necesarias en

tiempo de ejecución. Todos los procedimientos almacenados utilizando la infraestructura .NET deben estar catalogados como 'LANGUAGE CLR'.

El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

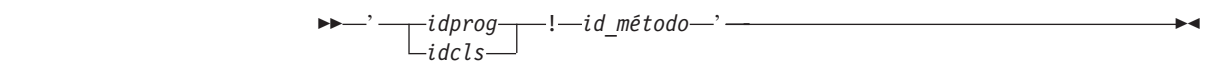
ensamblaje
Identifica el DLL u otro archivo de ensamblaje en el que reside la clase. Se debe especificar alguna extensión de archivo (por ejemplo .dll). Si no se facilita el nombre de vía de acceso completo, el archivo debe residir en el directorio function de la vía de acceso de instalación de DB2 (por ejemplo, c:\sqllib\function). Si el archivo reside en un subdirectorio del directorio function de la instalación, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instalación es c:\sqllib y el archivo de ensamblaje es c:\sqllib\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

id_clase
Especifica el nombre de la clase del ensamblaje en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id_método
Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La serie de caracteres especificada es el identificador de programa OLE (*idprog*) o el identificador de clase (*idcls*) y el identificador de método (*id_método*), que el gestor de bases de datos invoca para ejecutar el procedimiento almacenado que la sentencia está creando. No es necesario que el identificador de programa o de clase ni el identificador de método existan cuando se ejecuta la sentencia CREATE PROCEDURE. Sin embargo, cuando el procedimiento se utiliza en una sentencia CALL, el identificador de método debe existir y ser accesible desde la máquina del servidor de bases de datos; de lo contrario se produce un error (SQLSTATE 42724).



El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

idprog
Identifica el identificador de programa del objeto OLE.

El gestor de bases de datos no interpreta un *idprog*; sólo lo reenvía al controlador de automatización de OLE en tiempo de ejecución. El objeto OLE especificado debe poderse crear y debe dar soporte al enlace tardío (llamado también enlace basado en IDispatch). Según especifica el convenio, los idprog tienen el formato siguiente:

```
<nombre_programa>.<nombre_componente>.<versión>
```

Puesto que sólo se trata de un convenio y no de una norma, los *idprog* pueden tener, de hecho, un formato distinto.

idcls
Designa el identificador de clase del objeto OLE que se debe crear. Puede utilizarse como una alternativa a la especificación de un idprog cuando un objeto OLE no está registrado con un idprog. El *idcls* tiene el formato:

```
{nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn}
```

donde 'n' es un carácter alfanumérico. El gestor de bases de datos no interpreta un *idcls*; sólo lo reenvía a las API de OLE en tiempo de ejecución.

id_método
Identifica el nombre de método del objeto OLE que se debe invocar.

NAME *identificador*
Este *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre del esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

FENCED o **NOT FENCED**
Esta cláusula especifica si se considera que el procedimiento almacenado es "seguro" para ejecutarse en el espacio de dirección o en el proceso del entorno operativo del gestor de bases de (NOT FENCED) o no (FENCED).

Si un procedimiento almacenado se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que el procedimiento no pueda acceder a ellos. Todos los procedimientos tienen la opción de ejecutarse como FENCED o como NOT FENCED. En general, un procedimiento que se ejecute como FENCED no funcionará tan bien como otro de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para los procedimientos que no se han comprobado de forma adecuada puede comprometer la integridad de DB2. DB2 dispone de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no puede garantizar la integridad completa cuando se utilizan procedimientos almacenados NOT FENCED.

Para registrar un proceso almacenado como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial

(CREATE_NOT_FENCED). Para un procedimiento almacenado con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED.

No se pueden crear procedimientos almacenados LANGUAGE CLR cuando se especifica la cláusula NOT FENCED (SQLSTATE 42601).

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

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

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

Para los procedimientos FENCED, THREADSAFE es el valor por omisión si LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si el procedimiento se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para los procedimientos NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

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

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación.

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

PARAMETER STYLE
Esta cláusula sirve para especificar los convenios que se emplean para pasar los parámetros a los procedimientos almacenados y para devolver el valor de los mismos.

DB2GENERAL
Esto significa que el procedimiento almacenado utilizará un parámetro que pasa un convenio que se define para utilizarlo con métodos Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

DB2SQL
Además de los parámetros de la sentencia CALL, al procedimiento almacenado se le envían los argumentos siguientes:

- Un vector que contiene un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento almacenado
- El nombre específico del procedimiento almacenado
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR u OLE.

GENERAL
Significa que el procedimiento almacenado utilizará un mecanismo de envío de parámetros por medio del cual el procedimiento almacenado recibirá los parámetros que se han especificado en la sentencia CALL. Los parámetros se envían directamente, tal como espera el lenguaje; la estructura de la SQLDA no se utiliza. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

Los indicadores nulos *no* se envían directamente al programa.

GENERAL WITH NULLS
Además de los parámetros de la sentencia CALL que se especifica en GENERAL, al procedimiento almacenado se le envía otro argumento. Este argumento adicional es un vector de indicadores nulos, uno para cada parámetro de la sentencia CALL. En C, sería una matriz de enteros cortos. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

JAVA
Significa que el procedimiento almacenado utilizará un convenio de envío de parámetros que se ajusta a la especificación para el lenguaje Java y las Rutinas SQLJ. Los parámetros IN/OUT y OUT se pasarán como matrices de entrada para facilitar los valores de retorno. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

Los procedimientos PARAMETER STYLE JAVA no soportan las cláusulas DBINFO o PROGRAM TYPE.

SQL
Además de los parámetros de la sentencia CALL, al procedimiento almacenado se le envían los argumentos siguientes:

- Un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento almacenado
- El nombre específico del procedimiento almacenado
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR o OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula

PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En los dos casos, cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es 1208.

Si la base de datos no es Unicode y se crea un procedimiento con PARAMETER CCSID UNICODE, el procedimiento no puede tener tipos gráficos ni tipos definidos por el usuario (SQLSTATE 560C1). Los procedimientos PARAMETER CCSID UNICODE sólo se pueden llamar desde un cliente DB2 Versión 8.1 o posterior (SQLSTATE 42997).

Si la base de datos no es Unicode y se ha especificado el orden de clasificación alternativo en la configuración de la base de datos, se pueden crear procedimientos con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos que se pasan al procedimiento o desde él se convertirán en la página de códigos adecuada.

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

PROGRAM TYPE
Especifica si el procedimiento almacenado espera parámetros del tipo de una rutina principal o una subrutina. El valor por omisión es SUB.

SUB
El procedimiento almacenado espera que los parámetros se pasen como argumentos separados.

MAIN
El procedimiento almacenado espera que los parámetros se pasen como un contador de argumentos y como un vector de argumentos (argc, argv). El nombre del procedimiento almacenado que se debe invocar también debe ser ″main″. Los procedimientos almacenados de este tipo deben seguir incorporándose de la misma forma que para una biblioteca compartida, en lugar de como se haría para un ejecutable autónomo. PROGRAM TYPE MAIN sólo es válido cuando la cláusula LANGUAGE especifica C, COBOL o CLR.

DBINFO o **NO DBINFO**
Especifica si se pasa información específica conocida por DB2 al procedimiento almacenado cuando se invoca como argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613). Tampoco recibe soporte para PARAMETER STYLE JAVA o DB2GENERAL.

Si se especifica DBINFO, al procedimiento almacenado se le envía una estructura que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización en tiempo de ejecución de la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de la base de datos que invoca al procedimiento almacenado.
- Plataforma - contiene el tipo de plataforma del servidor.

La estructura DBINFO es común para todas las rutinas externas y contiene campos adicionales que no están relacionados con los procedimientos.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - ASUTIME NO LIMIT
      - COMMIT ON RETURN NO
      - NO COLLID
      - STAY RESIDENT NO
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - RESULT SETS puede especificarse en lugar de DYNAMIC RESULT SETS.
    - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT.
    - DB2GENRL puede especificarse en lugar de DB2GENERAL.
    - SIMPLE CALL puede especificarse en lugar de GENERAL.
    - SIMPLE CALL WITH NULLS puede especificarse en lugar de GENERAL WITH NULLS.
    - PARAMETER STYLE DB2DARI recibe soporte.
- La creación de un procedimiento con un nombre de esquema que todavía no existe dará como resultado la creación implícita de este esquema, siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- Un procedimiento que se llama desde una sentencia compuesta dinámica se ejecutará como si se hubiera creado especificando NEW SAVEPOINT LEVEL, aunque se haya especificado OLD SAVEPOINT LEVEL o se haya tomado por omisión al crear el procedimiento.
- ***Privilegios***

- El usuario que define un procedimiento siempre recibe el privilegio EXECUTE y WITH GRANT OPTION para el procedimiento, así como el derecho para poder eliminar el procedimiento.
- Cuando el procedimiento se utiliza en una sentencia de SQL, el usuario que define el procedimiento debe disponer de privilegio EXECUTE para cualquiera de los paquetes que el procedimiento utiliza.

Ejemplos:

Ejemplo 1: Cree la definición de procedimiento para un procedimiento almacenado, escrito en Java, al que se le pasa un número de pieza y que devuelve el coste de la pieza y la cantidad disponible actualmente.

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
    OUT COST DECIMAL(7,2),
    OUT QUANTITY INTEGER)
EXTERNAL NAME 'parts.onhand'
LANGUAGE JAVA PARAMETER STYLE JAVA
```

Ejemplo 2: Cree la definición de procedimiento para un procedimiento almacenado, escrito en C, al que se le pasa un número de ensamblaje y que devuelve el número de piezas que forman el ensamblaje, el coste total de las piezas y un conjunto de resultados en el que se listan los números de pieza, la cantidad y el coste unitario de cada pieza.

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,
    OUT NUM_PARTS INTEGER,
    OUT COST DOUBLE)
  EXTERNAL NAME 'parts!assembly'
  DYNAMIC RESULT SETS 1 NOT FENCED
  LANGUAGE C PARAMETER STYLE GENERAL
```

Información relacionada:

- "SAVEPOINT" en la página 687
- "Sentencias de SQL que se permiten en rutinas" en la publicación *Consulta de SQL, Volumen 1*
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "spcreate.db2 -- How to catalog the stored procedures contained in spserver.sqc (C)"
- "spcreate.db2 -- Catalog the DB2 CLI stored procedures contained in spserver.c (CLI)"
- "SpCreate.db2 -- How to catalog the stored procedures contained in SpServer.java "
- "SpCreate.db2 -- How to catalog the stored procedures contained in SpServer.sqlj "

CREATE PROCEDURE (SQL)

La sentencia CREATE PROCEDURE (SQL) se utiliza para definir un procedimiento de SQL con un servidor de aplicaciones.

Invocación:

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

Autorización:

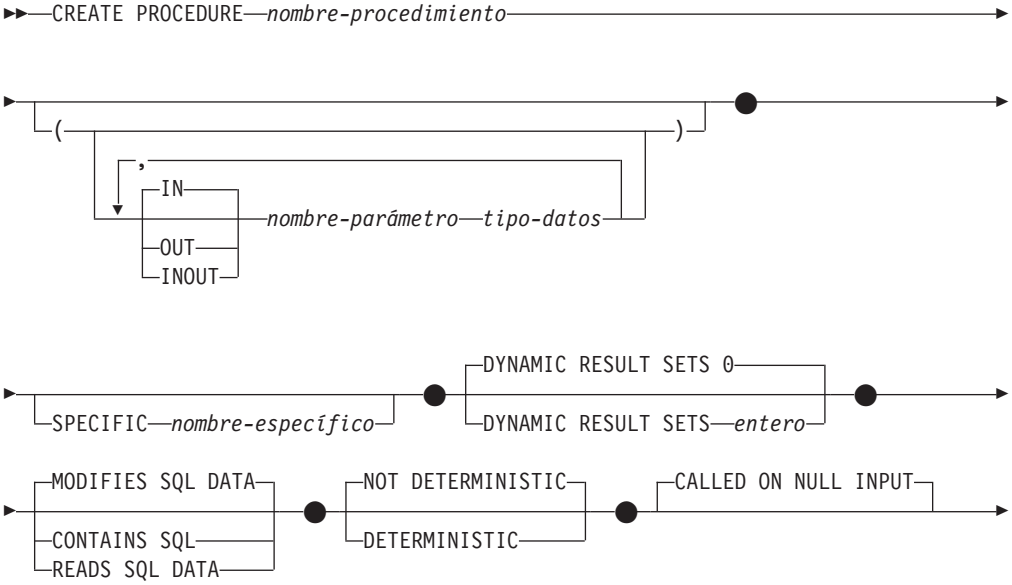
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

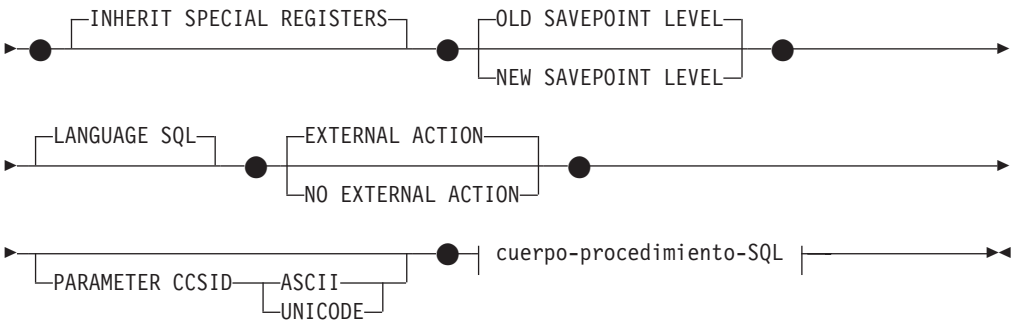
- Autorización SYSADM o DBADM
- Privilegio BINDADD para la base de datos, y uno de los siguientes:
  - Privilegio IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del procedimiento no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente

Si el ID de autorización de la sentencia no dispone de autorización SYSADM o DBADM, los privilegios del ID de autorización de la sentencia también deben incluir todos los privilegios necesarios para invocar las sentencias de SQL que se han especificado en el cuerpo de procedimiento.

Si el ID de autorización no tiene autorización suficiente para realizar la operación, se produce un error (SQLSTATE 42502).

Sintaxis:





cuerpo-procedimiento-SQL:

Descripción:

nombre-procedimiento
Indica el nombre del procedimiento que se está definiendo. Es un nombre calificado o no calificado que designa un procedimiento. La forma no calificada de *nombre-procedimiento* es un identificador de SQL (cuya longitud máxima es 128). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros, no deben identificar un procedimiento que se haya descrito en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número de parámetros, es exclusivo dentro de su esquema, pero no es necesario que sea exclusivo en todos los esquemas.

Si se especifica un nombre compuesto de dos partes, el *nombre-esquema* no puede empezar por 'SYS'; de lo contrario, se devuelve un error (SQLSTATE 42939).

*(***IN** | **OUT** | **INOUT** *nombre-parámetro tipo-datos,...)*
Identifica los parámetros del procedimiento y especifica la modalidad, el nombre y el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el procedimiento va a esperar.

Es posible registrar un procedimiento que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE PROCEDURE SUBWOOFER() ...
```

En un esquema, no se permite que dos procedimientos que se denominen igual tengan exactamente el mismo número de parámetros. Si hay una signatura duplicada se genera un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

la segunda sentencia no se ejecutará satisfactoriamente porque el número de parámetros del procedimiento es el mismo, aunque los tipos de datos no lo sean.

IN | OUT | INOUT
Especifica la modalidad del parámetro.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT
Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro
Especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos
Especifica el tipo de datos del parámetro.

- Pueden especificarse abreviaturas y especificaciones de tipo de datos de SQL, que pueden indicarse en la definición de *tipo-datos* de una sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje que está utilizándose para escribir el procedimiento.
- LONG VARCHAR, LONG VARGRAPHIC, DATALINK, REFERENCE y los tipos estructurados definidos por el usuario no reciben soporte (SQLSTATE 429BB).

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia del procedimiento que se está definiendo. El nombre específico puede utilizarse al eliminar el procedimiento o realizar un comentario en el procedimiento. No puede utilizarse nunca para invocar el procedimiento. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe designar otra instancia del procedimiento que exista en el servidor de aplicaciones; de lo contrario se genera un error (SQLSTATE 42710).

El *nombre-específico* puede ser igual a un *nombre-procedimiento* existente.

Si no se especifica ningún calificador, se utilizará el calificador que se ha utilizado para *nombre-procedimiento*. Si se especifica un calificador, éste deberá ser el mismo que el calificador explícito o implícito del *nombre-procedimiento* o se generará un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo será SQL seguido de una indicación de fecha y hora expresada en forma de caracteres, es decir, SQL*aammddhhmmsshhn*.

DYNAMIC RESULT SETS *entero*
Indica el enlace lógico superior estimado de los conjuntos del resultado devueltos para el procedimiento almacenado.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA
Indica el nivel de acceso a datos para las sentencias de SQL que se han incluido en el procedimiento.

CONTAINS SQL
Indica que el procedimiento almacenado puede ejecutar sentencias de SQL que no lean no modifiquen datos SQL (SQLSTATE 38004 ó 42985). Las sentencias que no pueden utilizarse en ningún procedimiento almacenado devuelven un error diferente (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en el procedimiento almacenado pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no pueden utilizarse en ningún procedimiento almacenado devuelven un error diferente (SQLSTATE 38003 ó 42985).

MODIFIES SQL DATA
Indica que el procedimiento almacenado puede ejecutar cualquier sentencia de SQL excepto aquéllas que no pueden utilizarse en ningún procedimiento almacenado (SQLSTATE 38003 ó 42985).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula especifica si el procedimiento devuelve siempre los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si el procedimiento depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un procedimiento DETERMINISTIC debe siempre devolver el mismo resultado ante invocaciones sucesivas con entradas de datos idénticas.

Actualmente esta cláusula no afecta al proceso del procedimiento almacenado.

CALLED ON NULL INPUT
>CALLED ON NULL INPUT siempre se aplica a procedimientos almacenados. Esto significa que el procedimiento almacenado se llamará, con independencia de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento almacenado comprobar si hay valores argumento nulos.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

OLD SAVEPOINT LEVEL o **NEW SAVEPOINT LEVEL**
Especifica si este procedimiento almacenado establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información acerca de los niveles de punto de salvaguarda, consulte la sección “Normas” de la descripción de la sentencia SAVEPOINT.

LANGUAGE SQL
Esta cláusula se utiliza para especificar que el cuerpo de procedimiento se ha escrito en el lenguaje SQL.

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

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE
Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

cuerpo-procedimiento-SQL
Especifica la sentencia de SQL que forma el cuerpo del procedimiento SQL. Se pueden especificar varias sentencias de procedimiento de SQL dentro de una sentencia compuesta de procedimiento. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

Normas:

- Un procedimiento que se llama desde una sentencia compuesta dinámica se ejecutará como si se hubiera creado especificando NEW SAVEPOINT LEVEL, aunque se haya especificado OLD SAVEPOINT LEVEL o se haya tomado por omisión al crear el procedimiento.

Notas:

- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - ASUTIME NO LIMIT
      - COMMIT ON RETURN NO
      - NO COLLID
      - STAY RESIDENT NO
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - RESULT SETS puede especificarse en lugar de DYNAMIC RESULT SETS.
    - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT.
  - La creación de un procedimiento con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema, siempre que el ID de autorización de la sentencia disponga de autorización

IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

– ***Privilegios***

El usuario que define un procedimiento siempre recibe el privilegio EXECUTE y WITH GRANT OPTION para el procedimiento, así como el derecho para poder eliminar el procedimiento.

Ejemplos:

Ejemplo 1: Cree un procedimiento de SQL que devuelva el salario medio de la plantilla de trabajadores. Se obtendrá un conjunto de resultados que contiene el nombre, el puesto de trabajo y el salario de todos los empleados que ganan más que el salario medio.

```
CREATE PROCEDURE MEDIAN_RESULT_SET (OUT medianSalary DOUBLE)
  RESULT SETS 1
  LANGUAGE SQL
BEGIN
  DECLARE v_numRecords INT DEFAULT 1;
  DECLARE v_counter INT DEFAULT 0;

  DECLARE c1 CURSOR FOR
    SELECT CAST(salary AS DOUBLE)
      FROM staff
      ORDER BY salary;
  DECLARE c2 CURSOR WITH RETURN FOR
    SELECT name, job, CAST(salary AS INTEGER)
              FROM staff
           WHERE salary > medianSalary
      ORDER BY salary;

  DECLARE EXIT HANDLER FOR NOT FOUND
    SET medianSalary = 6666;

  SET medianSalary = 0;
  SELECT COUNT(*) INTO v_numRecords
    FROM STAFF;
  OPEN c1;
  WHILE v_counter < (v_numRecords / 2 + 1)
  DO
    FETCH c1 INTO medianSalary;
    SET v_counter = v_counter + 1;
  END WHILE;
  CLOSE c1;
  OPEN c2;
END
```

Información relacionada:

- "SAVEPOINT" en la página 687
- "SQL compuesto (procedimiento)" en la página 139
- "Sentencias de SQL que se permiten en rutinas" en la publicación *Consulta de SQL, Volumen 1*
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "basecase.db2 -- To create the UPDATE_SALARY SQL procedure "
- "nestcase.db2 -- To create the BUMP_SALARY SQL procedure "
- "nestedsp.db2 -- To create the OUT_AVERAGE, OUT_MEDIAN and MAX_SALARY SQL procedures"

- "rsultset.db2 -- To register and create the MEDIAN_RESULT_SET SQL procedure"

CREATE SCHEMA

La sentencia CREATE SCHEMA define un esquema. También es posible crear algunos objetos y otorgar privilegios en los objetos dentro de la sentencia.

Invocación:

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

Autorización:

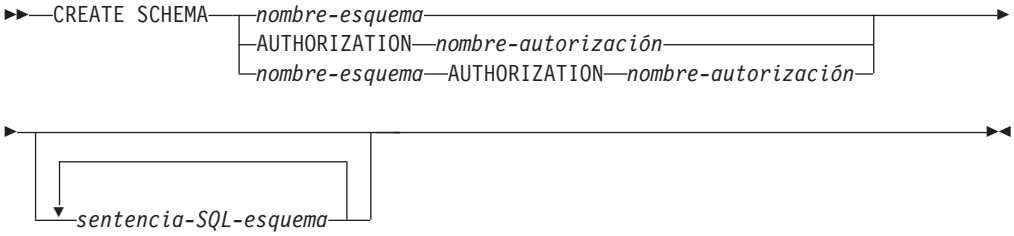
Un ID de autorización que tenga la autorización SYSADM o DBADM puede crear un esquema con cualquier *nombre-esquema* o *nombre-autorización* válido.

Un ID de autorización que no tenga la autorización SYSADM o DBADM sólo puede crear un esquema con un *nombre-esquema* o *nombre-autorización* que coincida con el ID de autorización de la sentencia.

Si la sentencia incluye cualquier *sentencia-SQL-esquema* los privilegios que tiene el *nombre-autorización* (si no se especifica, toma por omisión el ID de autorización de la sentencia) debe incluir como mínimo uno de los siguientes:

- Los privilegios necesarios para ejecutar cada *sentencia-SQL-esquema*
- Autorización SYSADM o DBADM.

Sintaxis:



Descripción:

nombre-esquema
Indica el nombre del esquema. El nombre no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710). El nombre no puede empezar por 'SYS' (SQLSTATE 42939). El propietario del esquema es el ID de autorización que ha emitido la sentencia.

AUTHORIZATION *nombre-autorización*
Identifica el usuario que es el propietario del esquema. El valor de *nombre-autorización* también se utiliza para denominar el esquema. El *nombre-autorización* no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710).

nombre-esquema **AUTHORIZATION** *nombre-autorización*
Identifica un esquema denominado *nombre-esquema*, cuyo propietario es *nombre-autorización*. El *nombre-esquema* no debe identificar un esquema que ya esté descrito en el catálogo (SQLSTATE 42710). El *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

sentencia-SQL-esquema
Las sentencias de SQL que se pueden incluir como parte de la sentencia CREATE SCHEMA son:

- Sentencia CREATE TABLE, excluyendo las tablas con tipo y las tablas de consultas materializadas
- Sentencia CREATE VIEW, excluyendo las vistas con tipo
- Sentencia CREATE INDEX
- Sentencia COMMENT
- Sentencia GRANT

Notas:

- El propietario del esquema se determina de la manera siguiente:
  - Si se especifica la cláusula AUTHORIZATION, el *nombre-autorización* especificado es el propietario del esquema
  - Si no se especifica la cláusula AUTHORIZATION, el ID de autorización que emite la sentencia CREATE SCHEMA es el propietario del esquema.
- Se supone que el propietario del esquema es un usuario (no un grupo).
- Cuando se crea explícitamente el esquema con la sentencia CREATE SCHEMA, se otorga al propietario del esquema los privilegios CREATEIN, DROPIN y ALTERIN en el esquema con la posibilidad de otorgar estos privilegios a otros usuarios.
- La persona que define cualquier objeto creado como parte de la sentencia CREATE SCHEMA es el propietario del esquema. El propietario del esquema es el otorgante de cualquier privilegio que se otorga como parte de la sentencia CREATE SCHEMA.
- Los nombres de objeto no calificados en ninguna sentencia de SQL dentro de la sentencia CREATE SCHEMA se califican implícitamente por el nombre del esquema creado.
- Si la sentencia CREATE contiene un nombre calificado para el objeto que se está creando, el nombre de esquema especificado en el nombre calificado debe ser el mismo que el nombre del esquema que se está creando (SQLSTATE 42875). Cualquier otro objeto que se haga referencia en las sentencias pueden calificarse con un nombre de esquema válido.
- Es recomendable no utilizar ″SESSION″ como nombre de esquema. Debido a que las tablas temporales declaradas deben estar calificadas por ″SESSION″, es posible que una aplicación declare una tabla temporal que tenga el mismo nombre que el de una tabla permanente. Una tabla incluida en una sentencia de SQL y que tiene el nombre de esquema ″SESSION″ se convierte (al compilarse la sentencia) en la tabla temporal declarada, y no en la tabla permanente del mismo nombre. Debido a que las sentencias estáticas y dinámicas del SQL intercalado se compilan en momentos diferentes, los resultados dependen del momento en que se define la tabla temporal declarada. Estas cuestiones no se plantean cuando no se utiliza el nombre de esquema ″SESSION″ para definir una tabla permanente, vista o alias.

Ejemplos:

Ejemplo 1: Como usuario con autorización DBADM, cree un esquema llamado RICK con el usuario RICK como el propietario.

```
CREATE SCHEMA RICK AUTHORIZATION RICK
```

Ejemplo 2: Cree un esquema que tenga una tabla de piezas del inventario y un índice de los números de piezas. Otorgue la autorización sobre la tabla al usuario JONES.

```
CREATE SCHEMA INVENTRY

  CREATE TABLE PART (PARTNO   SMALLINT NOT NULL,
                     DESCR    VARCHAR(24),
                     QUANTITY INTEGER)

  CREATE INDEX PARTIND ON PART (PARTNO)

  GRANT ALL ON PART TO JONES
```

Ejemplo 3: Cree un esquema llamado PERS con dos tablas que cada una tenga una clave foránea que haga referencia a otra tabla. Es un ejemplo de una característica de la sentencia CREATE SCHEMA que permite la creación de un par de tablas como éstas sin la utilización de la sentencia ALTER TABLE.

```
CREATE SCHEMA PERS

  CREATE TABLE ORG (DEPTNUMB  SMALLINT NOT NULL,
                    DEPTNAME VARCHAR(14),
                    MANAGER  SMALLINT,
                    DIVISION VARCHAR(10),
                    LOCATION VARCHAR(13),
                    CONSTRAINT PKEYDNO
                       PRIMARY KEY (DEPTNUMB),
                    CONSTRAINT FKEYMGR
                       FOREIGN KEY (MANAGER)
                       REFERENCES STAFF (ID) )

  CREATE TABLE STAFF (ID        SMALLINT NOT NULL,
                      NAME      VARCHAR(9),
                      DEPT      SMALLINT,
                      JOB       VARCHAR(5),
                      YEARS     SMALLINT,
                      SALARY    DECIMAL(7,2),
                      COMM      DECIMAL(7,2),
                      CONSTRAINT PKEYID
                         PRIMARY KEY (ID),
                      CONSTRAINT FKEYDNO
                         FOREIGN KEY (DEPT)
                         REFERENCES ORG (DEPTNUMB) )
```

Información relacionada:

- "COMMENT" en la página 118
- "CREATE INDEX" en la página 273
- "CREATE TABLE" en la página 341
- "CREATE VIEW" en la página 466
- "GRANT (Privilegios de tabla, vista o apodo)" en la página 592

CREATE SEQUENCE

La sentencia CREATE SEQUENCE crea una secuencia en el servidor de aplicaciones.

Invocación:

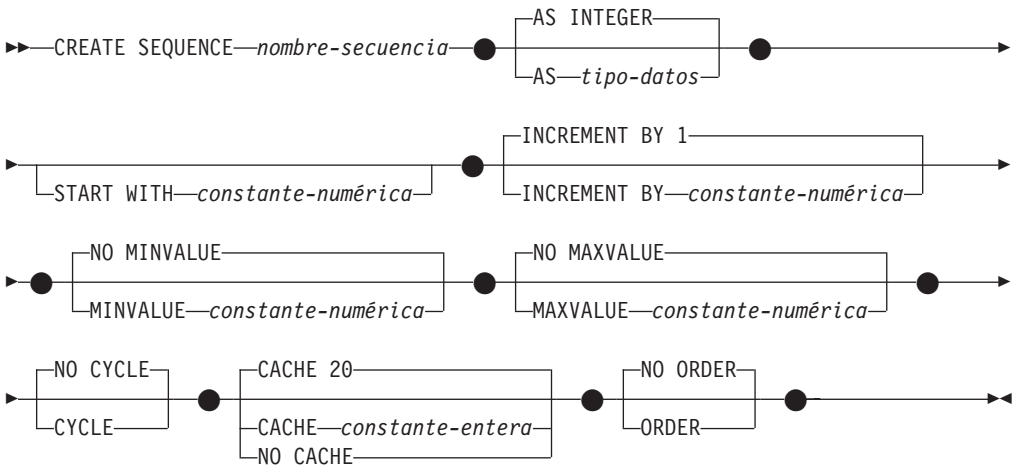
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la secuencia no existe
- Privilegio CREATEIN si el nombre de esquema de la secuencia hace referencia a un esquema existente

Sintaxis:



Descripción:

nombre-secuencia
Indica el nombre de la secuencia. La combinación del nombre y del nombre de esquema implícito o explícito no debe identificar una secuencia existen en el servidor actual (SQLSTATE 42710).

El formato no calificado de un nombre-secuencia es un identificador SQL. El formato calificado es un calificador seguido de un punto y un identificador SQL. El calificador es un nombre de esquema.

Si el nombre de secuencia se califica explícitamente con un nombre de esquema, el nombre de esquema no puede empezar con 'SYS' o se producirá un error (SQLSTATE 42939).

AS *tipo-datos*
Especifica el tipo de datos que se debe utilizar para el valor de secuencia. El

tipo de datos puede ser cualquier tipo numérico exacto (SMALLINT, INTEGER, BIGINT o DECIMAL) con una escala de cero o un tipo diferenciado o tipo de referencia definido por el usuario para el cual el tipo de fuente sea un tipo numérico exacto con una escala de cero (SQLSTATE 42815). El valor por omisión es INTEGER.

START WITH *constante-numérica*
Especifica el primer valor de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). El valor por omisión es MINVALUE para secuencias ascendentes y MAXVALUE para secuencias descendentes.

Este valor no es necesariamente el valor en el que una secuencia realizaría un ciclo después de alcanzar el valor máximo o mínimo de la secuencia. Se puede utilizar la cláusula START WITH para iniciar una secuencia fuera del rango que se usa para los ciclos. El rango utilizado para los ciclos se define mediante MINVALUE y MAXVALUE.

INCREMENT BY *constante-numérica*
Especifica el intervalo entre valores consecutivos de la secuencia. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820) y sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente. Si este valor es 0 o positivo, es una secuencia ascendente. El valor por omisión es 1.

MINVALUE o NO MINVALUE
Especifica el valor mínimo en el que una secuencia descendente pasa por un ciclo o deja de generar valores o en el que una secuencia ascendente pasa por un ciclo después de alcanzar el valor máximo.

MINVALUE *constante-numérica*
Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor que o igual al valor máximo (SQLSTATE 42815).

NO MINVALUE
Para una secuencia ascendente, el valor es el valor START WITH o bien 1 si no se ha especificado START WITH. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos asociado con la secuencia. Este es el valor por omisión.

MAXVALUE o NO MAXVALUE
Especifica el valor máximo en el que una secuencia ascendente pasa por un ciclo o deja de generar valores o en el que una secuencia descendente pasa por un ciclo después de alcanzar el valor mínimo.

MAXVALUE *constante-numérica*
Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos que se asocia a la secuencia (SQLSTATE 42815), sin

dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser mayor que o igual al valor mínimo (SQLSTATE 42815).

NO MAXVALUE
Para una secuencia ascendente, el valor es el valor máximo del tipo de datos asociado con la secuencia. Para una secuencia descendente, el valor es el valor START WITH o bien -1 si no se ha especificado START WITH.

CYCLE o NO CYCLE
Especifica si la secuencia debe continuar generando valores después de alcanzar el valor máximo o el valor mínimo. El límite de la secuencia puede alcanzarse con el siguiente valor que coincide exactamente con la condición de límite o excediendo el valor.

CYCLE
Especifica que se continúan generando valores para esta secuencia después de haber alcanzado el valor máximo o mínimo. Si se utiliza esta opción, cuando una secuencia ascendente haya alcanzado su valor máximo, generará su valor mínimo o cuando una secuencia descendente haya alcanzado su valor mínimo, generará su valor máximo. Los valores máximo y mínimo para la secuencia determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, se pueden generar valores duplicados para la secuencia.

NO CYCLE
Especifica que no se generarán valores para la secuencia una vez que se haya alcanzado el valor máximo o mínimo para la secuencia. Este es el valor por omisión.

CACHE o NO CACHE
Especifica si se deben mantener algunos valores preasignados en memoria para obtener un acceso más rápido. Esta opción se utiliza para el rendimiento y el ajuste.

CACHE *constante-entera*
Especifica el número máximo de valores de secuencia que se preasignan y se mantienen en memoria. La preasignación y el almacenamiento de valores en la antememoria reducen la E/S síncrona en las anotaciones cronológicas cuando se generan valores para la secuencia.

En el caso de producirse una anomalía del sistema, todos los valores de secuencia almacenados en antememoria que no se han utilizando en sentencias confirmadas se pierden (es decir, no se utilizarán nunca). El valor especificado para la opción CACHE es el número máximo de valores de secuencia que puede perderse en caso de anomalía del sistema.

El valor mínimo es 2 (SQLSTATE 42815). El valor por omisión es CACHE 20.

NO CACHE
Especifica que los valores de la secuencia no se deben preasignar. Asegura que no haya ninguna pérdida de valores en el caso de producirse una anomalía del sistema, un cierre o una desactivación de la base de datos. Cuando se especifica esta opción, los valores de la secuencia no se almacenan en la antememoria. En este caso, cada petición de un valor nuevo para la secuencia produce E/S síncrona en las anotaciones cronológicas.

NO ORDER o ORDER
Especifica si los números de secuencia deben generarse según el orden de petición.

ORDER
Especifica que los números de secuencia se generan según el orden de petición.

NO ORDER
Especifica que los números de secuencia no necesitan generarse según el orden de petición. Este es el valor por omisión.

Notas:

- *Compatibilidades*
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - Puede utilizarse una coma para separar varias opciones en la secuencia.
  - También recibe soporte la sintaxis siguiente:
    - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER.
- Es posible definir una secuencia constante, es decir, una que devuelva siempre un valor constante. Esto puede hacerse especificando un valor cero para INCREMENT y un valor para START WITH que no exceda el valor de MAXVALUE o bien especificando el mismo valor para START WITH, MINVALUE y MAXVALUE. Para una secuencia constante, cada vez que se invoque NEXT VALUE para la secuencia, se devolverá el mismo valor. Una secuencia constante puede utilizarse como variable global numérica. ALTER SEQUENCE puede utilizarse para ajustar los valores que se generarán para una secuencia constante.
- El ciclo de una secuencia puede especificarse manualmente utilizando la sentencia ALTER SEQUENCE. Si se especifica NO CYCLE de forma implícita o explícita, se puede reiniciar o ampliar la secuencia utilizando la sentencia ALTER SEQUENCE para hacer que los valores continúen generándose una vez que se haya alcanzado el valor máximo o mínimo para la secuencia.
- Una secuencia puede definirse explícitamente para que ejecute un ciclo especificando la palabra clave CYCLE. Al definir una secuencia, utilice la opción CYCLE para indicar que los valores generados deben ejecutar un ciclo cuando se haya alcanzado el límite. Cuando se ha definido una secuencia para la ejecución automática de un ciclo (es decir, CYCLE se ha especificado explícitamente), el valor máximo o mínimo generado para una secuencia puede que no sea el valor MAXVALUE o MINVALUE real que se ha especificado, si el incremento es un valor distinto de 1 ó -1. Por ejemplo, la secuencia que se ha definido con `START WITH=1, INCREMENT=2, MAXVALUE=10` generará un valor máximo de 9 y no generará el valor 10. Al definir una secuencia con CYCLE, considere detenidamente el efecto que pueden tener los valores para MINVALUE, MAXVALUE y START WITH.
- El almacenamiento en antememoria de los números de secuencia implica que un rango de números de secuencia puede mantenerse en la memoria para acceder a ellos rápidamente. Cuando una aplicación accede a una secuencia que puede asignar el siguiente número de secuencia desde la antememoria, la asignación de números de secuencia se produce rápidamente. Sin embargo, si una aplicación accede a una secuencia que no puede asignar el siguiente número de secuencia desde la antememoria, puede que para asignar el número de secuencia sea necesario esperar a que se realicen las operaciones de E/S en el almacenamiento permanente. La elección del valor para CACHE debe realizarse teniendo en cuenta los cambios de requisitos del rendimiento y de la aplicación.

- Al usuario que define una secuencia se le otorgan los privilegios ALTER y USAGE y el privilegio WITH GRANT OPTION. El usuario que define una secuencia también puede eliminar la secuencia.

Ejemplos:

Ejemplo 1: Cree una secuencia denominada ORG_SEQ que empiece por el 1, que se ejecute en incrementos de 1, que no ejecute un ciclo y que coloque en antememoria 24 valores al mismo tiempo:

```
CREATE SEQUENCE ORG_SEQ
   START WITH 1
   INCREMENT BY 1
   NO MAXVALUE
   NO CYCLE
  CACHE 24
```

Ejemplos relacionados:

- "DbSeq.java -- How to create, alter and drop a sequence in a database (JDBC)"

CREATE SERVER

La sentencia CREATE SERVER define una fuente de datos para una base de datos federada. En esta sentencia, el término SERVER y los nombres de parámetro que empiezan por *server-* sólo hacen referencia a fuentes de datos de un sistema federado. No hacen referencia al servidor federado de ese sistema ni a los servidores de aplicaciones DRDA.

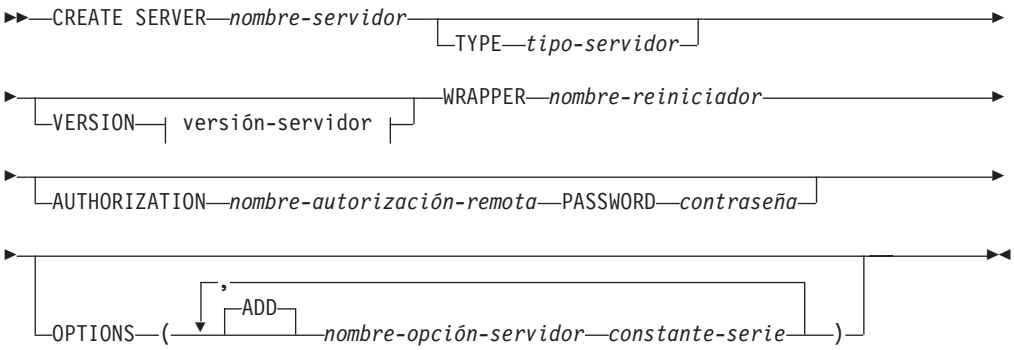
Invocación:

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

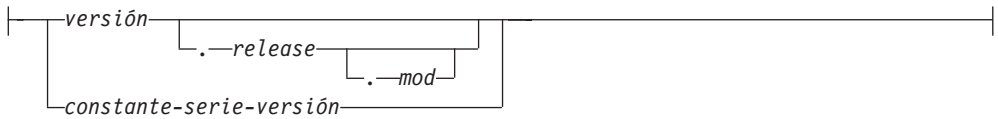
Autorización:

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

Sintaxis:



versión-servidor:



Descripción:

nombre-servidor
Nombra la fuente de datos que se está definiendo en la base de datos federada. El nombre no debe identificar una fuente de datos que esté descrita en el catálogo. El *nombre-servidor* no debe ser el mismo que el nombre de cualquier espacio de tablas de la base de datos federada.

Normalmente, una definición de servidor para fuentes de datos relacionales representa una base de datos remota. Algunos sistemas de gestión de base de datos relacionales como, por ejemplo, Oracle, no admiten múltiples bases de datos en cada instancia. En su lugar, cada instancia representa un servidor en un sistema federado.

Para las fuentes de datos no relacionales, el propósito de una definición de servidor varía según la fuente de datos. Algunas definiciones de servidor se correlacionan con un tipo de búsqueda y daemon, un sitio Web o un servidor Web. Para otras fuentes de datos no relacionales, se crea una definición de servidor porque la jerarquía de objetos federados requiere que los archivos de fuente de datos (identificados por apodos) se asocien a un objeto de servidor específico.

TYPE *tipo-servidor*
Especifica el tipo de fuente de datos que *nombre-servidor* indica. Este parámetro es necesario para algunos reiniciadores.

VERSION
Especifica la versión de la fuente de datos indicada por *nombre-servidor*. Este parámetro es necesario para algunos reiniciadores.

versión
Especifica el número de versión. El valor debe ser un entero.

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

mod
Especifica el número de la modificación del release indicado por *release*. El valor debe ser un entero.

constante-serie-versión
Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i') o puede estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*
Nombra el reiniciador que el servidor federado utiliza para interactuar con el objeto de servidor especificado por *nombre-servidor*.

AUTHORIZATION *nombre-autorización-remota*
Sólo se requiere para las fuentes de datos de la familia DB2. Especifica el ID de autorización bajo el cual se realiza cualquier acción necesaria en la fuente de datos cuando se procesa la sentencia CREATE SERVER. Este ID debe tener la autorización (BINDADD o su equivalente) que las acciones necesarias necesitan. Si se especifica el *nombre-autorización-remota* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota contiene nombres de autorización sensibles a las mayúsculas y minúsculas), el *nombre-autorización-remota* se debe especificar entre comillas dobles.

PASSWORD *contraseña*
Sólo se requiere para las fuentes de datos de la familia DB2. Especifica la contraseña asociada al ID de autorización representado por *nombre-autorización-remota.* Si se especifica la *contraseña* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota tiene contraseñas sensibles a las mayúsculas y minúsculas), la *contraseña* se debe especificar entre comillas dobles.

OPTIONS
Indica las opciones que se habilitan cuando se crea la definición de servidor. Las opciones de servidor se utilizan para configurar la definición de servidor. Algunas opciones de servidor pueden utilizarse para crear la definición de

servidor para cualquiera de las definiciones de servidor de fuente de datos. Algunas opciones de servidor son específicas para una fuente de datos en particular.

ADD
Habilita una o varias opciones de servidor.

nombre-opción-servidor
Nombra una opción de servidor que se utilizará para configurar o proporcionar información acerca de la fuente de datos indicada por *nombre-servidor*.

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

Notas:

- La *contraseña* se debe especificar cuando la fuente de datos requiere una contraseña. Si las letras de una *contraseña* deben ir en minúsculas, especifique la *contraseña* entre comillas.
- Si se utiliza la sentencia CREATE SERVER para definir una instancia de la familia de DB2 como fuente de datos, puede que DB2 deba vincular determinados paquetes con esa instancia. Si es necesario una vinculación, el *nombre-autorización-remota* de la sentencia debe tener la autorización BIND. El tiempo necesario para que se complete la operación de vinculación depende de la velocidad de la fuente de datos y de la velocidad de conexión de la red.

Ejemplos:

Ejemplo 1: Registre una definición de servidor para acceder a una fuente de datos DB2 para z/OS y OS/390, Versión 7.1. CRANDALL es el nombre asignado a la definición de servidor DB2 para z/OS y OS/390. DRDA es el nombre del reiniciador utilizado para acceder a esta fuente de datos. Además, especifique que:

- GERALD y drowssap son el ID de autorización y la contraseña bajo los cuales se vinculan los paquetes en CRANDALL cuando se procesa esta sentencia.
- El alias para la base de datos DB2 para z/OS y OS/390 que se ha especificado con la sentencia CATALOG DATABASE es CLIENTS390.
- Los ID de autorización y las contraseñas bajo los que se puede acceder a CRANDALL se deben enviar a CRANDALL en mayúsculas.
- CLIENTS390 y la base de datos federada utilizan el mismo orden de clasificación.

```
CREATE SERVER CRANDALL
  TYPE DB2/ZOS
  VERSION 7.1
  WRAPPER DRDA
  AUTHORIZATION "GERALD"
  PASSWORD drowssap
    OPTIONS
      (DBNAME 'CLIENTS390',
       FOLD_ID 'U',
       FOLD_PW 'U',
      COLLATING_SEQUENCE 'Y')
```

Ejemplo 2: Registre una definición de servidor para acceder a una fuente de datos Oracle 9. CUSTOMERS es el nombre asignado a la definición de servidor Oracle. NET8 es el nombre del reiniciador utilizado para acceder a esta fuente de datos. Además, especifique que:

- ABC es el nombre del nodo en el que reside el servidor de bases de datos Oracle.
- La CPU para el servidor federado se ejecuta el doble de rápido que la CPU que soporta CUSTOMERS.
- Los dispositivos de E/S del servidor federado procesan los datos 1,5 veces más rápido que los dispositivos de E/S de CUSTOMERS.

```
CREATE SERVER CUSTOMERS
    TYPE ORACLE
  VERSION 9
  WRAPPER NET8
    OPTIONS
      (NODE 'ABC',
       CPU_RATIO '2.0',
      IO_RATIO '1.5')
```

Ejemplo 3: Registre una definición de servidor para el reiniciador Excel. La definición de servidor es necesaria para conservar la jerarquía de objetos federados. BIOCHEM_LAB es el nombre asignado a la definición de servidor Excel. EXCEL_2000_WRAPPER es el nombre del reiniciador utilizado para acceder a esta fuente de datos.

```
CREATE SERVER BIOCHEM_DATA
  WRAPPER EXCEL_2000_WRAPPER
```

Ejemplo 4: Registre una definición de servidor para acceder a una fuente de datos BLAST. BLAST_SERVER es el nombre asignado a la definición de servidor BLAST. El tipo de búsqueda a la que esta definición de servidor da soporte es el tipo de búsqueda BLASTn. VERSION es la versión del programa de búsqueda BLAST. BLAST_WRAPPER es el nombre del reiniciador utilizado para acceder a esta fuente de datos. Además, especifique que:

- NODE es el nombre de sistema principal del servidor en el que se ejecuta el proceso de daemon BLAST.
- El número de puerto en el que el daemon BLAST escucha las peticiones de trabajo sometidas por el reiniciador BLAST es 4007.

```
CREATE SERVER BLAST_SERVER
  TYPE BLASTn
  VERSION 2.1.2
  WRAPPER BLAST_WRAPPER
    OPTIONS
      (NODE 'big.rs.company.com',
     DAEMON_PORT '4007')
```

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*
- "Server definitions and server options" en la publicación *Federated Systems Guide*

Información relacionada:

- "Server options for federated systems" en la publicación *Federated Systems Guide*
- "Valid server types in SQL statements" en la publicación *Federated Systems Guide*

CREATE TABLE

La sentencia CREATE TABLE define una tabla. Esta definición debe incluir el nombre de la tabla y los nombres y atributos de sus columnas. La definición puede incluir otros atributos de la tabla, como su clave primaria o restricciones de comprobación.

Para declarar una tabla temporal global, utilice la sentencia DECLARE GLOBAL TEMPORARY TABLE.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Autorización CREATETAB para la base de datos y privilegio USE para el espacio de tablas, y también uno de los elementos siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla hace referencia a un esquema existente.

  Si se define una subtabla, el ID de autorización debe ser el mismo que el definidor de la tabla raíz de la jerarquía de tablas.

Para definir una clave foránea, los privilegios del ID de autorización de la sentencia deben incluir uno de los siguientes en la tabla padre:

- Privilegio REFERENCES para la tabla
- Privilegio REFERENCES para todas las columnas de la clave padre especificada
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

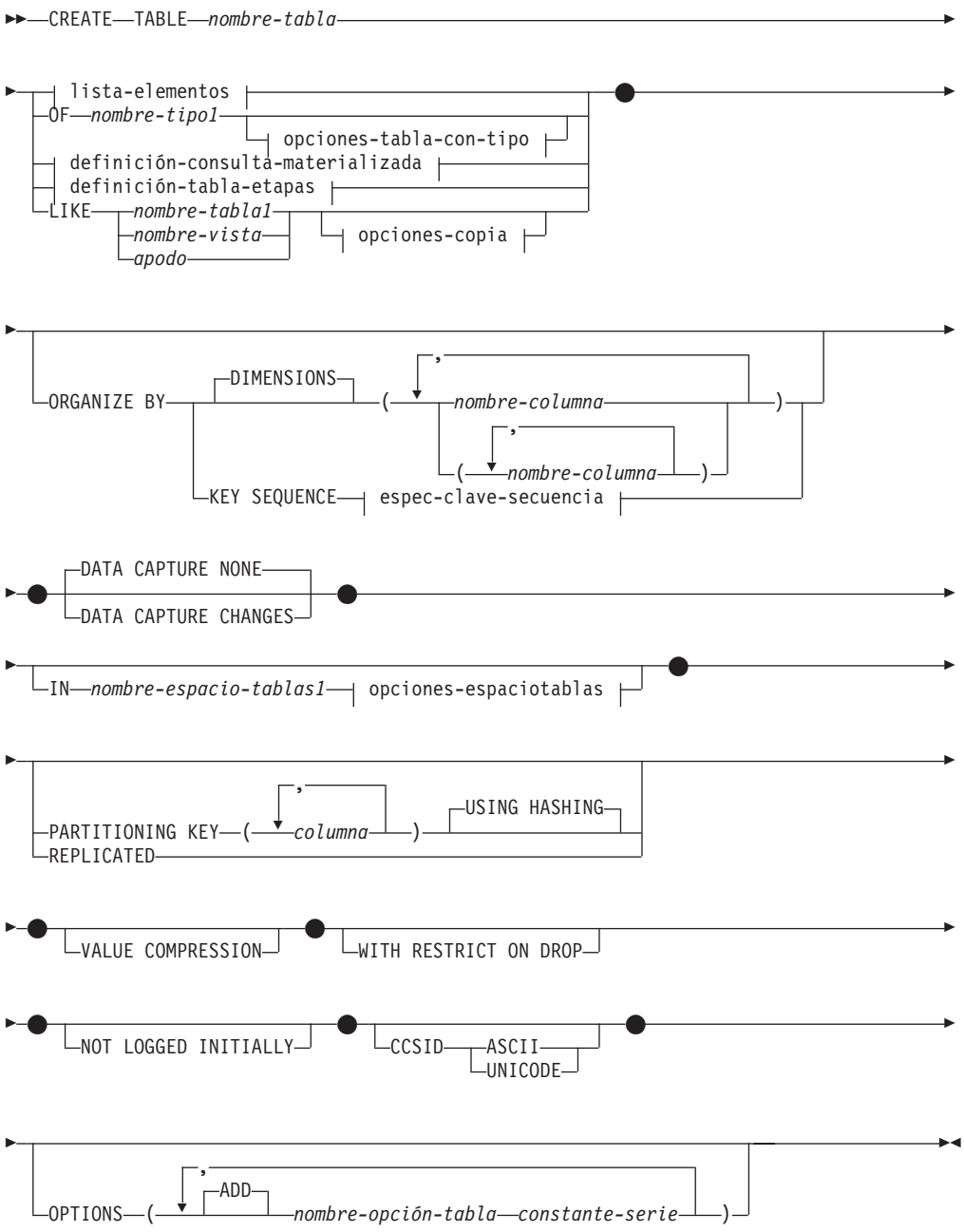
Para definir una tabla de consultas materializadas (utilizando una selección completa), los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes elementos en cada tabla o vista identificada en la selección completa:

- Privilegio SELECT para la tabla o vista y el privilegio ALTER si se especifica REFRESH DEFERRED o REFRESH IMMEDIATE
- Privilegio CONTROL para la tabla o vista
- Autorización SYSADM o DBADM.

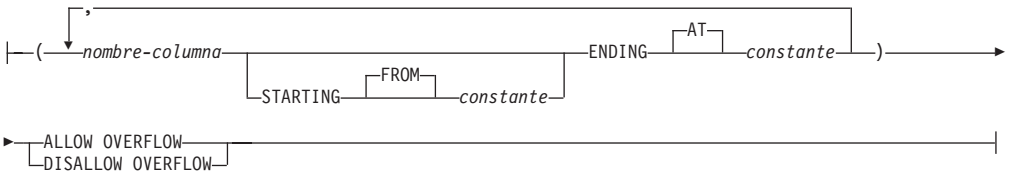
Para definir una tabla de etapas asociada a una tabla de consultas materializadas, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

- Privilegio CONTROL o privilegio ALTER para la tabla de consultas materializadas y, como mínimo, uno de los siguientes para cada tabla o vista que se identifique en la selección completa de la tabla de consultas materializadas:
  - Privilegio SELECT y privilegio ALTER para la tabla o vista
  - Privilegio CONTROL para la tabla o vista
- Autorización SYSADM o DBADM

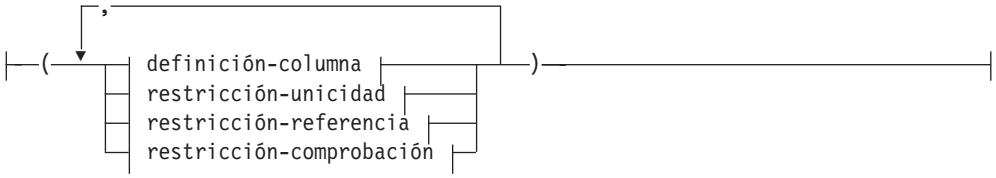
Sintaxis:



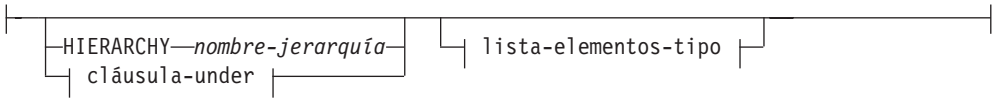
espec-clave-secuencia:



lista-elementos:

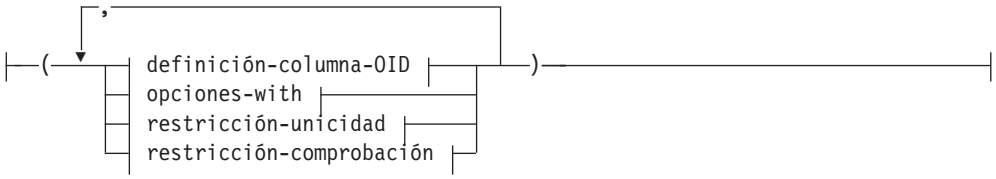


opciones-tabla-con-tipo:

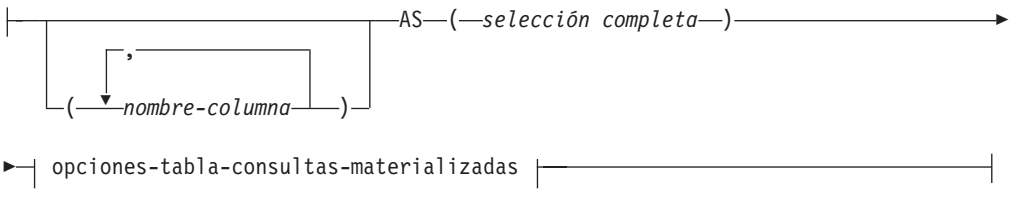


cláusula-under:

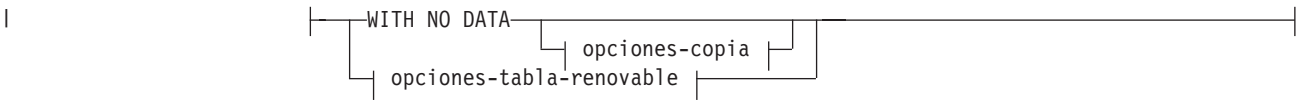
lista-elementos-tipo:



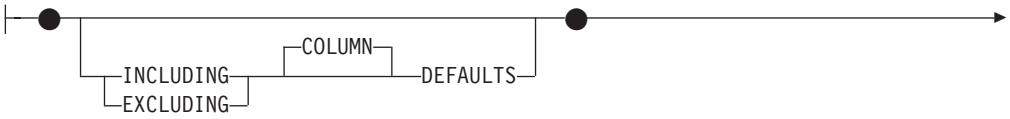
definición-consulta-materializada:

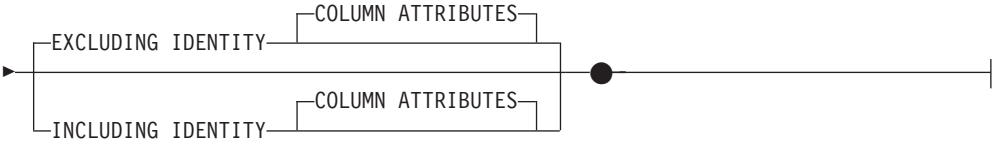


opciones-tabla-consultas-materializadas:

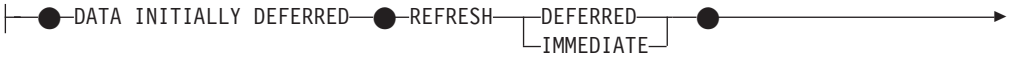


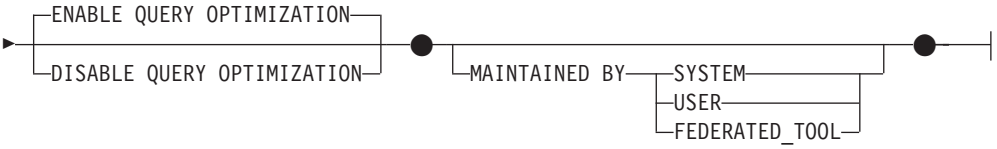
opciones-copia:



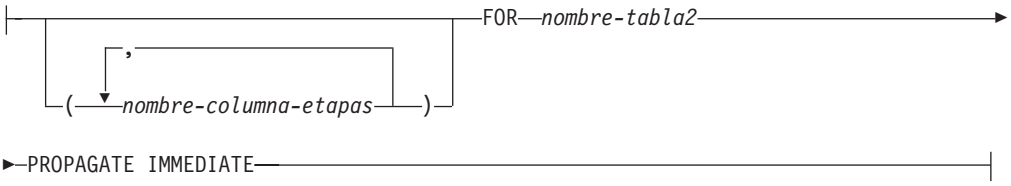


opciones-tabla-renovable:

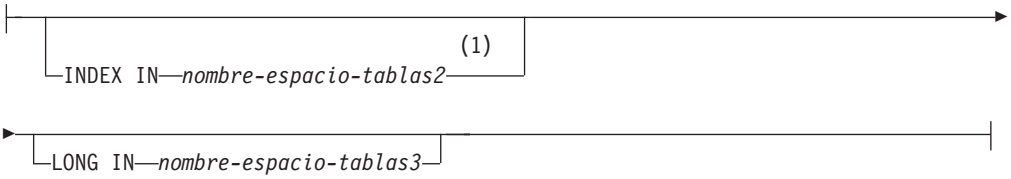




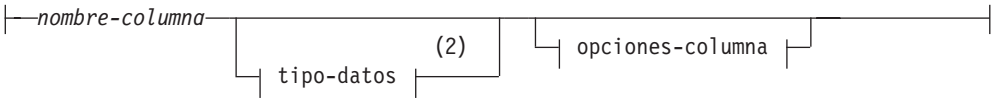
definición-tabla-etapas:



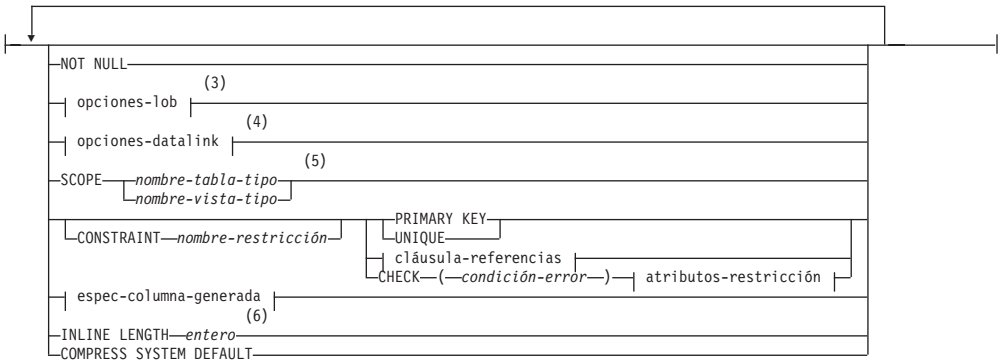
opciones-espaciotablas:



definición-columna:



opciones-columna:



Notas:

1 Sólo se puede especificar el espacio de tablas que contendrá un índice de tabla cuando se crea la tabla.

2 Si la primera opción-columna elegida es una especificación-columna-generada, el tipo-datos puede omitirse. Éste se determinará a partir del tipo de datos resultante de la expresión de generación.

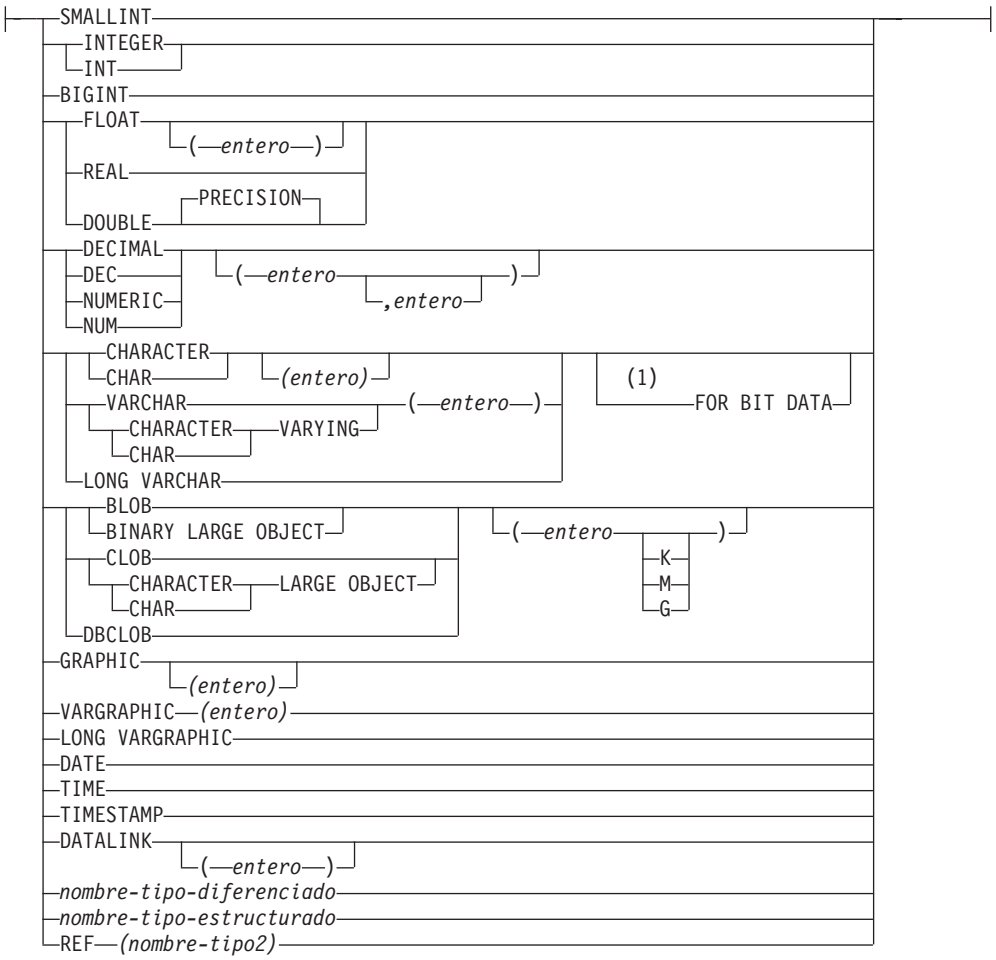
3 La cláusula opciones-lob sólo se aplica a tipos de gran objeto (BLOB, CLOB y DBCLOB) y a los tipos diferenciados basados en tipos de gran objeto.

4 La cláusula opciones-datalink sólo se aplica al tipo DATALINK y a los tipos diferenciados basados en el tipo DATALINK.

5 La cláusula SCOPE sólo se aplica al tipo REF.

6 INLINE LENGTH sólo es aplicable a columnas definidas como tipos estructurados.

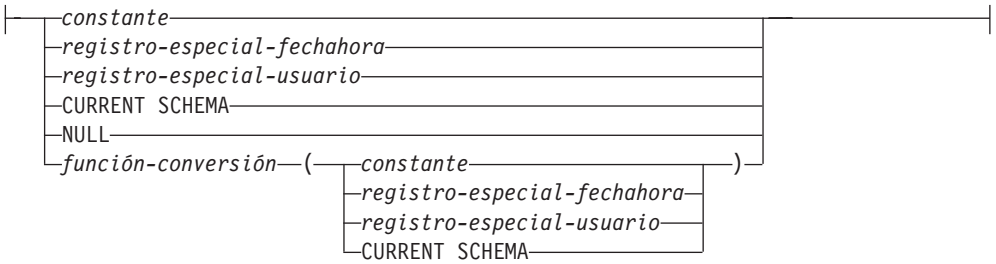
tipo-datos:



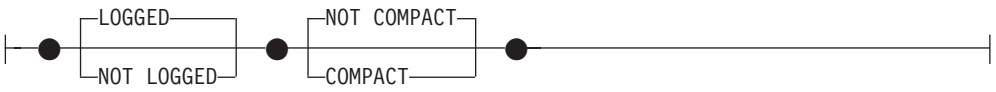
Notas:

1 La cláusula FOR BIT DATA se puede especificar en cualquier orden con respecto a las restricciones de columna siguientes.

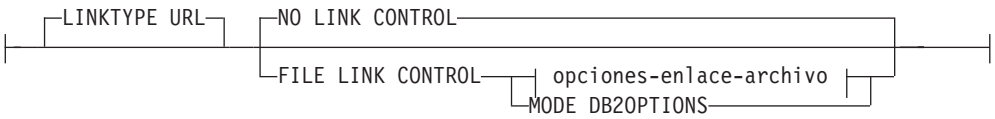
valores-omisión:



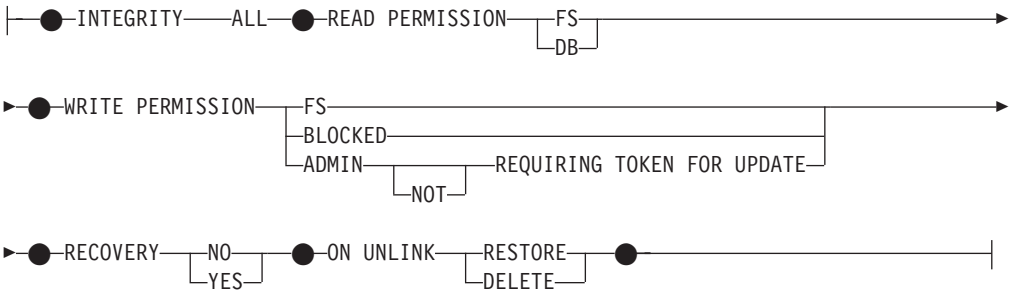
opciones-lob:



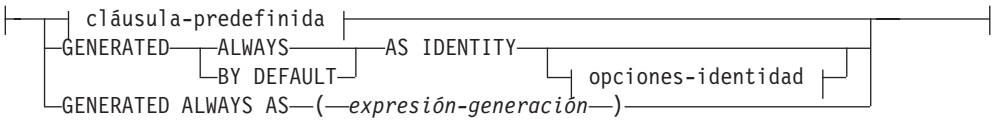
opciones-datalink:



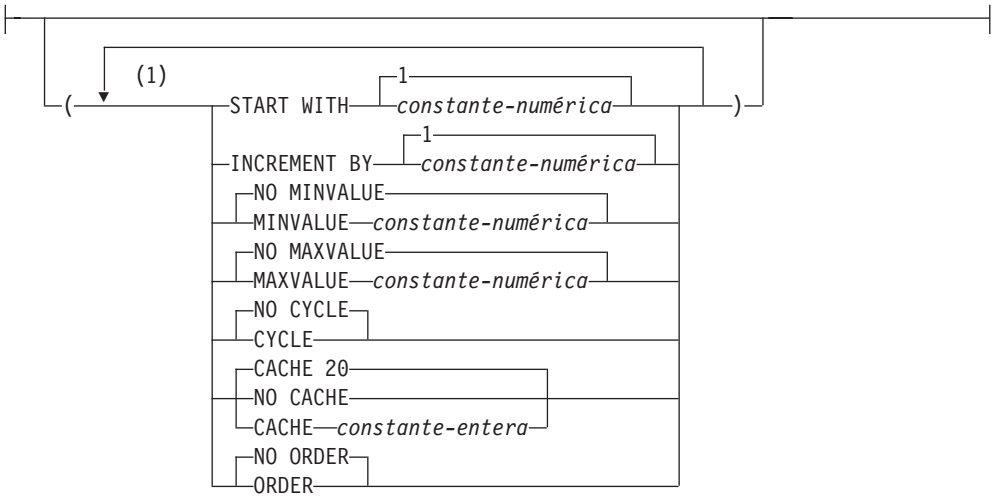
opciones-enlace-archivo:



espec-columna-generada:



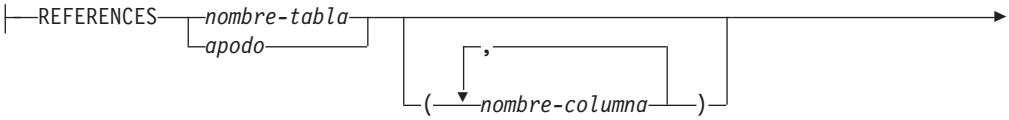
opciones-identidad:



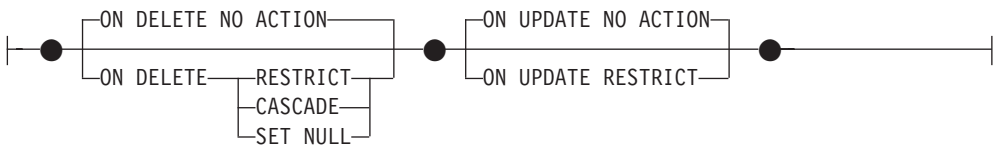
Notas:

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

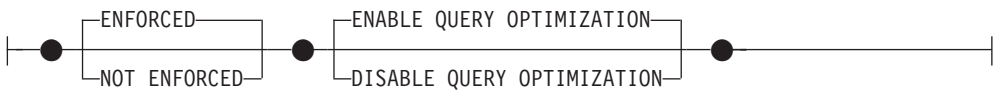
cláusula-referencias:



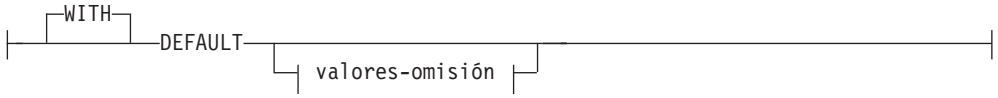
cláusula-norma:



atributos-restricción:

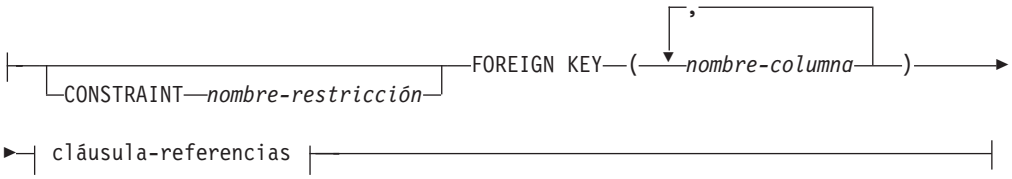


cláusula-predefinida:

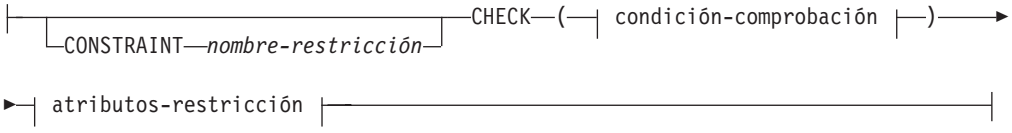


restricción-unicidad:

restricción-referencia:



restricción-comprobación:

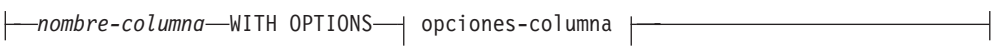


condición-comprobación:

dependencia-funcional:

definición-columna-OID:

opciones-with:



Descripción:
A las tablas de consultas materializadas mantenidas por el sistema y a las tablas de consultas materializadas mantenidas por el usuario se hace referencia por medio del término común *tabla de consultas materializadas*, a menos que sea necesario identificarlas por separado.

nombre-tabla
Indica el nombre de la tabla. El nombre, incluido el calificador implícito o explícito, no debe identificar una tabla, vista, apodo ni alias descrito en el catálogo. El nombre de esquema no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

OF *nombre-tipo1*
Especifica que las columnas de la tabla están basadas en los atributos del tipo estructurado identificado por *nombre-tipo1*. Si se especifica *nombre-tipo1* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico). El nombre de tipo debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se puede crear una instancia (SQLSTATE 428DP), con al menos un atributo (SQLSTATE 42997).

Si no se especifica UNDER, debe especificarse una columna de identificador de objeto (consulte *definición-columna-OID*). Esta columna de identificador de objeto es la primera columna de la tabla. La columna de ID de objeto va seguida de columnas basadas en los atributos de *nombre-tipo1*.

HIERARCHY *nombre-jerarquía*
Indica la tabla de jerarquía asociada con la jerarquía de tabla. Se crea a la vez que la tabla raíz de la jerarquía. Los datos para todas las subtablas en la jerarquía de tablas con tipo se almacenan en la tabla de jerarquía. No se puede hacer referencia a una tabla de jerarquía directamente en una sentencia de SQL. Un *nombre-jerarquía* es un *nombre-tabla*. El *nombre-jerarquía*, incluido el nombre de esquema implícito o explícito, no debe identificar una tabla, apodo, vista o alias descritos en el catálogo. Si se especifica el nombre de esquema,

debe ser el mismo que el nombre de esquema de la tabla que se está creando (SQLSTATE 428DQ). Si se omite esta cláusula al definir la tabla raíz, el sistema genera un nombre que consiste en el nombre de la tabla que se está creando seguido de un sufijo exclusivo de modo que el identificador es exclusivo dentro de los identificadores de las tablas, vistas, alias y apodos existentes.

UNDER *nombre-supertabla*
Indica que la tabla es una subtabla de *nombre-supertabla*. La supertabla debe ser una tabla existente (SQLSTATE 42704) y la tabla debe estar definida mediante un tipo estructurado que sea el supertipo inmediato de *nombre-tipo1* (SQLSTATE 428DB). El nombre de esquema de *nombre-tabla* y *nombre-supertabla* debe ser el mismo (SQLSTATE 428DQ). La tabla identificada por *nombre-supertabla* no debe tener ninguna subtabla existente ya definida mediante *nombre-tipo1* (SQLSTATE 42742).

Las columnas de la tabla incluyen la columna de identificador de objeto de la supertabla con su tipo modificado para que sea REF(*nombre-tipo1*), seguida de columnas basadas en los atributos de *nombre-tipo1* (recuerde que el tipo incluye los atributos de su supertipo). Los atributos no pueden tener el mismo nombre que la columna de OID (SQLSTATE 42711).

No pueden especificarse otras opciones de tabla como el espacio de tablas, DATA CAPTURE, NOT LOGGED INITIALLY y la clave de particionamiento. Estas opciones se heredan de la supertabla (SQLSTATE 42613).

INHERIT SELECT PRIVILEGES
Cualquier usuario o grupo que sostenga un privilegio SELECT sobre la supertabla recibirá un privilegio equivalente sobre la subtabla recién creada. Se considera que el definidor de la subtabla es el encargado de otorgar este privilegio.

lista-elementos
Define los elementos de una tabla. Esto incluye la definición de las columnas y las restricciones de la tabla.

lista-elementos-tipo
Define los elementos adicionales de una tabla con tipo. Esto incluye las opciones adicionales para las columnas, la adición de una columna de identificador de objeto (sólo la tabla raíz) y las restricciones de la tabla.

definición-consulta-materializada
Si la definición de tabla se basa en el resultado de una consulta, la tabla es una tabla de consultas materializadas basada en la consulta.

nombre-columna
Designa las columnas de la tabla. Si se especifica una lista de nombres de columna, ésta debe constar de tantos nombres como columnas haya en la tabla de resultados de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifica una lista de nombres de columnas, las columnas de la tabla heredan los nombres de las columnas de la tabla de resultados de la selección completa.

Debe especificarse una lista de nombres de columna si la tabla de resultados de la selección completa tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es una columna derivada de una constante, función, expresión u operación de conjuntos que no se designa utilizando la cláusula AS de la lista de selección.

AS

Introduce la consulta que se utiliza para la definición de la tabla y que determina los datos que se deben incluir en la tabla.

selección completa

Define la consulta en la que se basa la tabla. Las definiciones de columna resultantes son las mismas que las de una vista definida con la misma consulta.

Todos los elementos de la lista de selección deben tener un nombre (utilice la cláusula AS para las expresiones). La *definición-consulta-materializada* define los atributos de la tabla de consultas materializadas. La opción elegida también define el contenido de la selección completa de la manera siguiente.

Cuando se especifica WITH NO DATA, se puede especificar cualquier selección completa que no haga referencia a una tabla con tipo ni a una vista con tipo.

Cuando se especifica REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa no puede incluir (SQLSTATE 428EC):

- Referencias a una tabla de consultas materializadas, una tabla temporal declarada o una tabla con tipo en ninguna cláusula FROM
- Referencias a una vista donde la selección completa de la vista infrinja cualquiera de las restricciones que aparecen en la selección completa de la tabla de consultas materializadas
- Expresiones que sean un tipo de referencia o un tipo DATALINK (o un tipo diferenciado basado en estos tipos)
- Funciones que tengan cualquiera de los atributos siguientes:
  - EXTERNAL ACTION
  - LANGUAGE SQL
  - CONTAINS SQL
  - READS SQL DATA
  - MODIFIES SQL DATA
- Funciones que dependan de características físicas (por ejemplo, DBPARTITIONNUM, HASHEDVALUE)
- Referencias de tabla o vista a objetos del sistema (tampoco se deben especificar tablas Explain)
- Expresiones que sean un tipo estructurado o un tipo LOB (o un tipo diferenciado basado en un tipo LOB)

Cuando se especifica REPLICATED, se aplican las siguientes restricciones:

- No está permitida la cláusula GROUP BY.
- La tabla de consultas materializadas sólo debe hacer referencia a una única tabla; es decir, no puede incluir una unión.

Cuando se especifica REFRESH IMMEDIATE:

- La consulta debe ser una subselección, con la excepción de que se da soporte a UNION ALL en la expresión de tabla de entrada de GROUP BY.
- La consulta no puede ser recursiva.
- La consulta no puede incluir:
  - Referencias a un apodo

  - Funciones que no sean deterministas
  - Selecciones completas escalares
  - Predicados con selecciones completas
  - Registros especiales
  - SELECT DISTINCT
- Si la cláusula FROM hace referencia a más de una tabla o vista, sólo puede definir una unión interna sin utilizar la sintaxis INNER JOIN explícita.
- Cuando se especifica una cláusula GROUP BY, se aplican las siguientes consideraciones:
  - Las funciones de columna que reciben soporte son SUM, COUNT, COUNT_BIG y GROUPING (sin DISTINCT). La lista de selección debe contener una columna COUNT(*) o COUNT_BIG(*). Si la lista de selección de la tabla de consultas materializadas contiene SUM(X), donde la X es un argumento que puede ser nulo, la tabla de consultas materializadas también debe tener COUNT(X) en su lista de selección. Estas funciones de columna no pueden formar parte de ninguna expresión.
  - No está permitida una cláusula HAVING.
  - Si se utiliza en un grupo de particiones de base de datos con varias particiones, la clave de particionamiento debe ser un subconjunto de los elementos de GROUP BY.
- La tabla de consultas materializadas no debe contener filas duplicadas y se aplicarán las siguientes restricciones específicas de este requisito de exclusividad, según se especifique la cláusula GROUP BY o no.
  - Cuando se especifica una cláusula GROUP BY, se aplican la siguientes restricciones relacionadas con la exclusividad:
    - Todos los elementos de GROUP BY se deben incluir en la lista de selección.
    - Cuando GROUP BY contiene GROUPING SETS, CUBE o ROLLUP, los elementos de GROUP BY y las funciones de columna GROUPING de la lista de selección deben formar una clave exclusiva del conjunto de resultados. Por lo tanto, deberán satisfacerse las restricciones siguientes:
      - No se pueden repetir conjuntos de agrupación. Por ejemplo, `ROLLUP(X,Y),X` no está permitido, porque es equivalente a `GROUPING SETS((X,Y),(X),(X))`.
      - Si X es un elemento anulable de GROUP BY que aparece en GROUPING SETS, CUBE o ROLLUP, GROUPING(X) deberá aparecer en la lista de selección.
  - Cuando no se especifica la cláusula GROUP BY, se aplican las siguientes restricciones relacionadas con la exclusividad:
    - El requisito de exclusividad de la tabla de consultas materializadas se consigue por derivación de una clave exclusiva para la vista materializada a partir de una de las restricciones de clave exclusiva definidas en cada una de las tablas subyacentes. Por lo tanto, las tablas subyacentes deben tener definida como mínimo una restricción de clave exclusiva y las columnas de estas claves deben aparecer en la lista de selección de la definición de tabla de consultas materializadas.

- Cuando se especifica MAINTAINED BY FEDERATED_TOOL, sólo se permiten referencias a apodos en una cláusula FROM.

Cuando se especifica REFRESH DEFERRED y la tabla de consultas materializadas se crea con intención de proporcionarle una tabla de etapas asociada en una sentencia posterior, la selección completa de la tabla de consultas materializadas debe seguir las mismas restricciones y normas que una selección completa utilizada para crear una tabla de consultas materializadas con la opción REFRESH IMMEDIATE.

Una tabla de consultas materializadas cuya selección completa contiene una cláusula GROUP BY, obtiene el resumen de los datos de las tablas a las que se hace referencia en la selección completa. Este tipo de tabla de consultas materializadas también se denomina *tabla de resumen*. Una tabla de resumen en un tipo especializado de tabla de consultas materializadas.

WITH NO DATA
La consulta sólo se utiliza para definir la tabla. La tabla no se rellena con los resultados de la consulta y no puede utilizarse la sentencia REFRESH TABLE. Cuando se ha completado la sentencia CREATE TABLE, ya no se considera que la tabla sea una tabla de consultas materializadas.

Se definen las columnas de la tabla basándose en las definiciones de las columnas que resultan de la selección completa. Si la selección completa hace referencia a una tabla individual de la cláusula FROM, los elementos de la lista de selección que son columnas de esa tabla se definen utilizando el nombre de columna, tipo de datos y posibilidad de contener nulos de la tabla referenciada.

opciones-tabla-renovable
Defina las opciones que pueden renovarse de los atributos de tabla de consultas materializadas.

DATA INITIALLY DEFERRED
Los datos no se insertan en la tabla como parte de la sentencia CREATE TABLE. Se utiliza una sentencia REFRESH TABLE que especifique el *nombre-tabla* para insertar los datos en la tabla.

REFRESH
Indica cómo se mantienen los datos de la tabla.

DEFERRED
Los datos de la tabla pueden renovarse en cualquier momento mediante la sentencia REFRESH TABLE. Los datos de la tabla sólo reflejan el resultado de la consulta en forma de instantánea en el momento en que se procesa la sentencia REFRESH TABLE. Las tablas de consultas materializadas mantenidas por el sistema que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807). Las tablas de consultas materializadas mantenidas por el usuario que se han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE.

IMMEDIATE
Los cambios que se han realizado en las tablas subyacentes como parte de una sentencia DELETE, INSERT o UPDATE se aplican en cascada a la tabla de consultas materializadas. En este caso, el contenido de la tabla, en cualquier momento, es igual que cuando se procesa la *subselección* especificada. Las tablas de consultas materializadas que se

han definido con este atributo no admiten las sentencias INSERT, UPDATE o DELETE (SQLSTATE 42807).

ENABLE QUERY OPTIMIZATION
La tabla de consultas materializadas puede utilizarse para la optimización de la consulta cuando se dan las circunstancias adecuadas.

DISABLE QUERY OPTIMIZATION
La tabla de consultas materializadas no se utilizará para la optimización de la consulta. La tabla todavía puede consultarse directamente.

MAINTAINED BY
Especifica quién mantiene los datos de la tabla de consultas materializadas: el sistema, el usuario o una herramienta de duplicación. El valor por omisión es SYSTEM.

SYSTEM
Especifica que el sistema mantiene los datos de la tabla de consultas materializadas.

USER
Especifica que el usuario mantiene los datos de la tabla de consultas materializadas. El usuario puede realizar operaciones de actualización, supresión o inserción en las tablas de consultas materializadas mantenidas por el usuario. La sentencia REFRESH TABLE, que se utiliza para las tablas de consultas materializadas mantenidas por el sistema, no puede invocarse para las tablas de consultas materializadas mantenidas por el usuario. Sólo una tabla de consultas materializadas REFRESH DEFERRED puede definirse como MAINTAINED BY USER.

FEDERATED_TOOL
Especifica que la herramienta de duplicación mantiene los datos de la tabla de consultas materializadas. La sentencia REFRESH TABLE, que se utiliza para las tablas de consultas materializadas mantenidas por el sistema, no se puede invocar para las tablas de consultas materializadas mantenidas por herramientas federadas. Sólo se puede definir una única tabla de consultas materializadas REFRESH DEFERRED como MAINTAINED BY FEDERATED_TOOL.

definición-tabla-etapas
Define la consulta que recibe el soporte de la tabla de etapas indirectamente por medio de una tabla de consultas materializadas asociada. Las tablas subyacentes de la tabla de consultas materializadas también son las tablas subyacentes de su tabla de etapas asociada. La tabla de etapas recopila los cambios que deben aplicarse a la tabla de consultas materializadas para sincronizarla con el contenido de las tablas subyacentes.

nombre-columna-etapas
Indica los nombres de las columnas de la tabla de etapas. Si se especifica una lista de nombres de columna, ésta deberá estar compuesta de *dos* nombres más que columnas hay en la tabla de consultas materializadas para la que define la tabla. Si la tabla de consultas materializas es una tabla de consultas materializadas duplicada, o si la consulta que define la tabla de consultas materializadas no contiene una cláusula GROUP BY, la lista de nombres de columna deberá estar compuesta de *tres* nombres más que columnas hay en la tabla de consultas materializadas para la que se define la tabla de etapas. Cada nombre de columna debe ser exclusivo y no calificado. Si no se especifica una lista de nombres de columna, las columnas de la tabla heredan los nombres de las columnas de la tabla de consultas materializadas asociada. Las columnas adicionales se denominan

GLOBALTRANSID y GLOBALTRANSTIME y, si se necesita una tercera columna, ésta se denomina OPERATIONTYPE.

Tabla 4. Columnas adicionales que se añaden a las tablas de etapas

Nombre de columna	Tipo de datos	Descripción de la columna
GLOBALTRANSID	CHAR(8) FOR BIT DATA	El ID de transacción global de cada fila propagada
GLOBALTRANSTIME	CHAR(13) FOR BIT DATA	La indicación de la hora de la transacción
OPERATIONTYPE	INTEGER	Operación para la fila propagada, que puede ser una inserción, una actualización o una supresión.

Deberá especificarse una lista de nombres de columnas si cualquiera de las columnas de la tabla de consultas materializadas asociada duplica cualquiera de los nombres de columna generados (SQLSTATE 42711).

FOR *nombre-tabla2*
Especifica la tabla de consultas materializadas que se utiliza para la definición de la tabla de etapas. El nombre, incluido el esquema implícito o explícito, debe identificar una tabla de consultas materializadas que exista en el servidor actual definido con REFRESH DEFERRED. La selección completa de la tabla de consultas materializadas asociada debe seguir las mismas restricciones y normas que una selección completa que se ha utilizado para crear una tabla de consultas materializadas con la opción REFRESH IMMEDIATE.

El contenido de la tabla de etapas puede utilizarse para renovar la tabla de consultas materializadas, invocando la sentencia REFRESH TABLE, si el contenido de la tabla de etapas es coherente con la tabla de consultas materializadas asociada y las tablas fuente subyacentes.

PROPAGATE IMMEDIATE
Los cambios que se han realizado en las tablas subyacentes como parte de una operación de supresión, inserción o actualización, se aplican en cascada a la tabla de etapas en la misma operación de supresión, inserción o actualización. Si la tabla de etapas no se ha marcado como no coherente, su contenido, en cualquier momento, son los cambios delta para la tabla subyacente desde la última renovación de la tabla de consultas materializadas.

LIKE *nombre-tabla1* **o** *nombre-vista* **o** *apodo*
Especifica que las columnas de la tabla tienen exactamente el mismo nombre y descripción que las columnas de la tabla (*nombre-tabla1*), vista (*nombre-vista*) o apodo (*apodo*) identificado. El nombre especificado después de LIKE debe identificar una tabla, vista o apodo existentes en el catálogo, o una tabla temporal declarada. No se puede especificar una tabla con tipo ni una vista con tipo (SQLSTATE 428EC).

El uso de LIKE es una definición implícita de *n* columnas, donde *n* es el número de columnas de la tabla, vista o apodo designados.

- Si se designa una tabla, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *nombre-tabla1*. Si no se especifica EXCLUDING COLUMN DEFAULTS, también se incluye el valor por omisión de la columna.

- Si se designa una vista, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna resultante de la selección completa definida en *nombre-vista*.
- Si se designa una apodo, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *apodo*.

Según cuáles sean las cláusulas de atributos de copia, se pueden incluir o excluir el valor por omisión de la columna y los atributos IDENTITY de la columna. La definición implícita no incluye ningún otro atributo de la tabla, vista o apodo designados. Por lo tanto, la nueva tabla no tiene ninguna restricción de unicidad, restricción de clave foránea, activadores ni índices. La tabla se crea en el espacio de tablas implícita o explícitamente especificado mediante la cláusula IN y la tabla tiene cualquier otra cláusula opcional sólo si la cláusula opcional se especifica.

opciones-copia
Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados fuente (tabla, vista o selección completa).

INCLUDING COLUMN DEFAULTS
Especifica que deben copiarse los valores por omisión de cada columna actualizable contenida en la definición de la tabla de resultados fuente. Las columnas que no son actualizables no tienen un valor por omisión definido en la correspondiente columna de la tabla creada.

Si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada, la opción por omisión es INCLUDING COLUMN DEFAULTS.

EXCLUDING COLUMN DEFAULTS
Especifica que no deben copiarse los valores por omisión de las columnas de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada.

INCLUDING IDENTITY COLUMN ATTRIBUTES
Los atributos de columna de identidad se copian desde la definición de la tabla de resultados fuente, si es posible. Es posible copiar los atributos de columna de identidad, si el elemento de la correspondiente columna de la tabla, vista o selección completa es el nombre de una columna de tabla o de vista que, directa o indirectamente, se correlaciona con el nombre de una columna de tabla base que tiene el atributo de identidad. En todos los demás casos, las columnas de la nueva tabla no obtendrán el atributo de identidad. Por ejemplo:

- La lista de selección de la selección completa contiene varias instancias de un nombre de columna de identidad (es decir, se selecciona la misma columna más de una vez)
- la lista de selección de la selección completa contiene varias columnas de identidad (es decir, supone la ejecución de una operación de unión)
- la columna de identidad está contenida en una expresión de la lista de selección
- la selección completa contiene una operación de conjuntos (union, except o intersect).

EXCLUDING IDENTITY COLUMN ATTRIBUTES
Especifica que no deben copiarse los atributos de columna de identidad contenidos en la definición de la tabla de resultados fuente.

ORGANIZE BY DIMENSIONS *(nombre-columna,...)*
Especifica una dimensión para cada columna o grupo de columnas que se utiliza para el establecimiento de los datos de tabla en un clúster. La utilización de paréntesis dentro de la lista de dimensiones especifica que un grupo de columnas va a tratarse como una dimensión. La palabra clave DIMENSIONS es opcional.

Se mantiene automáticamente un índice de bloques de clúster para cada dimensión especificada y se mantiene un índice de bloques, compuesto de todas las columnas que se utilizan en la cláusula, si ninguno de los índices de bloques de clúster las incluyen. El conjunto de columnas que se utiliza en la cláusula ORGANIZE BY debe seguir las normas de la sentencia CREATE INDEX.

Cada nombre de columna que se especifica en la cláusula ORGANIZE BY deberá haberse definido para la tabla (SQLSTATE 42703), y una dimensión no puede aparecer más de una vez en la lista de dimensiones (SQLSTATE 42709).

Las páginas de la tabla se organizan en bloques de igual tamaño, que es el tamaño de extensión del espacio de tabla, y todas las filas de cada bloque contienen la misma combinación de valores de dimensión.

ORGANIZE BY KEY SEQUENCE *espec-clave-secuencia*
Especifica que la tabla está organizada en secuencia de clave ascendente, con un tamaño fijo basado en el rango especificado de los valores de secuencia de clave. Una tabla organizada de este modo recibe el nombre de *tabla de clúster de rango*. Cada posible valor de clave del rango definido tiene una ubicación predeterminada en la tabla física. El almacenamiento necesario para una tabla de clúster de rango debe estar disponible cuando se crea la tabla y debe ser suficiente como para que contenga el número de filas del rango especificado multiplicado por el tamaño de fila (para ver detalles sobre cómo determinar los requisitos de espacio, consulte “Tamaño de fila” en la página 393 y “Cuentas de bytes” en la página 393).

nombre-columna
Especifica una columna de la tabla que se incluye en la clave exclusiva que determina la secuencia de la tabla de clúster de rango. El tipo de datos de la columna debe ser SMALLINT, INTEGER o BIGINT (SQLSTATE 42611) y las columnas deben estar definidas como NOT NULL (SQLSTATE 42831). La misma columna no puede estar identificada más de una vez e la clave de secuencia. El número de columnas identificadas no debe superar 16 (SQLSTATE 54008).

Se creará una entrada de índice exclusiva automáticamente en el catálogo para las columnas de la secuencia de clave especificadas con orden ascendente para cada columna. El nombre del índice será SQL, seguido de los caracteres de indicación horaria (*aammddhhmmssxxx*), con SYSIBM como el nombre de esquema. Un objeto de índice real no se crea en almacenamiento, porque la organización de la tabla se ordena por esta clave. Si se define una clave principal o una restricción exclusiva en las mismas columnas que la clave de secuencia de clave de clúster de rango, se utiliza esta misma entrada de índice para la restricción.

Para la especificación de secuencia de clave, existe una restricción de comprobación para reflejar las restricciones de columna. Si se especifica la cláusula DISALLOW OVERFLOW, el nombre de la restricción de

comprobación será RCT y se obliga el cumplimiento con la restricción de comprobación. Si se especifica la cláusula SALLOW OVERFLOW, el nombre de la restricción de comprobación será RCT_OFLOW y no se obliga el cumplimiento con la restricción de comprobación.

STARTING FROM *constante*
Especifica el valor constante en el extremo inferior del rango para *nombre-columna*. Los valores menores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821). Si no se especifica ninguna constante inicial, el valor por omisión es 1.

ENDING AT *constante*
Especifica el valor constante en el extremo superior del rango para *nombre-columna*. Los valores mayores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. El valor de la constante final debe ser mayor que la constante inicial. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821).

ALLOW OVERFLOW
Especifica que la tabla de clúster de rango permite filas con valores de clave que quedan fuera del rango de valores definido. Cuando se crea una tabla de clúster de rango que permite desbordamientos, las filas con valores de clave que quedan fuera del rango se colocan al final del rango definido sin ningún orden predeterminado. Las operaciones en las que intervienen estas filas de desbordamiento son menos eficientes que las operaciones sobre filas que tienen valores que quedan dentro del rango definido.

DISALLOW OVERFLOW
Especifica que la tabla de clúster de rango no permite filas con valores de clave que no quedan dentro del rango de valores definido (SQLSTATE 23513). Las tablas de clúster de rango que no permiten desbordamientos siempre mantendrán todas las filas en secuencia de clave ascendente.

definición-columna
Define los atributos de una columna.

nombre-columna
Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no deben superar el valor 4 005.
- Un tamaño de página de 8K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 8 101.
- Un tamaño de página de 16K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 16 293.
- Un tamaño de página de 32K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 32 677.

Para obtener más detalles, consulte el apartado "Tamaño de fila" en la página 393.

tipo-datos
Se trata de uno de los tipos de la lista siguiente. Utilice:

SMALLINT
Para especificar un entero pequeño.

INTEGER o **INT**
Para especificar un entero grande.

BIGINT
Para especificar un entero superior.

FLOAT(*entero***)**
Para especificar un número de coma flotante de precisión simple o doble, en función del valor del *entero*. El valor del entero debe estar en el rango 1-53. Los valores 1 a 24 indican precisión simple y los valores 25 a 53 indican precisión doble.

También puede especificar:

REAL	Para especificar un valor de coma flotante de precisión simple.
DOUBLE	Para especificar coma flotante de precisión doble.
DOUBLE PRECISION	Para especificar coma flotante de precisión doble.
FLOAT	Para especificar coma flotante de precisión doble.

DECIMAL*(entero-precisión, entero-escala)* o **DEC***(entero-precisión, entero-escala)*
Para especificar un número decimal. El primer entero es la precisión del número, es decir, el número total de dígitos; los valores permitidos son del 1 al 31. El segundo entero es la escala del número (es decir, el número de dígitos situados a la derecha de la coma decimal); su valor varía entre 0 y la precisión del número.

Si no se especifican la precisión ni la escala, se emplean los valores por omisión 5,0. Las palabras **NUMERIC** y **NUM** pueden utilizarse como sinónimos de **DECIMAL** y **DEC**.

CHARACTER*(entero)* o **CHAR***(entero)* o **CHARACTER** o **CHAR**
Para una serie de caracteres de longitud fija de longitud *entero*, que puede ser del 1 al 254. Si se omite la especificación de la longitud, se supone que la longitud es de 1 carácter.

VARCHAR*(entero)*, o **CHARACTER VARYING***(entero)*, o **CHAR VARYING***(entero)*
Para una serie de caracteres de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 32 672.

LONG VARCHAR
Para una serie de caracteres de longitud variable con una longitud máxima de 32 700.

FOR BIT DATA
Especifica que el contenido de la columna se tratará como datos de bit (binarios). Durante el intercambio de datos con otros sistemas, no se

efectúan conversiones de página de códigos. Las comparaciones se efectúan en binario, sin tener en cuenta el orden de clasificación de la base de datos.

BLOB o BINARY LARGE OBJECT*(entero [K | M | G])*
Para una serie de gran objeto binario de la longitud máxima especificada en bytes.

La longitud puede encontrarse dentro del rango de 1 byte hasta 2 147 483 647 bytes.

Si el *entero* ya está especificado por sí solo, se entiende que esa es la longitud máxima.

Si se especifica *entero K* (ya sea en mayúsculas o en minúsculas), la longitud máxima es el *entero* multiplicado por 1 024. El valor máximo de *entero* es 2 097 152. Si se especifica un múltiplo de K, M o G cuyo cálculo da 2 147 483 648, el valor real que se utiliza es 2 147 483 647 (o 2 gigabytes menos 1 byte), que es la longitud máxima para una columna LOB.

Si se especifica *entero M*, la longitud máxima es 1 048 576 multiplicada por *entero*. El valor máximo de *entero* es 2 048.

Si se especifica *entero G*, la longitud máxima es 1 073 741 824 multiplicada por *entero*. El valor máximo de *entero* es 2.

Si se omite la especificación de la longitud, se da por supuesta una longitud de 1 048 576 (1 megabyte).

Para crear series BLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

Está permitido especificar cualquier número de espacios entre el entero y K, M o G; por otra parte, no es obligatorio especificar un espacio. Por ejemplo, todos los valores siguientes son válidos:

```
BLOB(50K)    BLOB(50 K)    BLOB (50   K)
```

CLOB o CHARACTER (CHAR) LARGE OBJECT*(entero [K | M | G])*
Para una serie de gran objeto de caracteres de la longitud máxima especificada en bytes.

El significado de *entero K | M | G* es el mismo que para BLOB.

Si se omite la especificación de la longitud, se da por supuesta una longitud de 1 048 576 (1 megabyte).

Para crear series CLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

No es posible especificar la cláusula FOR BIT DATA para las columnas CLOB. Sin embargo, una serie CHAR FOR BIT DATA puede asignarse a una columna CLOB y una serie CHAR FOR BIT DATA puede concatenarse con una serie CLOB.

DBCLOB*(entero [K | M | G])*
Para una serie de gran objeto de caracteres de doble byte de la longitud máxima especificada en caracteres de doble byte.

El significado de *entero K | M | G* es parecido al de BLOB. La diferencia es que el número especificado es el número de caracteres de doble byte y que el tamaño máximo es de 1 073 741 823 caracteres de doble byte.

Si se omite la especificación de la longitud, se da por supuesta una longitud de 1 048 576 caracteres de doble byte.

Para crear series DBCLOB con más de 1 gigabyte, debe especificar la opción NOT LOGGED.

GRAPHIC*(entero)*
Para una serie gráfica de longitud fija de longitud *entero*, que puede ir de 1 a 127. Si se omite la especificación de longitud, se supone una longitud de 1.

VARGRAPHIC*(entero)*
Para una serie gráfica de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 16 336.

LONG VARGRAPHIC
Para una serie gráfica de longitud variable con una longitud máxima de 16 350.

DATE
Para la fecha.

TIME
Para la hora.

TIMESTAMP
Para la indicación horaria.

DATALINK o **DATALINK***(entero)*
Para un enlace con un archivo de datos que se ha almacenado fuera de la base de datos.

La columna de la tabla consta de ″valores ancla″ que contienen la información de referencia que se necesita para establecer y mantener el enlace con los datos externos así como un comentario opcional.

La longitud de una columna DATALINK es de 200 bytes. Si se especifica *entero*, debe ser 200. Si se omite la especificación de longitud, se supone una longitud de 200 bytes.

Un valor DATALINK es un valor encapsulado con un conjunto de funciones escalares incorporadas. Existe una función denominada DLVALUE que se utiliza para crear un valor DATALINK, y también pueden utilizarse las funciones denominadas DLNEWCOPY, DLPREVIOUSCOPY y DLREPLACECONTENT para construir un valor DATALINK en circunstancias especiales. (DLVALUE debe utilizarse para construir un valor DATALINK normal.) Pueden utilizarse las funciones siguientes para extraer atributos de un valor DATALINK.

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLCOMPLETEONLY
- DLURLCOMPLETEWRITE
- DLURLPATH
- DLURLPATHONLY
- DLURLPATHWRITE
- DLURLSCHEME
- DLURLSERVER

Una columna DATALINK tiene las restricciones siguientes:

- La columna no puede formar parte de ningún índice. Por lo tanto, no puede incluirse como columna de una clave primaria ni de una restricción de unicidad (SQLSTATE 42962).
- La columna no puede ser una clave foránea de una restricción de referencia (SQLSTATE 42830).
- No puede especificarse un valor por omisión (WITH DEFAULT) para la columna. Si la columna puede ser nula, el valor por omisión para la columna es NULL (SQLSTATE 42894).

nombre-tipo-diferenciado
Para un tipo definido por el usuario que es un tipo diferenciado. Si se especifica un nombre de tipo diferenciado sin un nombre de esquema, el nombre del tipo diferenciado se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico).

Si se define una columna con un tipo diferenciado, el tipo de datos de la columna es el tipo diferenciado. La longitud y la escala de la columna son, respectivamente, la longitud y la escala del tipo de fuente del tipo diferenciado.

Si una columna definida con un tipo diferenciado es la clave externa de una restricción de referencia, el tipo de datos de la columna correspondiente de la clave primaria debe tener el mismo tipo diferenciado.

nombre-tipo-estructurado
Para especificar un tipo definido por el usuario que es un tipo estructurado. Si se especifica un nombre de tipo estructurado sin un nombre de esquema, el nombre de tipo estructurado se resuelve buscando en los esquemas especificados en la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el SQL estático y por el registro CURRENT PATH en el SQL dinámico).

Si se define una columna utilizando un tipo estructurado, el tipo de datos estáticos de la columna es el tipo estructurado. La columna puede contener valores con un tipo dinámico que es un subtipo de *nombre-tipo-estructurado*.

Si se define una columna utilizando un tipo estructurado, la columna no se puede utilizar en una clave primaria, restricción de unicidad, clave foránea, clave de índice ni clave de particionamiento (SQLSTATE 42962).

Si se define una columna utilizando un tipo estructurado, y la columna contiene un atributo de tipo de referencia, a cualquier nivel de anidamiento, ese atributo no tiene ámbito. Para utilizar un atributo de esa clase en una operación de eliminación de referencia, es necesario especificar explícitamente un ámbito (SCOPE), utilizando una especificación CAST.

Si se define una columna utilizando un tipo estructurado con un atributo de tipo DATALINK, o un tipo diferenciado con origen en DATALINK, esta columna sólo puede ser nula. Si se intenta utilizar la función constructora para este tipo, se obtendrá un error (SQLSTATE 428ED), y por tanto no se puede insertar ninguna instancia de este tipo en la columna.

REF (*nombre-tipo2***)**
Para una referencia a una tabla con tipo. Si se especifica *nombre-tipo2* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso de SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). El tipo de datos subyacente de la columna se basa en el tipo de datos de representación que se ha especificado en la cláusula REF USING de la sentencia CREATE TYPE para *nombre-tipo2* o el tipo raíz de la jerarquía de tipos de datos que incluye a *nombre-tipo2*.

opciones-columna
Define opciones adicionales relacionadas con las columnas de la tabla.

NOT NULL
Evita que la columna contenga valores nulos.

Si no se especifica NOT NULL, la columna puede contener valores nulos, y su valor por omisión es un valor nulo o bien el valor proporcionado por la cláusula WITH DEFAULT.

opciones-lob
Especifica opciones para los tipos de datos LOB.

LOGGED
Especifica que los cambios efectuados en la columna deben registrarse en el archivo de anotaciones cronológicas. Acto seguido, los programas de utilidad de la base de datos (como RESTORE DATABASE) pueden recuperar los datos de estas columnas. LOGGED es el valor por omisión.

Los LOB superiores a 1 gigabyte no se pueden registrar (SQLSTATE 42993).

NOT LOGGED
Especifica que los cambios efectuados en la columna no se van a anotar cronológicamente.

NOT LOGGED no tiene ningún efecto en una operación de confirmación o retrotracción; es decir, la coherencia de la base de datos se mantiene incluso si una transacción se retrotrae, no importa si se anota cronológicamente el valor LOB o no. La implicación de no efectuar el registro es que durante una operación de avance, tras una operación de carga o de copia de seguridad, los datos de LOB se sustituirán por ceros en todos aquellos valores de LOB que hubieran generado entradas de la anotación cronológica que se habrían reproducido durante el avance. Durante la recuperación de una colisión, todos los cambios confirmados y los cambios retrotraídos reflejarán los resultados esperados.

COMPACT
Especifica que los valores de la columna LOB deben ocupar el mínimo espacio de disco (liberar las páginas de disco sobrantes del último grupo utilizado por el valor LOB), en lugar de dejar el espacio restante al final del espacio de almacenamiento de LOB que podría facilitar operaciones posteriores de adición. Observe que almacenar datos de esta manera puede influir negativamente en el rendimiento de cualquier operación de adición (aumento de longitud) que se lleve a cabo en la columna.

NOT COMPACT
Especifica cierto espacio para las inserciones para facilitar los cambios futuros que se efectúen en los valores LOB de la columna. Este es el valor por omisión.

opciones-datalink
Especifica las opciones asociadas a un tipo de datos DATALINK.

LINKTYPE URL
Define el tipo de enlace como un Localizador de recursos uniformes (URL).

NO LINK CONTROL
Especifica que no se realizará ninguna comprobación para determinar si existe el archivo. Sólo se comprobará la sintaxis del URL. El gestor de bases de datos no tiene control sobre el archivo.

FILE LINK CONTROL
Especifica que debe comprobarse la existencia del archivo. Se pueden utilizar opciones adicionales para proporcionar al gestor de bases de datos un control mayor sobre el archivo.

opciones-enlace-archivo
Opciones adicionales para definir el nivel de control de gestor de bases de datos del enlace del archivo.

INTEGRITY
Especifica el nivel de integridad del enlace entre un valor DATALINK y el archivo real.

ALL
Cualquier archivo especificado como un valor DATALINK está bajo el control del gestor de bases de datos y NO puede suprimirse ni renombrarse mediante interfaces de programación de sistema de archivos de tipo estándar.

READ PERMISSION
Especifica cómo se determina el permiso para leer el archivo especificado en un valor DATALINK.

FS El permiso de acceso de lectura está determinado por los permisos del sistema de archivos. Se puede acceder a estos archivos sin recuperar el nombre de archivo de la columna.

DB
El permiso de acceso de lectura está determinado por la base de datos. Sólo se permitirá el acceso al archivo si se pasa un símbolo de accesos de archivo válido, devuelto en la recuperación del valor DATALINK de la tabla, en la operación de la apertura.

WRITE PERMISSION
Especifica cómo se determina el permiso para grabar en el archivo especificado en un valor DATALINK.

FS El permiso de acceso de grabación está determinado por los permisos del sistema de archivos. Se puede acceder a estos archivos sin recuperar el nombre de archivo de la columna.

BLOCKED
El acceso de grabación está bloqueado. El archivo no puede

actualizarse directamente mediante ninguna interfaz. Debe utilizarse un mecanismo alternativo para que puedan realizarse actualizaciones en la información. Por ejemplo, se copia el archivo, se actualiza la copia y se actualiza el valor DATALINK para señalar la nueva copia del archivo.

ADMIN
El permiso de grabación lo determina Data Links Manager. El permiso de grabación para el archivo sólo se otorgará pasando un símbolo de grabación válido, devuelto al recuperarse del valor DATALINK de la tabla, mediante la utilización de la función escalar DLURLCOMPLETEWRITE o DLURLPATHWRITE, en la operación de apertura. Este valor sólo puede especificarse cuando también se especifica READ PERMISSION DB.

El privilegio de acceso para un archivo enlazado determinado se define y se mantiene en Data Links Manager.

Cuando un usuario (el usuario de la actualización) ha abierto un archivo para grabar en él con un símbolo de grabación válido, los demás usuarios pueden seguir abriendo el archivo para leerlo utilizando un símbolo de lectura o grabación válido. Sin embargo, sólo el mismo usuario de la actualización podrá abrir repetidamente el archivo para grabar en él con el mismo símbolo de grabación. El mismo usuario de la actualización también necesitará el mismo símbolo de grabación para realizar cualquier operación de lectura posterior deseada.

REQUIRING TOKEN FOR UPDATE
Para completar la actualización del archivo, el símbolo de grabación que se ha utilizado para abrir y para modificar el archivo deberá estar contenido en la referencia de archivo que se ha especificado durante la invocación de las funciones escalares DLNEWCOPY o DLPREVIOUSCOPY en la sentencia UPDATE de SQL.

NOT REQUIRING TOKEN FOR UPDATE
Para completar la actualización del archivo, no se necesita un símbolo de grabación en la referencia de archivo durante la invocación de las funciones escalares DLNEWCOPY o DLPREVIOUSCOPY en la sentencia UPDATE de SQL.

RECOVERY
Especifica si DB2 va a dar soporte a la recuperación puntual en el tiempo de los archivos referidos por los valores de esta columna.

YES
DB2 va a dar soporte a la recuperación puntual en el tiempo de los archivos referidos por los valores de esta columna. Este valor sólo puede especificarse cuando también se especifican INTEGRITY ALL y WRITE PERMISSION BLOCKED o WRITE PERMISSION ADMIN.

NO
Especifica que no se va a dar soporte a la recuperación puntual en el tiempo.

ON UNLINK
Especifica la acción realizada en un archivo cuando se cambia o se suprime un valor DATALINK (desenlace). Tenga en cuenta que esto no se puede aplicar cuando se utiliza WRITE PERMISSION FS.

RESTORE
Especifica que, cuando se elimine el enlace de un archivo, Data Links File Manager intentará devolver el archivo al propietario que tiene los permisos que existían en el momento de enlazarse el archivo. En caso de que el usuario ya no esté registrado en el servidor de archivos, el archivo se asigna a un ID de usuario especial previamente definido, "dfmunknown". Sólo puede especificarse cuando también se especifican INTEGRITY ALL y WRITE PERMISSION BLOCKED o WRITE PERMISSION ADMIN.

DELETE
Especifica que el archivo se suprimirá cuando se desenlace. Sólo puede especificarse cuando también se especifican READ PERMISSION DB y WRITE PERMISSION BLOCKED o WRITE PERMISSION ADMIN.

MODE DB2OPTIONS
Esta modalidad define un conjunto de opciones de enlace de archivo por omisión. Los valores por omisión definidos por DB2OPTIONS son:

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

No puede aplicarse ON UNLINK desde el momento en que utiliza WRITE PERMISSION FS.

SCOPE
Identifica el ámbito de la columna de tipo de referencia.

Debe especificarse un ámbito para cualquier columna que vaya a utilizarse como operando izquierdo de un operador de eliminación de referencia o como argumento de la función DEREF. La especificación del ámbito de una columna de tipo de referencia puede diferirse a una sentencia ALTER TABLE subsiguiente para permitir que se defina la tabla de destino, normalmente en el caso de tablas que se hacen referencia mutuamente.

nombre-tabla-tipo
El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar a ninguna restricción que ya esté especificada en la misma sentencia CREATE TABLA (SQLSTATE 42710).

Si se omite esta cláusula, el sistema genera un identificador de 18 caracteres que es exclusivo entre los identificadores de las restricciones existentes que se han definido en la tabla. (El identificador se compone de la palabra "SQL" seguida de una secuencia de 15 caracteres numéricos que genera una función basada en la indicación de la hora.)

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* puede utilizarse como el nombre de un índice que se ha creado para dar soporte a la restricción.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. De este modo, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

Consulte el apartado PRIMARY KEY en la descripción de la *restricción-unicidad* más abajo.

UNIQUE
Proporciona un método abreviado de definir una clave de unicidad compuesta de una sola columna. Por lo tanto, si se especifica UNIQUE en la definición de la columna C, el efecto es el mismo que si se especificase la cláusula UNIQUE(C) como una cláusula separada.

No puede especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

Consulte el apartado UNIQUE en la descripción de la *restricción-unicidad* más abajo.

cláusula-referencias
Proporciona un método abreviado de definir una clave externa compuesta de una sola columna. Así, si se especifica una cláusula-referencias en la definición de la columna C, el efecto es el mismo que si se especificase esa cláusula-referencias como parte de una cláusula FOREIGN KEY en la que C fuera la única columna identificada.

Consulte *cláusula-referencias* en *restricción-referencia*, que encontrará más adelante.

CHECK (*condición-error***)**
Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Vea CHECK (*condición-error*) más adelante.

INLINE LENGTH *entero*
Esta opción sólo es válida para una columna que se ha definido utilizando un tipo estructurado (SQLSTATE 42842) e indica el tamaño máximo, en bytes, de una instancia de un tipo estructurado para su almacenamiento en serie con el resto de valores de la fila. Las instancias de tipos estructurados que no se pueden almacenar en serie se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB. Esta operación se realiza automáticamente.

El valor por omisión de INLINE LENGTH para una columna de tipo estructurado es la longitud ″inline″ de su tipo (especificada explícitamente o por omisión en la sentencia CREATE TYPE). Si el valor INLINE LENGTH del tipo estructurado es menor que 292, se utiliza el valor 292 para la columna.

Nota: Las longitudes ″inline″ de los subtipos no se incluyen en la longitud ″inline″ por omisión, debido a que las instancias de los subtipos pueden no caber ″inline″ a menos que, al crear la tabla, se especifique explícitamente un valor INLINE LENGTH para tener en cuenta los subtipos actuales y futuros.

El valor INLINE LENGTH explícito debe ser igual a 292 como mínimo y no ser mayor que 32672 (SQLSTATE 54010).

COMPRESS SYSTEM DEFAULT
Especifica que los valores por omisión del sistema (es decir, los valores por omisión que se utilizan para los tipos de datos cuando no se ha especificado ningún valor específico) van a almacenarse utilizando el espacio mínimo. Si no se especifica la cláusula VALUE COMPRESSION, se devuelve un mensaje de aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el espacio mínimo.

El hecho de permitir que los valores por omisión del sistema se almacenen de esta forma da lugar a una ligera pérdida de rendimiento durante las operaciones de inserción y actualización que se realizan en la columna debido a la comprobación adicional que tiene lugar.

El tipo de datos base no debe ser DATE, TIME ni TIMESTAMP (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto. Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

espec-columna-generada

cláusula-predefinida
Especifica un valor por omisión para la columna.

WITH
Palabra clave opcional.

DEFAULT
Proporciona un valor por omisión en el caso de que no se suministre ningún valor en INSERT o se especifique uno como DEFAULT en INSERT o UPDATE. Si no se especifica un valor por omisión a continuación de la palabra clave DEFAULT, el valor por omisión depende del tipo de datos de la columna, tal como se muestra en "ALTER TABLE".

Si una columna se define como DATALINK, no puede especificarse un valor por omisión (SQLSTATE 42613). El único valor por omisión posible es NULL.

Si la columna está basada en una columna de una tabla con tipo, debe especificarse un valor por omisión específico cuando se defina un valor por omisión. No se puede especificar un valor por omisión para la columna de identificador de objeto de una tabla con tipo (SQLSTATE 42997).

Si se define una columna utilizando un tipo diferenciado, el valor por omisión de la columna es el valor por omisión del tipo de datos fuente convertido al tipo diferenciado.

Si una columna se define utilizando un tipo estructurado, no puede especificarse la *cláusula-predefinida* (SQLSTATE 42842).

La omisión de DEFAULT en una *definición-columna* da como resultado la utilización del valor nulo como valor por omisión para la columna. Si dicha columna está definida como NOT NULL, la columna no tiene un valor por omisión válido.

valores-omisión
Los tipos específicos de los valores por omisión que pueden especificarse son los siguientes.

constante
Especifica la constante como el valor por omisión para la columna. La constante especificada debe:

- representar un valor que pueda asignarse a la columna de acuerdo con las normas de asignación descritas en el Capítulo 3
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- no tener dígitos diferentes de cero fuera de la escala del tipo de datos de la columna si la constante es decimal (por ejemplo, 1,234 no puede ser el valor por omisión para una columna DECIMAL(5,2))
- estar expresada con un máximo de 254 caracteres, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificado y paréntesis, cuando la constante es el argumento de una *función-conversión*.

registro-especial-fechahora
Especifica el valor que tenía el registro especial de indicación de fecha y hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por

omisión de la columna. El tipo de datos de la columna debe ser el tipo de datos que corresponde al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE).

registro-especial-usuario
Especifica el valor del registro especial de usuario (CURRENT USER, SESSION_USER, SYSTEM_USER) en el momento de ejecutar INSERT, UPDATE o LOAD como valor por omisión de la columna. El tipo de datos de la columna debe ser de serie de caracteres con una longitud no inferior al atributo de longitud de un registro especial de usuario. Tenga en cuenta que se puede especificar USER en lugar de SESSION_USER y CURRENT_USER en lugar de CURRENT USER.

CURRENT SCHEMA
Especifica el valor que tenía el registro especial CURRENT SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA.

NULL
Especifica NULL como valor por omisión para la columna. Si se ha especificado NOT NULL, puede especificarse DEFAULT NULL en la misma definición de columna, pero se producirá un error si se intenta establecer la columna en el valor por omisión.

función-conversión
Esta forma de valor por omisión sólo puede utilizarse con las columnas definidas como tipo de datos diferenciado, BLOB o de indicación de fecha y hora (DATE, TIME o TIMESTAMP). Para el tipo diferenciado, a excepción de los tipos diferenciados basados en tipos BLOB o de indicación de fecha y hora, el nombre de la función debe coincidir con el nombre del tipo diferenciado de la columna. Si está calificado con un nombre de esquema, debe ser el mismo que el nombre de esquema del tipo diferenciado. Si no está calificado, el nombre de esquema procedente de la resolución de la función debe ser el mismo que el nombre de esquema del tipo diferenciado. Para un tipo diferenciado basado en un tipo de indicación de fecha y hora, en el que el valor por omisión es una constante, debe utilizarse una función y el nombre de ésta debe coincidir con el nombre del tipo de fuente del tipo diferenciado, con un nombre de esquema implícito o explícito de SYSIBM. Para las demás columnas de indicación de fecha y hora, también puede utilizarse la correspondiente función de indicación de fecha y hora. Para un BLOB o tipo diferenciado basado en BLOB, debe utilizarse una función, y el nombre de la función debe ser BLOB, junto con SYSIBM como nombre de esquema implícito o explícito.

constante
Especifica una constante como argumento. La constante debe cumplir las normas de una constante para el tipo de fuente del tipo diferenciado, o para el tipo de datos si no se trata de un tipo diferenciado. Si la *función-conversión* es BLOB, la constante debe ser una constante de serie de caracteres.

registro-especial-fechahora
Especifica CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP. El tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

registro-especial-usuario
Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos serie con una longitud mínima de 8 bytes. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

CURRENT SCHEMA
Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud del registro especial CURRENT SCHEMA. Si la función-conversión es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

Si el valor especificado no es válido, se devuelve un error (SQLSTATE 42894).

GENERATED
Indica que DB2 genera valores para la columna. Deberá especificarse GENERATED si la columna va a considerarse una columna IDENTITY.

ALWAYS
Especifica que DB2 generará siempre un valor para la columna cuando se inserte una fila en la tabla o cuando cambie el valor del resultado de la *expresión-generación*. El resultado de la expresión se almacena en la tabla. GENERATED ALWAYS es el valor recomendado a menos que se estén realizando operaciones de propagación de datos o de descarga y de recarga. GENERATED ALWAYS es el valor obligatorio para columnas generadas.

BY DEFAULT
Especifica que DB2 generará un valor para la columna cuando se inserte una fila o se actualice mediante la especificación de la cláusula DEFAULT, a menos que se especifique un valor explícito. BY DEFAULT es el valor recomendado cuando se utiliza la propagación de datos o se realiza una operación de descarga o recarga.

Aunque no se requiere explícitamente, se debe definir un índice de una sola columna exclusivo en la columna generada para garantizar la exclusividad de los valores.

AS IDENTITY
Especifica que la columna va a ser la columna de identidad para esta tabla. Una tabla puede contener una sola columna de identidad (SQLSTATE 428C1). La palabra clave IDENTITY sólo puede especificarse si el tipo de datos que se asocia a la columna es un tipo numérico exacto con una escala cero o un tipo diferenciado definido por el usuario para el que el tipo de fuente es un tipo numérico exacto con una escala cero (SQLSTATE 42815). Se consideran tipos numéricos exactos SMALLINT, INTEGER, BIGINT o DECIMAL con una escala cero o un tipo diferenciado basado en uno de estos tipos. En cambio, las comas flotantes de precisión simple o doble se consideran tipos de datos numéricos aproximados. Los tipos de referencia, aunque se hayan representando por medio de un tipo numérico exacto, no pueden definirse como columnas de identidad.

Las columnas de identidad están definidas implícitamente como no nulas (NOT NULL). Una columna de identidad no puede tener una cláusula DEFAULT (SQLSTATE 42623).

START WITH *constante-numérica*
Especifica el primer valor de la columna de identidad. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA). El valor por omisión es MINVALUE para las secuencias ascendentes y MAXVALUE para las secuencias ascendentes.

INCREMENT BY *constante-numérica*
Especifica el intervalo existente entre valores consecutivos de la columna de identidad. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), que no exceda el valor de una constante de enteros grande (SQLSTATE 42820) y sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA).

Si este valor es negativo, es una secuencia descendente. Si este valor es 0 o positivo, es una secuencia ascendente. El valor por omisión es 1.

NO MINVALUE o **MINVALUE**
Especifica el valor mínimo en el que una columna de identidad descendente ejecuta un ciclo o detiene la generación de valores o en el que una columna de identidad ascendente ejecuta un ciclo tras haberse alcanzado el valor máximo.

NO MINVALUE
Para una secuencia ascendente, el valor es el valor START WITH o bien 1 si no se ha especificado START WITH. Para una secuencia descendente, el valor es el valor mínimo del tipo de datos de la columna. Este es el valor por omisión.

MINVALUE *constante-numérica*
Especifica la constante numérica que es el valor mínimo. Este valor puede ser cualquier valor positivo o negativo

que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser menor que o igual al valor máximo (SQLSTATE 42815).

NO MAXVALUE o **MAXVALUE**
Especifica el valor máximo en el que una columna de identidad ascendente ejecuta un ciclo o detiene la generación de valores o en el que una columna de identidad descendente ejecuta un ciclo tras haberse alcanzado el valor mínimo.

NO MAXVALUE
Para una secuencia ascendente, el valor es el valor máximo del tipo de datos de la columna. Para una secuencia descendente, el valor es el valor START WITH o bien -1 si no se ha especificado START WITH. Este es el valor por omisión.

MAXVALUE *constante-numérica*
Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser mayor que o igual al valor mínimo (SQLSTATE 42815).

NO CYCLE o **CYCLE**
Especifica si esta columna de identidad debe o no seguir generando valores tras la generación de su valor máximo o de su valor mínimo.

NO CYCLE
Especifica que no se generarán valores para la columna de identidad una vez que se haya alcanzado el valor máximo o el valor mínimo. Este es el valor por omisión.

CYCLE
Especifica que se continúen generando valores para esta columna después de haber alcanzado el valor máximo o el valor mínimo. Si se utiliza esta opción, después de que una columna de identidad ascendente haya alcanzado el valor máximo, generará su valor mínimo; o después de que una secuencia descendente haya alcanzado el valor mínimo, generará su valor máximo. Los valores máximo y mínimo para la columna de identidad determinan el rango que se utiliza para el ciclo.

Cuando CYCLE está en vigor, DB2 podría generar valores duplicados para una columna de identidad. Aunque no se solicita explícitamente, debe definirse un índice de una sola columna, exclusivo, en la columna generada para garantizar la existencia de valores exclusivos, si se desean valores exclusivos. Si existe un índice exclusivo en una columna de identidad de este tipo y se genera un valor que no es exclusivo, se produce un error (SQLSTATE 23505).

NO CACHE o **CACHE**
Especifica si deben mantenerse en la memoria algunos valores preasignados, para conseguir un acceso más rápido. Si es

necesario un nuevo valor para la columna de identidad y no hay ninguno disponible en la memoria intermedia, entonces el final del nuevo bloque de memoria intermedia se debe anotar en el archivo de anotaciones. Sin embargo, cuando se necesita un nuevo valor para la columna de identidad y existe un valor no utilizado en la antememoria, la asignación de dicho valor de identidad es más rápida, pues no es necesario realizar ninguna anotación cronológica. Esta opción se utiliza para el rendimiento y el ajuste.

NO CACHE
Especifica que no se deben preasignar los valores de la columna de identidad.

Si se especifica esta opción, los valores de la columna de identidad no se colocan en la memoria intermedia. En este caso, cada petición de un valor de identidad nuevo produce E/S síncrona en las anotaciones cronológicas.

CACHE *constante-entera*
Especifica cuántos valores de la secuencia de identidad deben asignarse previamente y mantenerse en la memoria. Cuando se generan valores para la columna de identidad, la asignación previa y el almacenamiento de los valores en la antememoria reducen la E/S síncrona en el archivo de anotaciones cronológicas.

Si se necesita un nuevo valor para la columna de identidad y no existen valores no utilizados disponibles en la antememoria, la asignación del valor implica tener que esperar a que se produzca la E/S en el archivo de anotaciones cronológicas. Sin embargo, cuando se necesita un valor nuevo para la columna de identidad y existe un valor no utilizado en la antememoria, la asignación de dicho valor de identidad puede suceder más rápidamente evitando la E/S en las anotaciones cronológicas.

En caso de que se produzca una desactivación de la base de datos, realizada con normalidad o como consecuencia de una anomalía en el sistema, todos los valores de secuencia que se han colocado en la antememoria y que no se han utilizado en las sentencias confirmadas se *pierden*; es decir, nunca se utilizarán. El valor especificado para la opción CACHE es el número máximo de valores de la columna de identidad que se podrían perder en el caso de una desactivación de la base de datos. (Si una base de datos no se activa explícitamente, utilizando el mandato ACTIVATE o la API, cuando la última aplicación se desconecte de la base de datos, tendrá lugar una desactivación implícita.)

El valor mínimo es 2 (SQLSTATE 42815). El valor por omisión es CACHE 20.

NO ORDER o **ORDER**
Especifica si los valores de identidad deben o no generarse en el orden en que se solicitan.

NO ORDER
Especifica que los valores no deben generarse en el orden en el que se solicitan. Este es el valor por omisión.

ORDER
Especifica que los valores deben generarse en el orden en el que se solicitan.

GENERATED ALWAYS AS (*expresión-generación***)**
Especifica que la definición de la columna se basa en una expresión. (Si la expresión para una columna GENERATED ALWAYS incluye una función externa definida por el usuario, el cambio del ejecutable de la función (de forma que los resultados cambien argumentos determinados) puede dar como resultado la existencia de datos no coherentes. Esto puede evitarse utilizando la sentencia SET INTEGRITY para forzar la generación de nuevos valores.) La *expresión-generación* no puede contener ningunos de los elementos siguientes (SQLSTATE 42621):

- Subconsultas
- Funciones de columna
- Operaciones de eliminación de referencia o funciones DEREF
- Funciones definidas por el usuario o funciones incorporadas que no son determinantes
- Funciones definidas por el usuario mediante la opción EXTERNAL ACTION
- Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
- Variables del lenguaje principal o marcadores de parámetros
- Registros especiales
- Referencias a columnas definidas más adelante en la lista de columnas
- Referencias a otras columnas generadas

El tipo de datos de la columna se basa en el tipo de datos resultante de la *expresión-generación*. Se puede utilizar una especificación CAST para forzar un tipo de datos determinado y proporcionar un ámbito (para un tipo de referencia solamente). Si se especifica *tipo-datos*, se asignarán valores a la columna de acuerdo con las normas de asignación correspondientes. Implícitamente se considera que una columna generada puede contener nulos, a menos que se especifique la opción NOT NULL para columnas. El tipo de datos de una columna generada debe ser un tipo para el que esté definida la función de igualdad. Por lo tanto, se excluyen las columnas de tipo LONG VARCHAR, LONG VARGRAPHIC o DATALINK; los tipos de datos LOB; los tipos estructurados; y los tipos diferenciados basados en estos tipos (SQLSTATE 42962).

definición-columna-OID
Define la columna de identificador de objeto para la tabla con tipo.

REF IS *nombre-columna-OID* **USER GENERATED**
Especifica que en la tabla se define una columna de identificador de objeto (OID) como la primera columna. Se necesita un OID para la tabla raíz de una jerarquía de tablas (SQLSTATE 428DX). La tabla debe ser una tabla con tipo (debe estar presente la cláusula

OF) que no sea una subtabla (SQLSTATE 42613). El nombre de la columna se define como *nombre-columna-OID* y no puede ser el mismo que el nombre de cualquier atributo del tipo estructurado *nombre-tipo1* (SQLSTATE 42711). La columna se define con el tipo REF(*nombre-tipo1*), NOT NULL y se genera un índice de unicidad requerido por el sistema (con un nombre de índice por omisión). Esta columna viene referida como la *columna de identificador de objeto* o *columna de OID*. Las palabras clave USER GENERATED indican que el usuario debe proporcionar el valor inicial de la columna de OID cuando inserte una fila. Una vez que se haya insertado una fila, la columna de OID no podrá actualizarse (SQLSTATE 42808).

opciones-with
Define opciones adicionales que se aplican a las columnas de una tabla con tipo.

nombre-columna
Especifica el nombre de la columna para la que se especifican las opciones adicionales. El *nombre-columna* debe corresponder al nombre de una columna de la tabla que no sea además una columna de una supertabla (SQLSTATE 428DJ). Sólo puede aparecer un nombre de columna en una cláusula WITH OPTIONS de la sentencia (SQLSTATE 42613).

Si ya está especificada una opción como parte de la definición de tipo (en CREATE TYPE), las opciones especificadas aquí alteran temporalmente las opciones de CREATE TYPE.

WITH OPTIONS *opciones-columna*
Define opciones para la columna especificada. Consulte el valor *opciones-columna* descrito anteriormente. Si la tabla es una subtabla, no pueden especificarse restricciones de unicidad ni de clave primaria (SQLSTATE 429B3).

DATA CAPTURE
Indica si se debe registrar en el archivo de anotaciones cronológicas información adicional para la duplicación de datos entre bases de datos. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613).

Si la tabla es una tabla con tipo, entonces esta opción no se soporta (SQLSTATE 428DH o 42HDR).

NONE
Indica que no se va a anotar ninguna información adicional.

CHANGES
Indica que en el archivo de anotaciones cronológicas se registrará información adicional referente a los cambios de SQL efectuados en esta tabla. Esta opción es necesaria para duplicar la tabla y cuando se utiliza el programa Capture para capturar los cambios contenidos en el archivo de anotaciones para esta tabla.

Si la tabla se ha definido para admitir datos en una partición distinta de la partición de catálogo (grupo de particiones de base de datos de varias particiones o grupo de particiones de base de datos con una partición distinta de la partición de catálogo), esta opción no recibe soporte (SQLSTATE 42997).

Si el nombre de esquema (implícito o explícito) de la tabla es más largo que 18 bytes, entonces no se da soporte a esta opción (SQLSTATE 42997).

WITH RESTRICT ON DROP
Indica que la tabla no puede eliminarse y que el espacio de tablas que contiene la tabla no puede eliminarse.

IN *nombre-espacio-tablas1*
Identifica el espacio de tablas en que se va a crear la tabla. El espacio de tablas debe existir y ser un espacio de tablas REGULAR. Si no se especifica ningún otro espacio de tablas, todas las partes de la tabla se almacenarán en este espacio de tablas. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613) porque el espacio de tablas se hereda de la tabla raíz de la jerarquía de tablas. Si no se especifica esta cláusula, el espacio de la tabla se determina de la manera siguiente:

```
IF espacio de tablas IBMDEFAULTGROUP para el que el usuario
   tiene privilegio USE existe con suficiente espacio de página
     THEN elegirlo
 ELSE IF (De lo contrario, si) existe un espacio de tablas sobre el
   que el usuario tiene el privilegio (de uso) USE
   con un tamaño de página suficiente
  (vea más abajo cuándo son idóneos múltiples espacios de tabla)
     THEN elegirlo
  ELSE (De lo contrario) emita un error (SQLSTATE 42727).
```

Si la condición ELSE IF identifica más de un espacio de tablas, entonces elija el espacio de tablas con el tamaño de página suficiente más pequeño, para el cual el ID de autorización de la sentencia tenga el privilegio USE. Cuando existe más de un espacio de tablas que puede elegirse, se establece una prioridad basada en quién recibió el privilegio USE:

1. el ID de autorización
2. un grupo al cual pertenece el ID de autorización
3. PUBLIC

Si todavía existe más de un espacio de tablas que puede elegirse, el gestor de bases de datos toma la decisión final.

La determinación del espacio de tablas puede cambiar cuando:

- se eliminan o crean espacios de tablas
- se otorgan o revocan privilegios USE.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Vea "Tamaño de fila" en la página 393 para obtener más información.

opciones-espaciotablas
Especifica el espacio tabla en que se almacenarán los valores de los índices y/o de las columnas largas. Para obtener detalles acerca de los tipos de espacios de tabla, consulte "CREATE TABLESPACE".

INDEX IN *nombre-espacio-tablas2*
Identifica el espacio de tablas donde se crearán todos los índices de la tabla. Esta opción sólo es válida cuando el espacio de tablas principal que se especifica en la cláusula IN sea un espacio de tablas DMS. El espacio de tablas especificado debe existir, debe ser

un espacio de tablas REGULAR o LARGE DMS para el que el ID de autorización de la sentencia debe disponer de privilegio USE y debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tabla1* (SQLSTATE 42838).

Observe que la especificación de qué espacio de tablas contendrá el índice de una tabla sólo puede hacerse al crear la tabla. La comprobación del privilegio USE para el espacio de tablas de índice sólo se realiza al crear la tabla. El gestor de bases de datos no exigirá que el ID de autorización de una sentencia CREATE INDEX tenga el privilegio USE para el espacio de tablas cuando se cree un índice más tarde.

LONG IN *nombre-espacio-tablas3*
Identifica el espacio de tablas donde se almacenarán los valores de todas las columnas largas (tipos de datos LONG VARCHAR, LONG VARGRAPHIC y LOB, tipos diferenciados basados en cualquiera de esos tipos o columnas definidas con tipos estructurados definidos por el usuario que contienen valores que no se pueden almacenar internamente (″inline″). Esta opción sólo es válida cuando el espacio de tablas principal que se especifica en la cláusula IN sea un espacio de tablas DMS. El espacio de tablas debe existir, debe ser un espacio de tablas LARGE DMS para el que el ID de autorización de la sentencia debe disponer de privilegio USE y debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tabla1* (SQLSTATE 42838).

Tenga en cuenta que la especificación del espacio de tablas que contendrá columnas largas y LOB de una tabla sólo puede realizarse durante la creación de la tabla. La comprobación del privilegio USE para el espacio de tablas que contendrá las columnas largas y de tipo LOB sólo se realiza al crear la tabla. El gestor de bases de datos no exigirá que el ID de autorización de una sentencia ALTER TABLE tenga el privilegio USE para el espacio de tablas cuando más tarde se añada una columna larga o de tipo LOB.

PARTITIONING KEY (*nombre-columna***,...)**
Especifica la clave de particionamiento utilizada cuando se particionan los datos de la tabla. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez. Una columna no se puede utilizar como parte de una clave de particionamiento si el tipo de datos de la columna es un LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK, un tipo diferenciado basado en cualquiera de estos tipos o un tipo estructurado (SQLSTATE 42962). No puede especificarse una clave de particionamiento para una tabla que sea una subtabla (SQLSTATE 42613) porque la clave de particionamiento se hereda de la tabla raíz de la jerarquía de tablas.

Si no se especifica esta cláusula y esta tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de particionamiento se define de la forma siguiente:

- si la tabla es una tabla con tipo, la columna de identificador de objeto es la clave de particionamiento
- si se especifica una clave primaria, la primera columna de la clave primaria es la clave de particionamiento

- en otros caso, la clave de particionamiento es la primera columna cuyo tipo de datos no sea una columna LOB, LONG VARCHAR, LONG VARGRAPHIC ni DATALINK, un tipo diferenciado basado en uno de estos tipos ni una columna de tipo estructurado.

Si ninguna de las columnas satisface el requisito de la clave de particionamiento por omisión, la tabla se crea sin ninguna. Este tipo de tablas sólo está permitido en los espacios de tabla que se han definido en grupos de particiones de base de datos de una sola partición.

Para las tablas de los espacios de tabla que se han definido en grupos de particiones de base de datos de una sola partición, para definir la clave de particionamiento puede utilizarse cualquier colección de columnas de tipo no largo. Si no especifica este parámetro, no se crea ninguna clave de particionamiento.

Para ver las restricciones relacionadas con la clave de particionamiento, consulte el apartado 388.

USING HASHING
Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

REPLICATED
Especifica que los datos almacenados en la tabla se duplican físicamente en cada partición de base de datos del grupo de particiones de base de datos del espacio de tablas en el que se define la tabla. Esto significa que existe una copia de todos los datos de la tabla en cada una de estas particiones de base de datos. Esta opción sólo puede especificarse para una tabla de consultas materializadas (SQLSTATE 42997).

VALUE COMPRESSION
Especifica que los valores de datos NULL y de longitud 0 van a almacenarse de forma más eficaz para la mayoría de tipos de datos. También determina el formato de fila que va a utilizarse. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

Los valores de datos de longitud 0 para las columnas cuyo tipo de datos es BLOB, CLOB, DBCLOB, LONG VARCHAR o LONG VARGRAPHIC se almacenan utilizando el espacio mínimo. Cada valor NULL se almacena sin utilizar un byte adicional. El formato de fila que se utiliza para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos durante las actualizaciones. El nuevo formato de fila (que, para una columna, se especifica mediante la opción COMPRESS SYSTEM DEFAULT) también permite que los valores por omisión del sistema correspondientes a la columna puedan almacenarse de forma más eficaz.

NOT LOGGED INITIALLY
Los cambios realizados en la tabla por una operación de Inserción, Supresión, Actualización, Creación de índice, Eliminación de índice o Modificación de tabla durante la misma unidad de trabajo en la que se crea la tabla no se registran en el archivo de anotaciones. Para conocer

otras consideraciones que deben observarse al utilizar esta opción, consulte el apartado “Notas” de esta sentencia.

Todos los cambios de catálogo e información relativa al almacenamiento se anotan cronológicamente, así como las todas las operaciones que se realizan en la tabla en unidades de trabajo subsiguientes.

Nota: Si se produce actividad que no es de anotaciones cronológicas para una tabla que tiene activado el atributo NOT LOGGED INITIALLY y si una sentencia no se ejecuta satisfactoriamente (dando lugar a una retrotracción) o si se ejecuta ROLLBACK TO SAVEPOINT, se retrotraerá la unidad de trabajo completa (SQL1476N). Además, la tabla para la que se había activado el atributo NOT LOGGED INITIALLY se marcará como no accesible tras producirse la retrotracción y sólo podrá eliminarse. Por lo tanto, debe minimizarse toda oportunidad de errores dentro de la unidad de trabajo en la que se haya activado el atributo NOT LOGGED INITIALLY.

CCSID

Especifica el esquema de codificación para los datos de serie almacenados en la tabla. Si no se especifica la cláusula CCSID, el valor por omisión es CCSID UNICODE para bases de datos Unicode y CCSID ASCII para las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar CCSID ASCII (SQLSTATE 56031).

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8.

Si la base de datos no es Unicode, se pueden crear tablas con CCSID UNICODE, pero se aplican las normas siguientes:

- Se debe especificar el orden de clasificación alternativo en la configuración de la base de datos antes de crear la tabla (SQLSTATE 56031). Las tablas CCSID UNICODE se clasifican con el orden de clasificación alternativo especificado en la configuración de la base de datos.
- No se pueden utilizar conjuntamente las tablas o las funciones de tablas creadas con CCSID ASCII y las tablas o las funciones de tablas creadas con CCSID UNICODE en una sola sentencia de SQL (SQLSTATE 53090). Esto se aplica a las tablas y a las funciones de tabla a las que se hace referencia directamente en la sentencia, así como a las tablas y las funciones de tablas a las que se hace referencia indirectamente (por ejemplo, mediante restricciones de integridad referencial, activadores, tablas de consultas materializadas y tablas de los cuerpos de las vistas).
- No se puede hacer referencia a las tablas creadas con CCSID UNICODE en funciones de SQL o en métodos de SQL (SQLSTATE 560C0).
- Una sentencia de SQL que hace referencia a una tabla creada con CCSID UNICODE no puede invocar una función de SQL ni un método de SQL (SQLSTATE 53090).

- Los tipos gráficos y los tipos definidos por el usuario no se pueden utilizar en tablas CCSID UNICODE (SQLSTATE 560C1).
- Las tablas no pueden especificar las cláusulas CCSID UNICODE y DATA CAPTURE CHANGES a la vez (SQLSTATE 42613).
- Las tablas Explain no se pueden crear con CCSID UNICODE (SQLSTATE 55002).
- Las tablas temporales globales declaradas no se pueden crear con CCSID UNICODE (SQLSTATE 56031).
- Las tablas CCSID UNICODE no se pueden crear en una sentencia CREATE SCHEMA (SQLSTATE 53090).
- La tabla de excepciones para una operación de carga debe tener el mismo CCSID que la tabla de destino de la operación (SQLSTATE 428A5).
- La tabla de excepciones para una sentencia SET INTEGRITY debe tener el mismo CCSID que la tabla de destino para la sentencia (SQLSTATE 53090).
- La tabla de destino para los datos del supervisor de sucesos no se debe declarar como CCSID UNICODE (SQLSTATE 55049).
- Las sentencias que hacen referencia a una tabla CCSID UNICODE sólo se pueden invocar desde un cliente DB2 Versión 8.1 o posterior (SQLSTATE 42997).
- Las sentencias de SQL siempre se interpretan en la página de códigos de la base de datos. En particular, significa que cada carácter de los literales, literales hexadecimales e identificadores delimitados debe tener una representación en la página de códigos de la base de datos; de lo contrario, el carácter se sustituirá por el carácter de sustitución.

Las variables del lenguaje principal de la aplicación siempre están en la página de códigos de la aplicación, sin tener en cuenta el CCSI de ninguna tabla de las sentencias de SQL que se invocan. DB2 realizará conversiones de páginas de códigos cuando sea necesario para convertir datos entre la página de códigos de la aplicación y la página de códigos de la sección. Se puede establecer la variable de registro DB2CODEPAGE en el cliente para cambiar la página de códigos de la aplicación.

OPTIONS (ADD *nombre-opción-tabla constante-serie***, ...)**
Se utilizan opciones de tabla para identificar la tabla base remota. El *nombre-opción-tabla* es el nombre de la opción. La *constante-serie* especifica el valor para la opción de tabla. La *constante-serie* debe especificarse entre comillas simples.

El servidor remoto (el nombre de servidor que se ha especificado en la sentencia CREATE SERVER) se debe especificar en la cláusula OPTIONS. La cláusula OPTIONS también se puede utilizar para alterar temporalmente el esquema o el nombre no calificado de la tabla base remota que se crea.

Se recomienda especificar el nombre de esquema. Si no se especifica un nombre de esquema remoto, se utiliza el calificador para el nombre de tabla. Si el nombre de tabla no tiene calificador, se utiliza el ID de autorización de la sentencia.

Si no se especifica un nombre no calificado para la tabla base remota, se utiliza *nombre-tabla*.

restricción-unicidad
Define una restricción de clave primaria o de unicidad. Si la tabla tiene una

clave de particionamiento, cualquier clave de unicidad o primaria debe ser un superconjunto de la clave de particionamiento. No puede especificarse una restricción de unicidad o de clave primaria para una tabla que sea una subtabla (SQLSTATE 429B3). Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE). Si la tabla es una tabla raíz, la restricción se aplica a la tabla y a todas sus subtablas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de clave primaria o de unicidad.

UNIQUE (*nombre-columna***,...)**
Define una clave de unicidad compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 393 para obtener información acerca de las longitudes almacenadas). Las claves exclusivas con partes de clave de variable que pueden tener un tamaño superior a 255 si la variable de registro DB2_INDEX_2BYTEVARLEN se ha establecido en ON. No puede utilizarse ningún tipo LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipo diferenciado basado en uno de estos tipos o tipo estructurado como parte de una clave exclusiva, incluso si el atributo de longitud de la columna es lo suficientemente pequeño como para caber dentro del límite de 1024 bytes (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

No podrá especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave de unicidad y su índice de unicidad. Se creará automáticamente un índice de unicidad para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre de índice entra en conflicto, el nombre será SQL, seguido de los caracteres indicación (*aammddhhmmssxxx*) de la hora, con SYSIBM como el nombre de esquema.

PRIMARY KEY (*nombre-columna***,...)**
Define una clave primaria formada por las columnas identificadas. La cláusula no debe especificarse más de una vez y las columnas identificadas deben definirse como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez.

El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado "Cuentas de bytes" en la página 393 para obtener información acerca de las longitudes almacenadas). Las claves primarias con partes de clave de variable pueden tener un tamaño superior a 255 si la variable de registro DB2_INDEX_2BYTEVARLEN se ha establecido en ON. No puede utilizarse ningún tipo LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK,

tipo diferenciado basado en uno de estos tipos o tipo estructurado como parte de una clave primaria, incluso si el atributo de longitud de la columna es lo suficientemente pequeño como para caber dentro del límite de 1024 bytes (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

En una tabla sólo se puede definir una clave primaria.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave primaria y su índice principal. Se creará automáticamente un índice de unicidad para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre de índice entra en conflicto, el nombre será SQL seguido de los caracteres de la indicación (*aammddhhmmssxxx*) de la hora, con SYSIBM como nombre de esquema.

Si la tabla tiene una clave de particionamiento, las columnas de una *restricción-unicidad* deben ser un superconjunto de las columnas de la clave de particionamiento; el orden de las columnas no es importante.

restricción-referencia
Define una restricción de referencia.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de referencia.

FOREIGN KEY (*nombre-columna***,...)**
Define una restricción de referencia con el *nombre-restricción* especificado.

T1 indicará la tabla de objetos de la sentencia. La clave foránea de la restricción de referencia se compone de las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de T1, y no debe identificarse la misma columna en más de una ocasión. El número de columnas identificadas no debe exceder de 16 y la suma de sus longitudes almacenadas no debe exceder de 1024 (consulte el apartado “Cuentas de bytes” en la página 393 para conocer las longitudes almacenadas). Las claves foráneas pueden definirse en columnas de longitud variable cuya longitud sea superior a 255 bytes. La clave foránea no puede contener ningún LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, un tipo diferenciado basado en uno de estos tipos ni una columna de tipo estructurado (SQLSTATE 42962). Debe haber el mismo número de columnas de clave foránea que hay en la clave padre y los tipos de datos de las columnas correspondientes deben ser compatibles (SQLSTATE 42830). Dos descripciones de columna son compatibles si tienen tipos de datos compatibles (ambas columnas son numéricas, de series de caracteres, gráficas, fecha/hora o tienen el mismo tipo diferenciado).

cláusula-referencias
Especifica la tabla padre o el apodo padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla* o *apodo*
La tabla o apodo que se especifica en una cláusula REFERENCES debe

identificar una tabla base o un apodo que se describa en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción de referencia es un duplicado si su clave foránea, clave padre y tabla padre o apodo padre son los mismos que la clave foránea, clave padre y tabla padre o apodo padre de una restricción de referencia especificada previamente. Las restricciones de referencia duplicadas se pasan por alto y se emite un aviso (SQLSTATE 01543).

En la explicación siguiente, T2 indica la tabla padre identificada y T1 indica la tabla que está creándose (o alterándose). (T1 y T2 pueden ser la misma tabla).

La clave foránea especificada debe tener el mismo número de columnas que la tabla padre de T2 y la descripción de la columna número *n* de la clave foránea debe ser similar a la descripción de la columna número *n* de esa clave padre. Las columnas de indicación de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta norma.

(*nombre-columna***,...)**
La clave padre de la restricción de referencia se compone de las columnas identificadas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de T2. La misma columna no se puede identificar más de una vez.

La lista de nombres de columna debe coincidir con el conjunto de columnas (en cualquier orden) de la clave primaria o con una restricción de unicidad que exista en T2 (SQLSTATE 42890). Si no se especifica una lista de nombres de columna, T2 debe tener una clave primaria (SQLSTATE 42888). La omisión de la lista de nombres de columna es una especificación implícita de las columnas de dicha clave primaria en la secuencia especificada originalmente.

La restricción de referencia especificada por una cláusula FOREIGN KEY define una relación en la que T2 es padre y T1 es dependiente.

cláusula-norma
Especifica la acción que debe realizarse en las tablas dependientes.

ON DELETE
Especifica la acción que se debe emprender en las tablas dependientes al suprimir una fila de la tabla superior. Existen cuatro acciones posibles:

- NO ACTION (valor por omisión)
- RESTRICT
- CASCADE
- SET NULL

La norma de supresión se aplica cuando una fila de la T2 es el objeto de una operación de DELETE o de supresión propagada y esa fila tiene dependientes en la T1. Supongamos que *p* indica dicha fila de T2.

- Si se especifica RESTRICT o NO ACTION, se produce un error y no se suprimir ninguna fila.
- Si se especifica CASCADE, la operación de supresión se propaga a los dependientes de *p* en T1.

- Si se especifica SET NULL, cada columna con posibilidad de nulos de la clave foránea de cada dependiente de *p* en T1 se establece en nulo.

No debe especificarse SET NULL a menos que alguna columna de las claves foráneas permita valores nulos. La omisión de esta cláusula es una especificación implícita de ON DELETE NO ACTION.

Si T1 está conectada para supresión con T2 mediante varias vías de acceso, no se permitirá definir dos reglas SET NULL con definiciones de claves foráneas solapadas. Por ejemplo: T1 (i1, i2, i3). No se permite utilizar la Regla1 con la clave foránea (i1, i2) y la Regla2 con la clave foránea (i2, i3).

El orden de aplicación de las reglas es el siguiente:

1. RESTRICT
2. SET NULL OR CASCADE
3. NO ACTION

Si dos reglas distintas afectan a cualquier fila de T1, se producirá un error y no se suprimirá ninguna fila.

No se puede definir una restricción de referencia si hará que una tabla se suprima-conecte a sí misma mediante un ciclo en el que intervienen dos o más tablas y donde una de las reglas de supresión es RESTRICT o SET NULL (SQLSTATE 42915).

Se puede definir una restricción de referencia que haga que una tabla se suprima-conecte a sí misma o a otra tabla mediante varias vías de acceso, excepto en los siguientes casos (SQLSTATE 42915):

- Una tabla no debe ser una tabla dependiente en una relación de tipo CASCADE (referencia a sí misma o referencia a otra tabla) y no debe tener una relación de auto referencia en la que la regla de supresión es RESTRICT o SET NULL.
- Una clave solapa otra clave cuando al menos una columna de una clave está en la misma columna que la otra clave. Cuando una tabla se suprime-conecta a otra tabla a través de varias relaciones con claves foráneas de solapamiento, estas relaciones deben tener la misma regla de supresión y ninguna de las reglas de supresión puede ser SET NULL.
- Cuando una tabla se suprime-conecta a otra tabla a través de varias relaciones, y al menos una de estas relaciones se especifica con una regla de supresión de SET NULL, las definiciones de clave foránea de estas relaciones no deben contener ninguna clave de particionamiento ni columna de clave MDC.
- Cuando dos tablas se suprimen-conectan a la misma tabla a través de relaciones CASCADE, las dos tablas no deben suprimirse-conectarse entre ellas si la regla de supresión de la última relación en cada vía de acceso de supresión-conexión es RESTRICT o SET NULL.

Si alguna fila de T1 se ve afectada por dos reglas de supresión distintas, el resultado sería el efecto de todas las acciones especificadas por estas reglas. Los activadores AFTER y las

restricciones CHECK sobre T1 también sufrirán el efecto de todas las acciones. Un ejemplo de esto es una fila que constituye el destino de anulación a través de una vía de acceso de supresión-conexión con una tabla progenitora y que es el destino de una supresión por parte de una segunda vía de acceso de supresión-conexión con la misma tabla progenitora. El resultado sería la supresión de la fila. Los activadores AFTER DELETE sobre esta tabla descendiente se activaría, pero los activadores AFTER UPDATE no.

Cuando se aplican las normas anteriores a restricciones de referencia, en las que la tabla padre o la tabla dependiente es un miembro de una jerarquía de tablas con tipo, se tienen en cuenta todas las restricciones de referencia aplicables a cualquier tabla de sus respectivas jerarquías.

ON UPDATE
Especifica la acción que se debe emprender en las tablas dependientes al actualizar una fila de la tabla padre. La cláusula es opcional. ON UPDATE NO ACTION es el valor por omisión y ON UPDATE RESTRICT es la única alternativa.

La diferencia entre NO ACTION y RESTRICT se describe en el apartado “Notas” de esta sentencia.

atributos-restricción
Define los atributos que se asocian a las restricciones de comprobación o de integridad referencial.

ENFORCED o **NOT ENFORCED**
Especifica si el gestor de bases de datos obliga a aplicar la restricción durante operaciones normales como, por ejemplo, insertar, actualizar o suprimir. El valor por omisión es ENFORCED.

ENFORCED
El gestor de bases de datos obliga a aplicar la restricción. No se puede especificar ENFORCED para una dependencia funcional (SQLSTATE 42621). ENFORCED no se puede especificar cuando una restricción de referencia hace referencia a un apodo (SQLSTATE 428G7).

NOT ENFORCED
El gestor de bases de datos no obliga a aplicar la restricción. Sólo debe especificarse si, independientemente, se sabe que los datos de tabla se ajustan a la restricción.

ENABLE QUERY OPTIMIZATION o **DISABLE QUERY OPTIMIZATION**
Especifica si se puede utilizar la restricción o la dependencia funcional para la optimización de la consulta bajo circunstancias adecuadas. El valor por omisión es ENABLE QUERY OPTIMIZATION.

ENABLE QUERY OPTIMIZATION
Se supone que la restricción es verdadera y se puede utilizar para la optimización de la consulta.

DISABLE QUERY OPTIMIZATION
La restricción no puede utilizarse para la optimización de la consulta.

restricción-comprobación
Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o una dependencia funcional que se define entre columnas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de comprobación.

CHECK (*condición-error***)**
Define una restricción de comprobación. La *condición-búsqueda* debe ser verdadera o desconocida para cada fila de la tabla.

condición-búsqueda
La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia de columna debe hacer referencia a una columna de la tabla que está creándose.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- La *condición-búsqueda* no puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
  - Subconsultas
  - Operaciones de eliminación de referencia o funciones DEREF donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
  - Especificaciones CAST con una cláusula SCOPE
  - Funciones de columna
  - Funciones que no sean deterministas
  - Funciones definidas para que exista una acción externa
  - Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
  - Variables del lenguaje principal
  - Marcadores de parámetro
  - Registros especiales
  - Referencias a columnas generadas que no correspondan a la columna de identidad

dependencia-funcional
Define una dependencia funcional entre columnas.

nombre-columna **DETERMINED BY** *nombre-columna* o
(*nombre-columna***,...) DETERMINED BY (***nombre-columna***,...)**
El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la columna no debe ser un tipo de datos LOB, un tipo diferenciado basado en el tipo de datos LOB ni un tipo estructurado (SQLSTATE 42962). Ninguna columna del conjunto de columnas hijo puede ser una columna anulable (SQLSTATE 42621).

Si se especifica una restricción de comprobación como parte de una *definición-columna*, sólo puede establecerse una referencia de columna que haga referencia a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de tabla pueden tener referencias de columna que identifiquen columnas que ya hayan sido

definidas previamente en la sentencia CREATE TABLE. No se comprueba si hay incoherencias, condiciones duplicadas ni condiciones equivalentes en las restricciones de comprobación. Por lo tanto, se pueden definir restricciones de comprobación contradictorias o redundantes, lo que podría dar lugar a posibles errores en tiempo de ejecución.

Se puede especificar la *condición-búsqueda* "IS NOT NULL"; sin embargo, se recomienda que la anulabilidad se fuerce directamente, utilizando el atributo NOT NULL de una columna. Por ejemplo, `CHECK (salario + bonificación > 30000)` se acepta si el salario se ha establecido en NULL, ya que las restricciones CHECK deben satisfacerse o bien no conocerse y, en este caso, el salario no se conoce. Sin embargo, se consideraría que `CHECK (salario IS NOT NULL)` es falso y una violación de la restricción si el salario se ha establecido en NULL.

Las restricciones de comprobación con *condición-búsqueda* se fuerzan cuando se insertan filas en la tabla o se actualizan. Una restricción de comprobación definida en una tabla se aplica automáticamente a todas las subtablas de esta tabla.

El gestor de bases de datos no fuerza una dependencia funcional durante las operaciones normales como, por ejemplo, insertar, actualizar, suprimir o establecer integridad. La dependencia funcional puede utilizarse durante la regrabación de consultas para optimizarlas. Se pueden devolver resultados incorrectos si no se mantiene la integridad de una dependencia funcional.

Normas:

- La suma de las cuentas de bytes de las columnas, incluidas las longitudes "inline" de todas las columnas de tipo estructurado, no debe ser mayor que el límite de tamaño de la fila, que está basado en el tamaño de página del espacio de tablas (SQLSTATE 54010). Para obtener más información, consulte el apartado "Cuentas de bytes" en la página 393. Para las tablas con tipo, el número total de bytes se aplica a las columnas de la tabla raíz de la jerarquía de tabla y cada columna adicional que cada subtabla incorpora a la jerarquía de tabla (las columnas de subtabla adicionales deben considerarse como columnas con posibilidad de nulos para la obtención del número total de bytes, incluso si se han definido como columnas sin posibilidad de nulos). También hay 4 bytes adicionales de actividad general para identificar la subtabla a la que pertenece cada fila.
- El número de columnas en una tabla no puede sobrepasar la cantidad de 1.012 (SQLSTATE 54011). Para las tablas con tipo, el número total de atributos de los tipos de todas las subtablas de la jerarquía de tablas no puede sobrepasar la cantidad de 1010.
- Una columna de identificador de objeto de una tabla con tipo no puede actualizarse (SQLSTATE 42808).
- Cualquier clave de unicidad o primaria definida en la tabla debe ser un superconjunto de la clave de particionamiento (SQLSTATE 42997).
- La tabla siguiente proporciona las combinaciones soportadas de opciones DATALINK dentro de las *opciones-enlace-archivo* (SQLSTATE 42613). WRITE PERMISSION ADMIN sólo puede combinarse con READ PERMISSION DB. (También reciben soporte otras combinaciones en la cláusula RECOVERY y en la cláusula ON UNLINK.)

Tabla 5. Combinaciones válidas de opciones de control de archivo DATALINK. No recibe soporte ninguna combinación que no pueda encontrarse en esta tabla; su resultado sería SQLSTATE 42613.

INTEGRITY	READ PERMISSION	WRITE PERMISSION	RECOVERY	ON UNLINK
ALL	FS	FS	NO	No aplicable
ALL	FS	BLOCKED	NO	RESTORE
ALL	FS	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	NO	RESTORE
ALL	DB	BLOCKED	NO	DELETE
ALL	DB	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	YES	DELETE
ALL	DB	ADMIN	NO	RESTORE
ALL	DB	ADMIN	NO	DELETE
ALL	DB	ADMIN	YES	RESTORE
ALL	DB	ADMIN	YES	DELETE

- Las normas siguientes sólo se aplican a las bases de datos particionadas.
  - Las tablas compuestas sólo de columnas de los tipos LOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, tipo diferenciado basado en uno de estos tipos o tipo estructurado sólo pueden crearse en espacios de tabla definidos en grupos de particiones de base de datos de una sola partición.
  - La definición de clave de particionamiento de una tabla de un espacio de tablas que se ha definido en un grupo de particiones de base de datos de varias particiones no puede modificarse.
  - La columna de clave de particionamiento de una tabla con tipo debe ser la columna de OID.
- No se puede especificar una tabla de clúster de rango en una base de datos con varias particiones de base de datos (SQLSTATE 42997).
- Se aplican las siguientes restricciones a las tablas de clúster de rango:
  - VALUE COMPRESSION no se puede activar.
  - No se puede crear un índice de clúster.
  - No se da soporte a la modificación de la tabla para añadir una columna.
  - No se da soporte a la modificación de la tabla para cambiar el tipo de datos de una columna.
  - No se da soporte a la modificación de la tabla para cambiar PCTFREE.
  - No se da soporte a la modificación de la tabla para establecer APPEND ON.
  - Las estadísticas DETALLADAS no están disponibles.
  - No se puede utilizar el programa de utilidad de carga para llenar la tabla.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - La palabra clave CONSTRAINT puede omitirse de una *definición-columna* que defina una cláusula de referencias.
    - El *nombre-restricción* puede especificarse a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT).

- SUMMARY puede especificarse opcionalmente tras CREATE.
- DEFINITION ONLY puede especificarse en lugar de WITH NO DATA.

– Para mantener la compatibilidad con las versiones anteriores de DB2 y por coherencia:
- Se puede utilizar una coma para separar las distintas opciones de la cláusula *opciones-identidad*.

– Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
- La sintaxis siguiente se acepta como comportamiento por omisión:
  • IN nombre-base-de-datos.nombre-espacio-tabla
  • IN DATABASE nombre-base-datos
  • FOR MIXED DATA
  • FOR SBCS DATA
- También recibe soporte la sintaxis siguiente:
  • NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER

• La creación de una tabla con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

• Si se especifica una clave foránea:
  – Se invalidan todos los paquetes que estén sujetos a supresión en la tabla padre.
  – Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.

• La creación de una subtabla causa la invalidación de todos los paquetes que dependan de cualquier tabla de la jerarquía de tablas.

• Columnas VARCHAR y VARGRAPHIC que son mayores de 4.000 y 2.000 respectivamente no deben utilizarse como parámetros de entrada en funciones en esquema SYSFUN. Se producirán errores cuando se invoca la función con un valor de argumento que sobrepasa estas longitudes (SQLSTATE 22001).

• La utilización de NO ACTION o RESTRICT como normas de supresión o actualización para restricciones de referencia determina cuándo se aplica la restricción. Una norma de supresión o actualización de RESTRICT se aplicará *antes* que las demás restricciones, incluidas las restricciones de referencia con normas de modificación como CASCADE o SET NULL. Una regla de supresión o de actualización de NO ACTION se aplica *después* de otras restricciones de referencia. Un ejemplo en el que es evidente un comportamiento distinto implica la supresión de filas de una vista que se ha definido como UNION ALL de las tablas relacionadas.

```
La tabla T1 es una tabla padre de la tabla T3; norma de supresión, como
   se explica a continuación.
La tabla T2 es una tabla padre de la tabla T3; norma de supresión CASCADE.

CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2

DELETE FROM V1
```

Si la tabla T1 es una tabla padre de la tabla T3 que tiene una norma de supresión de RESTRICT, se producirá una violación de restricción (SQLSTATE 23001) si existen filas hijas para las claves padre de T1 en T3.

Si la tabla T1 es una tabla padre de la tabla T3 que tiene la norma de supresión de NO ACTION, la norma de supresión de CASCADE puede suprimir las filas hijas al suprimirse las filas de T2, antes de que se aplique la norma de supresión

de NO ACTION para las supresiones de T1. Si las supresiones de T2 no han tenido como resultado la supresión de todas las filas hijos para las claves padre de T1 en T3, se generará una violación de restricción (SQLSTATE 23504).

Observe que el SQLSTATE que se devuelve es diferente dependiendo de si la norma de supresión o actualización es RESTRICT o NO ACTION.

- Para las tablas de los espacios de tabla que se han definido en grupos de particiones de base de datos de varias particiones, en la elección de las claves de particionamiento debe tenerse en cuenta la colocación de la tabla. A continuación encontrará una lista de elementos a tener en cuenta:
  - Para la colocación, las tablas deben encontrarse en el mismo grupo de particiones de base de datos. Los espacios de tabla pueden ser distintos, pero deberán haberse definido en el mismo grupo de particiones de base de datos.
  - Las claves de particionamiento de las tablas deben tener el mismo número de columnas y las columnas de clave correspondientes deben tener particiones compatibles para la colocación.
  - La elección de la clave de particionamiento también afecta al rendimiento de las uniones. Si una tabla se une con frecuencia a otra tabla, debe tomar en consideración la posibilidad de unir la columna o columnas como una clave de particionamiento para ambas tablas.
- La cláusula NOT LOGGED INITIALLY no puede utilizarse cuando columnas DATALINK con el atributo FILE LINK CONTROL están presentes en la tabla (SQLSTATE 42613).
- La opción NOT LOGGED INITIALLY es útil para las situaciones en que se debe crear un conjunto de resultados grande con datos de una fuente alternativa (otra tabla o un archivo) y la recuperación de la tabla no es necesaria. La utilización de esta opción ahorrará la actividad general de anotar cronológicamente los datos. Las siguientes consideraciones se aplican cuando se especifica esta opción:
  - Cuando se confirma la unidad de trabajo, todos los cambios que se han realizado en la tabla durante la unidad de trabajo fluyen al disco.
  - Cuando ejecuta el programa de utilidad de avance (Rollforward) y éste detecta un registro de anotaciones cronológicas que indica que el programa de utilidad de carga (Load) ha llenado una tabla de la base de datos o que ésta se ha creado con la opción NOT LOGGED INITIALLY, la tabla se marcará como no disponible. El programa de utilidad de avance (Rollforward) eliminará la tabla si posteriormente encuentra una anotación cronológica DROP TABLE. De lo contrario, después de recuperar la base de datos, se emitirá un error si se realiza un intento de acceder a la tabla (SQLSTATE 55019). La única operación permitida es eliminar la tabla.
  - Cuando se ha hecho copia de seguridad de la tabla como parte de una copia de seguridad de la base de datos o del espacio de tablas, se puede recuperar la tabla.
- Una tabla de consultas materializadas mantenida por el sistema REFRESH DEFERRED definida con ENABLE QUERY OPTIMIZATION puede utilizarse para optimizar el proceso de las consultas si CURRENT REFRESH AGE se ha establecido en ANY y si CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se ha establecido de forma que incluya tablas de consultas materializadas mantenidas por el sistema. Una tabla de consultas materializadas mantenida por el usuario REFRESH DEFERRED definida con ENABLE QUERY OPTIMIZATION puede utilizarse para optimizar el proceso de las consultas si CURRENT REFRESH AGE se ha establecido en ANY y si CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se ha establecido de forma que incluya tablas de consultas materializadas mantenidas por el usuario. Para la optimización, siempre se considera una tabla de consultas materializadas

REFRESH IMMEDIATE definida con ENABLE QUERY OPTIMIZATION. Para que esta optimización pueda utilizar una tabla de consultas materializadas REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa debe ajustarse a determinadas normas, además de las normas que ya se han descrito. La selección completa:

- debe ser una subselección con una cláusula GROUP BY o con una sola tabla de referencia
- no debe incluir DISTINCT en ninguna parte dentro de la lista de selección
- no debe incluir registros especiales
- no debe incluir funciones que no sean deterministas.

Si la consulta que se ha especificado al crear una tabla de consultas materializadas no se ajusta a estas normas, se devolverá un mensaje de aviso (SQLSTATE 01633).

- Si la tabla de consultas materializadas se define con REFRESH IMMEDIATE, o una tabla de etapas se define con PROPAGATE IMMEDIATE, es posible que se produzca un error al intentar aplicar el cambio resultante de una operación de inserción, actualización o supresión en una tabla subyacente. El error provocará la anomalía de la operación de inserción, actualización o supresión de la tabla subyacente.
- Las tablas de consultas materializadas o las tablas de etapas no se pueden utilizar como tablas de excepción cuando las restricciones se comprueban masivamente, por ejemplo, durante las operaciones da carga o durante la ejecución de la sentencia SET INTEGRITY.
- Algunas operaciones no se pueden realizar en la tabla a la que hace referencia una tabla de consultas materializadas definida con REFRESH IMMEDIATE, o definida con REFRESH DEFERRED con una tabla de etapas asociada:
  - No se puede utilizar IMPORT REPLACE.
  - No se puede llevar a cabo ALTER TABLE NOT LOGGED INITIALLY WITH EMPTY TABLE.
- En un sistema federado, los apodos para las fuentes de datos relacionales o las tablas locales pueden utilizarse como tablas subyacentes para crear una tabla de consultas materializadas. Los apodos para las fuentes de datos no relacionales no están soportados. Cuando un apodo es una de las tablas subyacentes, debe utilizarse la opción REFRESH DEFERRED. No se da soporte a las tablas de consultas materializadas mantenidas por el sistema que hacen referencia a apodos en un entorno de bases de datos particionadas.
- ***DDL transparente:*** En un sistema federado, se puede crear, modificar o descartar una tabla base remota utilizando DB2 UDB SQL. Esta posibilidad se conoce como *DDL transparente*. Para poder crear una tabla base remota en una fuente de datos, el servidor federado debe estar configurado para acceder a esa fuente de datos. Esta configuración incluye la creación de un reiniciador para la fuente de datos, el suministro de la definición del servidor para el servidor en el que se ubicará la tabla base remota y la creación de correlaciones de usuario entre el servidor federado y la fuente de datos.

  El DDL transparente impone algunas limitaciones en los elementos que pueden incluirse en la sentencia CREATE TABLE:

  - Sólo se pueden crear columnas y una clave primaria en la tabla base remota.
  - La fuente de datos remota debe dar soporte a:
    - Los tipos de datos de la columna remota con los que están correlacionados los tipos de datos de la columna DB2
    - La opción de clave primaria de la sentencia CREATE TABLE

Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.

Cuando se crea una tabla base remota utilizando un DDL transparente, se crea automáticamente un apodo para esta tabla base remota.

- Se puede definir una restricción de referencia de forma que la tabla padre o la tabla dependiente forme parte de una jerarquía de tablas. En este caso, el efecto de la restricción de referencia es el siguiente:
  1. Efectos de las sentencias INSERT, UPDATE y DELETE:
     - Si existe una restricción de referencia, en la que TP es una tabla padre y TD es una tabla dependiente, la restricción asegura que para cada fila de TD (o de cualquiera de sus subtablas) que tenga una clave foránea no nula, existe una fila en TP (o en una de sus subtablas) con una clave padre coincidente. Esta norma se aplica para cualquier acción que afecte a una fila de TP o TD, con independencia de cómo se inicie la acción.
  2. Efectos sobre las sentencias DROP TABLE:
     - para las restricciones de referencia en las que la tabla eliminada es la tabla padre o la tabla dependiente, se elimina la restricción
     - para las restricciones de referencia en las que la tabla padre es una supertabla de la tabla eliminada, se consideran eliminadas de la supertabla las filas de la tabla eliminada. Se comprueba el cumplimiento de la restricción de referencia y su norma de supresión se invoca para fila suprimida.
     - para las restricciones de referencia en las que la tabla dependiente es una supertabla de la tabla eliminada, no se comprueba el cumplimiento de la restricción. La supresión de una fila de una tabla dependiente no puede producir una violación de una restricción de referencia.
- ***Privilegios:*** Cuando se crea una tabla, se otorga el privilegio CONTROL al definidor de la tabla. Cuando se crea una subtabla, el definidor de la tabla otorga automáticamente para la subtabla el privilegio SELECT que cada usuario o grupo tiene sobre la supertabla inmediata.
- ***Tamaño de fila:*** El número máximo de bytes permitidos en la fila de una tabla depende del tamaño de página del espacio de tablas donde se crea la tabla (*nombre-espacio-tablas1*).La lista siguiente muestra el límite del tamaño de fila y el límite del número de columnas asociados a cada tamaño de página del espacio de tablas.

Tabla 6. Límites para el número de columnas y el tamaño de fila de cada tamaño de página del espacio de tabla

Tamaño de página	Límite del tamaño de fila	Límite de la cuenta de columnas
4K	4 005	500
8K	8 101	1 012
16K	16 293	1 012
32K	32 677	1 012

El número de columnas real para una tabla puede limitarse más mediante la fórmula siguiente:

```
Columnas en total * 8 + Número de columna LOB * 12 +
  Número de columnas Datalink * 28 <= límite de tamaño de fila para el
  tamaño de página.
```

- ***Cuentas de bytes:*** La tabla siguiente contiene el número total de bytes de las columnas, por tipo de datos, que corresponde a las columnas que no admiten

valores nulos. En las tablas que no tienen compresión de valor, cada columna que admite nulos necesita un byte adicional.

Si una tabla se basa en un tipo estructurado, se reservarán 4 bytes adicionales de actividad general para identificar las filas de subtablas, con independencia de si se han definido o no subtablas. Las columnas de subtabla adicionales deben considerarse como columnas con posibilidad de nulos para la obtención del número total de bytes, incluso si se han definido como columnas sin posibilidad de nulos.

Cuando se calculan las longitudes almacenadas, para garantizar que no se exceda el límite de 1024 bytes de un índice o de las restricciones (tenga en cuenta que las restricciones se aplican por medio de índices), a la actividad general corresponden 2 bytes en lugar de 4 bytes.

Tabla 7. Número total de bytes de las columnas por tipo de datos

Tipo de datos	Número total de bytes cuando se ha activado VALUE COMPRESSION para la tabla	Número total de bytes cuando no se ha activado VALUE COMPRESSION, implícita o explícitamente, para la tabla; si la columna tiene posibilidad de nulos, añada 1 al número total de bytes que se indica
ROW OVERHEAD	2	0
INTEGER	6	4
SMALLINT	4	2
BIGINT	10	8
REAL	6	4
DOUBLE	10	8
DECIMAL	La parte integrante de (p/2)+3, donde p es la precisión	La parte integrante de (p/2)+1, donde p es la precisión
CHAR (n)	n+2	n
VARCHAR (n)	n+2	n+4 (dentro de una tabla); n+2 (dentro de un índice)
LONG VARCHAR	22	24
GRAPHIC (n)	n*2+2	n*2
VARGRAPHIC (n)	(n*2)+2	(n*2)+4 (dentro de una tabla); (n*2)+2 (dentro de un índice)
LONG VARGRAPHIC	22	24
DATE	6	4
TIME	5	3
TIMESTAMP	12	10
DATALINK(n)	n+52	n+54
Longitud máxima de LOB 1024	70[1]	72
Longitud máxima de LOB 8192	94	96
Longitud máxima de LOB 65.536	118	120

Tabla 7. Número total de bytes de las columnas por tipo de datos (continuación)

Tipo de datos	Número total de bytes cuando se ha activado VALUE COMPRESSION para la tabla	Número total de bytes cuando no se ha activado VALUE COMPRESSION, implícita o explícitamente, para la tabla; si la columna tiene posibilidad de nulos, añada 1 al número total de bytes que se indica
Longitud máxima de LOB 524 000	142	144
Longitud máxima de LOB 4 190 000	166	168
Longitud máxima de LOB 134 000 000	198	200
Longitud máxima de LOB 536 000 000	222	224
Longitud máxima de LOB 1 070 000 000	254	256
Longitud máxima de LOB 1 470 000 000	278	280
Longitud máxima de LOB 2 147 483 647	314	316

[1] Cada valor de LOB tiene un *descriptor de LOB* en el registro base que apunta a la ubicación del valor real. El tamaño de dicho descriptor varía en función de la longitud máxima que se haya definido para la columna.

Para un *tipo diferenciado*, el número total de bytes equivale a la longitud del tipo de fuente del tipo diferenciado. Para un *tipo de referencia*, el número total de bytes equivale a la longitud del tipo de datos incorporado en el que se basa el tipo de referencia. Para un *tipo estructurado*, el número total de byte equivale a INLINE LENGTH + 4. El valor INLINE LENGTH es el valor que se especifica (o que se calcula implícitamente) para la columna de la cláusula *opciones-columna*.

- ***Columnas de dimensión:*** Puesto que cada valor diferenciado de una columna de dimensión se asigna a un bloque distinto de la tabla, puede que se desee aplicar un clúster en una expresión como, por ejemplo, "INTEGER(ORDER_DATE)/100". En este caso, puede definirse una columna generada para la tabla y, a continuación, esta columna generada puede utilizarse en la cláusula ORGANIZE BY DIMENSIONS. Si la expresión es monotónica respecto a una columna de la tabla, DB2 puede utilizar el índice de dimensiones para satisfacer los predicados de rango en esa columna. Por ejemplo, si la expresión es simplemente *nombre-columna + alguna-constante-positiva*, tiene lugar un incremento monotónico. Las funciones definidas por el usuario, determinadas funciones incorporadas y la utilización de más de una columna en una expresión impiden que tenga lugar la monotonicidad o su detección.

  Las dimensiones que implican columnas generadas cuyas expresiones son no monotónicas o cuya monotonicidad no puede determinarse, pueden seguir creándose, pero las consultas de rango, junto con los límites de porción o de célula, de estas dimensiones no reciben soporte. La igualdad y los predicados IN *pueden* procesarse por medio de porciones o células.

  Una columna generada será monotónica si se cumple lo siguiente respecto a la función generadora, fn:

  – Incremento monotónico.

Para cada par de valores x1 y x2 posible, si x2>x1, entonces fn(x2)>fn(x1). Por ejemplo:

```
SALARY - 10000
```

– Reducción monotónica.

  Para cada par de valores x1 y x2 posible, si x2>x1, entonces fn(x2)<fn(x1). Por ejemplo:

```
-SALARY
```

– Sin reducción monotónica.

  Para cada par de valores x1 y x2 posible, si x2>x1, entonces fn(x2)>=fn(x1). Por ejemplo:

```
SALARY/1000
```

– Sin incremento monotónico.

  Para cada par de valores x1 y x2 posible, si x2>x1, entonces fn(x2)<=fn(x1). Por ejemplo:

```
-SALARY/1000
```

La expresión ″PRICE*DISCOUNT″ no es monotónica, porque implica a más de una columna de la tabla.

- ***Tablas de clúster de rango:*** La organización de una tabla por secuencia de clave es efectivo para determinados tipos de tablas. La tabla debe tener una clave de entero que esté estrechamente agrupada (densa) sobre el rango de valores posibles. Las columnas de esta clave de entero no se deben poder anular y la clave debe ser lógicamente la clave principal de la tabla. La organización de una tabla de clúster de rango precede la necesidad de disponer de un objeto de índice exclusivo separado, proporcionando acceso directo a la fila para un valor de clave especificado o un rango de filas para un rango especificado de valores de clave. La asignación de todo el espacio correspondiente a un conjunto completo de filas en el rango de secuencia de clave definido se realiza durante la creación de la tabla y se debe tener en cuenta al definir una tabla de clúster de rango. El espacio de almacenamiento no está disponible para ningún otro uso, aunque las filas estén inicialmente marcadas para su supresión. Si el rango completo de secuencia de clave se va a llenar sólo con datos durante un largo periodo de tiempo, esta organización de tabla probablemente no constituya una opción adecuada.

Ejemplos:

Ejemplo 1: Cree la tabla TDEPT en el espacio de tablas DEPARTX. DEPTNO, DEPTNAME, MGRNO y ADMRDEPT son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. NOT NULL significa que la columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. La clave primaria es la columna DEPTNO.

```
CREATE TABLE TDEPT
  (DEPTNO   CHAR(3)     NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO    CHAR(6),
   ADMRDEPT CHAR(3)     NOT NULL,
   PRIMARY KEY(DEPTNO))
IN DEPARTX
```

Ejemplo 2: Cree la tabla PROJ en el espacio de tablas SCHED. PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE y MAJPROJ son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. DECIMAL significa que la columna contendrá datos decimales

empaquetados. 5,2 significa lo siguiente: 5 indica el número de dígitos decimales y 2 indica el número de dígitos a la derecha de la coma decimal. NOT NULL significa que la columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. DATE significa que la columna contendrá información de fecha en un formato de tres partes (año, mes y día).

```
CREATE TABLE PROJ
  (PROJNO   CHAR(6)      NOT NULL,
   PROJNAME VARCHAR(24)  NOT NULL,
   DEPTNO   CHAR(3)      NOT NULL,
   RESPEMP  CHAR(6)      NOT NULL,
   PRSTAFF  DECIMAL(5,2)         ,
   PRSTDATE DATE                 ,
   PRENDATE DATE                 ,
   MAJPROJ  CHAR(6)      NOT NULL)
IN SCHED
```

Ejemplo 3: Cree una tabla llamada EMPLOYEE_SALARY donde los salarios desconocidos se consideren 0. No se especifica ningún espacio de tablas, de modo que la tabla se creará en un espacio de tablas seleccionado por el sistema basándose en las normas descritas para la cláusula *IN nombre-espacio-tablas1*.

```
CREATE TABLE EMPLOYEE_SALARY
  (DEPTNO   CHAR(3)      NOT NULL,
   DEPTNAME VARCHAR(36)  NOT NULL,
   EMPNO    CHAR(6)      NOT NULL,
   SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT)
```

Ejemplo 4: Cree tipos diferenciados para el total de salario y millas (kilómetros) y utilícelos para las columnas de una tabla creada en el espacio de tablas por omisión. En una sentencia de SQL dinámica, suponga que el registro especial CURRENT SCHEMA es JOHNDOE y el registro especial CURRENT PATH tiene el valor por omisión ("SYSIBM","SYSFUN","JOHNDOE").

Si no se especifica un valor para SALARY, deberá establecerse en 0 y si no se especifica un valor para LIVING_DIST, deberá establecerse en 1 milla (1,6 Km).

```
CREATE DISTINCT TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS

CREATE DISTINCT TYPE JOHNDOE.MILES  AS FLOAT WITH COMPARISONS

CREATE TABLE EMPLOYEE
  (ID          INTEGER NOT NULL,
   NAME        CHAR (30),
   SALARY      T_SALARY NOT NULL WITH DEFAULT,
   LIVING_DIST MILES    DEFAULT MILES(1) )
```

Ejemplo 5: Cree tipos diferenciados para la imagen y audio y utilícelos para las columnas de una tabla. No se ha especificado ningún espacio de tablas para que la tabla se cree en un espacio de tablas que el sistema selecciona basándose en las normas que se describen para la cláusula IN *nombre-espacio-tabla1*. Suponga que el registro especial CURRENT PATH tiene el valor por omisión.

```
CREATE DISTINCT TYPE IMAGE AS BLOB (10M)

CREATE DISTINCT TYPE AUDIO AS BLOB (1G)

CREATE TABLE PERSON
  (SSN    INTEGER NOT NULL,
   NAME   CHAR (30),
   VOICE  AUDIO,
   PHOTO  IMAGE)
```

Ejemplo 6: Cree la tabla EMPLOYEE en el espacio de tablas HUMRES. Las restricciones definidas en la tabla son las siguientes:

- Los valores del número de departamento deben estar comprendidos entre 10 y 100.
- El trabajo de un empleado sólo puede ser 'Sales', 'Mgr' o 'Clerk'.
- Aquellos empleados que lleven en la compañía desde 1986 deben ganar más de 40.500 dólares.

Nota: Si las columnas incluidas en las restricciones de comprobación pueden contener valores nulos, también podrían ser NULL.

```
CREATE TABLE EMPLOYEE
  (ID          SMALLINT NOT NULL,
   NAME        VARCHAR(9),
   DEPT        SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
   JOB         CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
   HIREDATE    DATE,
   SALARY      DECIMAL(7,2),
   COMM        DECIMAL(7,2),
   PRIMARY KEY (ID),
   CONSTRAINT  YEARSAL CHECK (YEAR(HIREDATE) > 1986
     OR SALARY > 40500)
  )
  IN HUMRES
```

Ejemplo 7: Cree una tabla que esté completamente contenida en el espacio de tablas PAYROLL.

```
CREATE TABLE EMPLOYEE .....
  IN PAYROLL
```

Ejemplo 8: Cree una tabla con la parte de los datos en ACCOUNTING y la parte del índice en ACCOUNT_IDX.

```
CREATE TABLE SALARY.....
  IN ACCOUNTING INDEX IN ACCOUNT_IDX
```

Ejemplo 9: Cree una tabla y anote cronológicamente los cambios SQL en el formato por omisión.

```
CREATE TABLE SALARY1 .....
```

o

```
CREATE TABLE SALARY1 .....
  DATA CAPTURE NONE
```

Ejemplo 10: Cree una tabla y anote cronológicamente los cambios SQL en un formato expandido.

```
CREATE TABLE SALARY2 .....
  DATA CAPTURE CHANGES
```

Ejemplo 11: Cree una tabla EMP_ACT en el espacio de tablas SCHED. EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE y EMENDATE son nombres de columna. Las restricciones definidas en la tabla son:

- El valor para el conjunto de columnas EMPNO, PROJNO y ACTNO en cualquier fila debe ser exclusivo.
- El valor de PROJNO debe coincidir con un valor existente para la columna PROJNO de la tabla PROJECT y, si el proyecto se suprime, todas las filas que hacen referencia al proyecto en EMP_ACT también deben suprimirse.

```
CREATE TABLE EMP_ACT
  (EMPNO       CHAR(6) NOT NULL,
   PROJNO      CHAR(6) NOT NULL,
   ACTNO       SMALLINT NOT NULL,
   EMPTIME     DECIMAL(5,2),
   EMSTDATE    DATE,
   EMENDATE    DATE,
   CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),
   CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)
                          REFERENCES PROJECT (PROJNO) ON DELETE CASCADE
  )
  IN SCHED
```

Se crea automáticamente un índice exclusivo denominado EMP_ACT_UNIQ en el mismo esquema para aplicar la restricción de unicidad.

Ejemplo 12: Cree una tabla que vaya a contener información sobre goles famosos para el 'hall of fame' del hockey sobre hielo. La tabla listará información sobre el jugador que marcó el gol, el portero contra el que lo marcó, la fecha y el lugar, así como una descripción. Cuando se disponga de ello, también señalará los lugares en que se almacenan artículos de prensa sobre el partido e imágenes del gol fijas y en movimiento. Los artículos de prensa van a enlazarse, por lo que no pueden suprimirse ni renombrarse, pero deben seguir funcionando todas las aplicaciones de actualización y visualización existentes. Las imágenes fijas y en movimiento van a enlazarse con acceso bajo el control completo de DB2. Las imágenes fijas tendrán recuperación y se devolverán al propietario original si se desenlazan. Las imágenes en movimiento no tendrán recuperación y se suprimirán si se desenlazan. La columna de descripción y las tres columnas DATALINK tienen posibilidad de nulos.

```
CREATE TABLE HOCKEY_GOALS
  ( BY_PLAYER      VARCHAR(30)   NOT NULL,
    BY_TEAM        VARCHAR(30)   NOT NULL,
    AGAINST_PLAYER VARCHAR(30)   NOT NULL,
    AGAINST_TEAM   VARCHAR(30)   NOT NULL,
    DATE_OF_GOAL   DATE          NOT NULL,
    DESCRIPTION    CLOB(5000),
    ARTICLES       DATALINK  LINKTYPE URL FILE LINK CONTROL MODE DB2OPTIONS,
    SNAPSHOT       DATALINK  LINKTYPE URL FILE LINK CONTROL
                             INTEGRITY ALL
                             READ PERMISSION DB WRITE PERMISSION BLOCKED
                             RECOVERY YES ON UNLINK RESTORE,
    MOVIE          DATALINK  LINKTYPE URL FILE LINK CONTROL
                             INTEGRITY ALL
                             READ PERMISSION DB WRITE PERMISSION BLOCKED
                             RECOVERY NO ON UNLINK DELETE )
```

Ejemplo 13: Supongamos que se necesita una tabla de excepción para la tabla EMPLOYEE. Se puede crear una utilizando la sentencia siguiente.

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
  (SELECT EMPLOYEE.*,
    CURRENT TIMESTAMP AS TIMESTAMP,
    CAST ('' AS CLOB(32K)) AS MSG
  FROM EMPLOYEE
  ) WITH NO DATA
```

Ejemplo 14: Supongamos los espacios de tablas siguientes que tienen los atributos indicados:

```
TBSPACE            PAGESIZE    USER   USERAUTH
------------------ ----------- ------ --------
DEPT4K                    4096 BOBBY  S
PUBLIC4K                  4096 PUBLIC S
```

```
DEPT8K                   8192 BOBBY  S
DEPT8K                   8192 RICK   S
PUBLIC8K                 8192 PUBLIC S
```

- Si RICK crea la tabla siguiente, se colocará en el espacio de tablas PUBLIC4K, pues el número de bytes es menor que 4005; pero si BOBBY crea la misma tabla, se colocará en el espacio de tablas DEPT4K, pues BOBBY tiene un privilegio USE otorgado explícitamente:

```
CREATE TABLE DOCUMENTS
  (SUMMARY    VARCHAR(1000),
   REPORT     VARCHAR(2000))
```

- Si BOBBY crea la tabla siguiente, se colocará en el espacio de tablas DEPT8K, pues el número de bytes es mayor que 4005, y BOBBY tiene un privilegio USE otorgado explícitamente. En cambio, si DUNCAN crea la misma tabla, se colocará en el espacio de tablas PUBLIC8K, pues DUNCAN no tiene ningún privilegio específico:

```
CREATE TABLE CURRICULUM
  (SUMMARY    VARCHAR(1000),
   REPORT     VARCHAR(2000),
   EXERCISES  VARCHAR(1500))
```

Ejemplo 15: Creación de una tabla que tiene una columna LEAD definida con el tipo estructurado EMP. Especifique 300 bytes para el valor INLINE LENGTH de la columna LEAD, lo cual significa que cualquier instancia de LEAD que no pueda caber dentro de los 300 bytes se almacenará fuera de la tabla (separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB).

```
CREATE TABLE PROJECTS (PID INTEGER,
  LEAD EMP INLINE LENGTH 300,
  STARTDATE DATE,
            ...)
```

Ejemplo 16: Creación de una tabla DEPT, con cinco columnas llamadas DEPTNO, DEPTNAME, MGRNO, ADMRDEPT y LOCATION. La columna DEPT debe definirse como columna de identidad para que DB2 genere siempre un valor para ella. Los valores de la columna DEPT deben comenzar en 500 y aumentar según incrementos de 1.

```
CREATE TABLE DEPT
  (DEPTNO     SMALLINT      NOT NULL
                 GENERATED ALWAYS AS IDENTITY
                   (START WITH 500, INCREMENT BY 1),
   DEPTNAME   VARCHAR(36)   NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT   SMALLINT      NOT NULL,
   LOCATION   CHAR(30))
```

Ejemplo 17: Cree la tabla SALES que está particionada en la columna YEAR y que tiene dimensiones en las columnas REGION y YEAR. Los datos se distribuirán por las particiones según los valores de generación aleatoria de la columna YEAR. En cada partición, los datos se organizarán en extensiones basadas en combinaciones exclusivas de los valores de las columnas REGION y YEAR de esas particiones.

```
CREATE TABLE SALES
  (CUSTOMER   VARCHAR(80),
   REGION     CHAR(5),
   YEAR       INTEGER)
PARTITIONING KEY (YEAR)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
```

Ejemplo 18: Cree una tabla SALES con la columna PURCHASEYEARMONTH que se genera a partir de la columna PURCHASEDATE. Utilice una expresión para

crear una columna que sea monotónica con respecto a la columna PURCHASEDATE original y, por lo tanto, que sea adecuada para utilizarla como dimensión. La tabla se particiona en la columna REGION y, cada partición, se organiza en extensiones según la columna PURCHASEYEARMONTH; es decir, las distintas regiones estarán en particiones diferentes y los distintos meses de compras pertenecerán a celdas (o conjunto de extensiones) diferentes de esas particiones.

```
CREATE TABLE SALES
  (CUSTOMER            VARCHAR(80),
   REGION              CHAR(5),
   PURCHASEDATE        DATE,
   PURCHASEYEARMONTH   INTEGER
                         GENERATED ALWAYS AS (INTEGER(PURCHASEDATE)/100))
PARTITIONING KEY (REGION)
ORGANIZE BY DIMENSIONS (PURCHASEYEARMONTH)
```

Ejemplo 19: Cree la tabla CUSTOMER con la columna CUSTOMERNUMDIM que se genera a partir de la columna CUSTOMERNUM. Utilice una expresión para crear una columna que sea monotónica con respecto a la columna CUSTOMERNUM original y, por lo tanto, que sea adecuada para utilizarla como dimensión. La tabla se organiza en celdas según la columna CUSTOMERNUMDIM, de modo que hay una celda diferente en la tabla para cada 50 clientes. Si se crease un índice exclusivo en CUSTOMERNUM, los números de cliente se contendrían en clústeres de tal manera que, cada conjunto de 50 valores se encontraría en un conjunto de extensiones en particular de la tabla.

```
CREATE TABLE CUSTOMER
  (CUSTOMERNUM      INTEGER,
   CUSTOMERNAME     VARCHAR(80),
   ADDRESS          VARCHAR(200),
   CITY             VARCHAR(50),
   COUNTRY          VARCHAR(50),
   CODE             VARCHAR(15),
   CUSTOMERNUMDIM   INTEGER
                      GENERATED ALWAYS AS (CUSTOMERNUM/50))
ORGANIZE BY DIMENSIONS (CUSTOMERNUMDIM)
```

Ejemplo 20: Cree una tabla base remota denominada EMPLOYEE en el servidor Oracle, ORASERVER. También se creará automáticamente un apodo, denominado EMPLOYEE, que hará referencia a esta tabla base remota que acaba de crear.

```
CREATE TABLE EMPLOYEE
  (EMP_NO       CHAR(6)       NOT NULL,
   FIRST_NAME   VARCHAR(12)   NOT NULL,
   MID_INT      CHAR(1)       NOT NULL,
   LAST_NAME    VARCHAR(15)   NOT NULL,
   HIRE_DATE    DATE,
   JOB          CHAR(8),
   SALARY       DECIMAL(9,2),
   PRIMARY KEY (EMP_NO))
OPTIONS
  (REMOTE_SERVER 'ORASERVER',
   REMOTE_SCHEMA 'J15USER1',
   REMOTE_TABNAME 'EMPLOYEE')
```

Las sentencias CREATE TABLE siguientes muestran cómo se debe especificar el nombre de tabla o el nombre de tabla y el nombre de tabla base remota explícito, para obtener la situación deseada. El identificador en minúsculas, employee, se utiliza para ilustrar la conversión implícita de identificadores.

Cree una tabla base remota denominada EMPLOYEE (caracteres en mayúsculas) en un servidor Informix, y cree un apodo denominado EMPLOYEE (caracteres en mayúsculas) en esa tabla:

```
CREATE TABLE employee
  (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* no está delimitado, el nombre de tabla base remota aparecerá en letras mayúsculas, incluso si la fuente de datos remota almacena los nombres en minúsculas.

Cree una tabla base remota denominada employee (caracteres en minúsculas) en un servidor Informix, y cree un apodo denominado EMPLOYEE (caracteres en mayúsculas) en esa tabla:

```
CREATE TABLE employee
  (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER',
   REMOTE_TABNAME 'employee')
```

Al crear una tabla en una fuente de datos remota que da soporte a identificadores delimitados, utilice la opción REMOTE_TABNAME y una constante de serie de caracteres que especifique el nombre de tabla en mayúsculas o minúsculas, según desee.

Cree una tabla base remota denominada employee (caracteres en minúsculas) en un servidor Informix, y cree un apodo denominado employee (caracteres en minúsculas) en esa tabla:

```
CREATE TABLE "employee"
  (EMP_NO CHAR(6) NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* está delimitado, el nombre de tabla base remota será idéntico a *nombre-tabla*.

Ejemplo 21: Cree una tabla de clúster de rango que se pueda utilizar para localizar un estudiante utilizando un ID de estudiante. Para cada registro de estudiante, incluya el ID de escuela, ID de programa, número de estudiante, ID de estudiante, nombre de estudiante, apellido de estudiante y promedio de puntos del curso de estudiante (GPA).

```
CREATE TABLE STUDENTS
  (SCHOOL_ID     INTEGER   NOT NULL,
   PROGRAM_ID    INTEGER   NOT NULL,
   STUDENT_NUM   INTEGER   NOT NULL,
   STUDENT_ID    INTEGER   NOT NULL,
   FIRST_NAME    CHAR(30),
   LAST_NAME     CHAR(30),
   GPA           DOUBLE)
ORGANIZE BY KEY SEQUENCE
  (STUDENT_ID
    STARTING FROM 1
    ENDING AT 1000000)
  DISALLOW OVERFLOW
```

El tamaño de cada registro es la suma de las columnas, más alineación, más cabecera de fila de tabla de clúster de rango. En este caso, el tamaño de fila es 98 bytes: 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (para columnas que se pueden anular) + 1 (para alineación) + 10 (para la cabecera). Con un tamaño de página de 4 KB (o 4096 bytes), después de contabilizar la actividad general de página, hay 4038 bytes disponibles, lo que constituye espacio suficiente para 41 registros por página. Si se permiten 1 millón de registros de estudiantes, se necesitan (1 millón dividido por 41 registros por página) 24.391 páginas. Con dos páginas adicionales por actividad general de tabla, el número final de páginas de 4 KB que se asignan cuando se crea la tabla es 24.393.

Ejemplo 22: Cree una tabla denominada DEPARTMENT con una dependencia funcional que no tenga especificada ningún nombre de restricción.

```
CREATE TABLE DEPARTMENT
  (DEPTNO     SMALLINT      NOT NULL,
   DEPTNAME   VARCHAR(36)   NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT   SMALLINT      NOT NULL,
   LOCATION   CHAR(30),
  CHECK (DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED)
```

Conceptos relacionados:

- "Multidimensional clustering tables" en la publicación *Administration Guide: Planning*
- "What is transparent DDL?" en la publicación *Federated Systems Guide*

Tareas relacionadas:

- "Creating new remote tables using transparent DDL" en la publicación *Federated Systems Guide*

Información relacionada:

- "Subselección" en la publicación *Consulta de SQL, Volumen 1*
- "ALTER TABLE" en la página 46
- "CREATE TABLESPACE" en la página 405
- "DECLARE GLOBAL TEMPORARY TABLE" en la página 489
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*
- "Tipos de datos compatibles entre particiones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtudt.c -- How to create, use, and drop user-defined distinct types."
- "tbconstr.c -- How to work with constraints associated with tables"
- "tbcreate.c -- How to create, alter and drop tables"
- "dtudt.sqc -- How to create, use, and drop user-defined distinct types (C)"
- "tbconstr.sqc -- How to create, use, and drop constraints (C)"
- "tbcreate.sqc -- How to create and drop tables (C)"
- "tbident.sqc -- How to use identity columns (C)"
- "tbtrig.sqc -- How to use a trigger on a table (C)"
- "dtudt.sqC -- How to create, use, and drop user-defined distinct types (C++)"
- "tbconstr.sqC -- How to create, use, and drop constraints (C++)"
- "tbcreate.sqC -- How to create and drop tables (C++)"
- "tbtrig.sqC -- How to use a trigger on a table (C++)"

- “DtUdt.java -- How to create, use and drop user defined distinct types (JDBC)”
- “TbConstr.java -- How to create, use and drop constraints (JDBC)”
- “TbCreate.java -- How to create and drop tables (JDBC)”
- “TbGenCol.java -- How to use generated columns (JDBC)”
- “TbIdent.java -- How to use Identity Columns (JDBC)”
- “TbTrig.java -- How to use triggers (JDBC)”
- “DtUdt.sqlj -- How to create, use and drop user defined distinct types (SQLj)”
- “TbConstr.sqlj -- How to create, use and drop constraints (SQLj)”
- “TbCreate.sqlj -- How to create and drop tables (SQLj)”
- “TbIdent.sqlj -- How to use Identity Columns (SQLj)”
- “TbTrig.sqlj -- How to use triggers (SQLj)”
- “impexp.sqb -- Export and import tables with table data (MF COBOL)”

CREATE TABLESPACE

La sentencia CREATE TABLESPACE crea un nuevo espacio de tablas en la base de datos, asigna contenedores al espacio de tablas y registra la definición y los atributos del espacio de tablas en el catálogo.

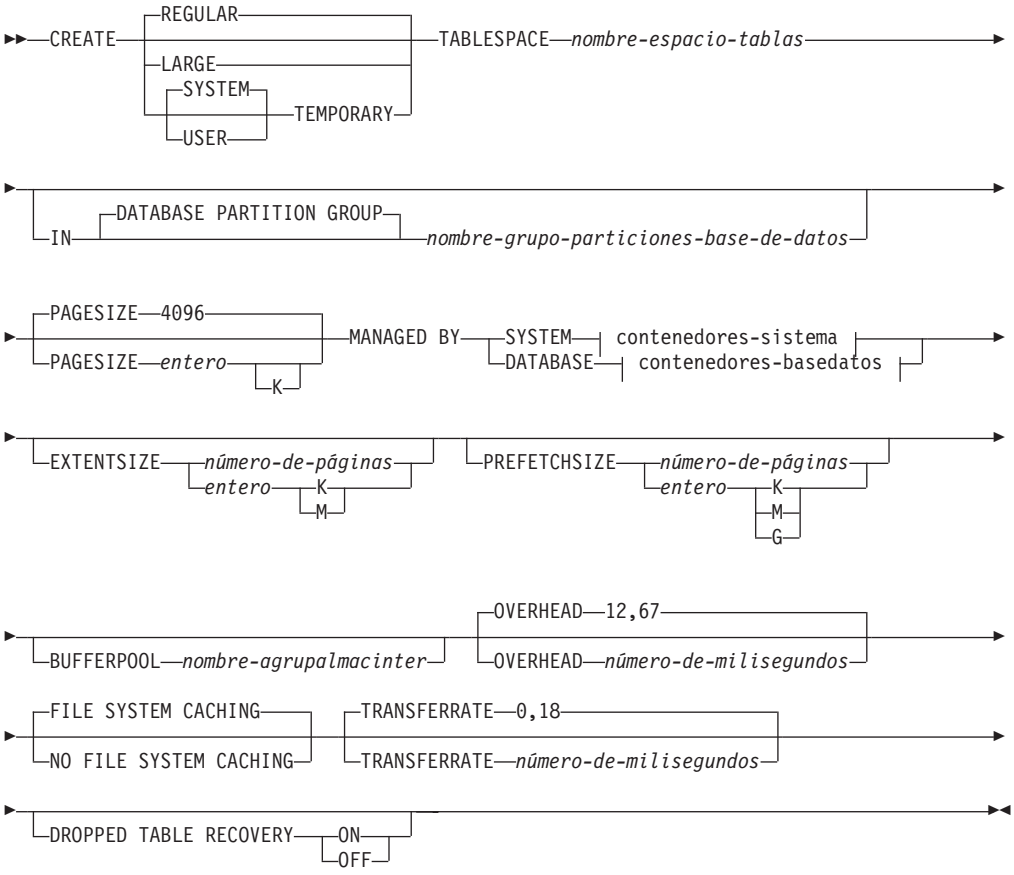
Invocación:

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

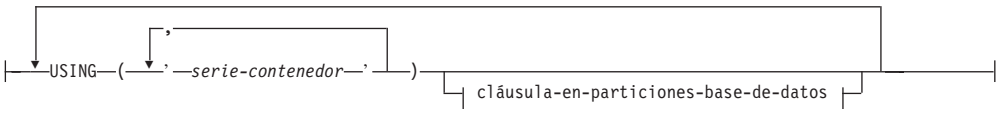
Autorización:

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

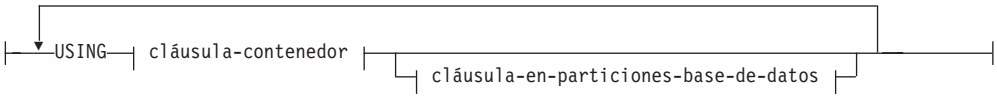
Sintaxis:



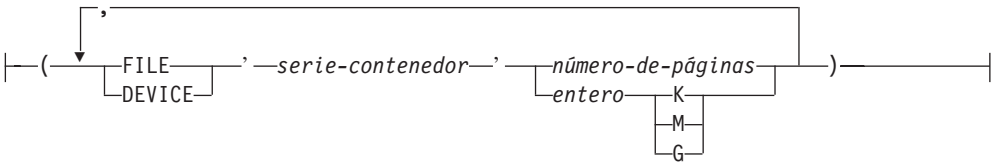
contenedores-sistema:



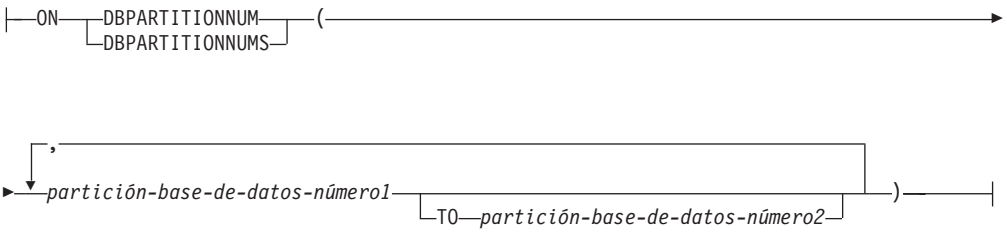
contenedores-basedatos:



cláusula-contenedor:



cláusula-en-particiones-base-de-datos:



Descripción:

REGULAR
Almacena todos los datos salvo las tablas temporales.

LARGE
Almacena columnas de tabla largas o de tipo LOB. También puede almacenar columnas de tipo estructurado o datos de índice. El espacio de tablas debe ser DMS.

SYSTEM TEMPORARY
Almacena tablas temporales (son áreas de trabajo que el gestor de bases de datos utiliza para realizar operaciones tales como clasificaciones o uniones). La palabra clave SYSTEM es opcional. Tenga en cuenta que una base de datos debe tener siempre como mínimo un espacio de tablas SYSTEM TEMPORARY, ya que las tablas temporales sólo pueden almacenarse en un espacio de tablas de esta clase. Al crear una base de datos, se crea automáticamente un espacio de tablas temporal.

USER TEMPORARY
Almacena tablas temporales globales declaradas. Tenga en cuenta que no existen espacios de tablas temporales de usuario cuando se crea una base de

datos. Se debe crear como mínimo un espacio de tablas temporales de usuario con los privilegios USE adecuados, para permitir la definición de tablas temporales declaradas.

nombre-espacio-tablas
Nombra el espacio de tablas. Es un nombre que consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-espacio-tablas* no debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42710). El *nombre-espacio-tablas* no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

IN DATABASE PARTITION GROUP *nombre-grupo-particiones-base-de-datos*
Especifica el grupo de particiones de base de datos para el espacio de tablas. El grupo de particiones de base de datos debe existir. El único grupo de particiones de base de datos que se puede especificar al crear un espacio de tablas SYSTEM TEMPORARY es IBMTEMPGROUP. Las palabras clave DATABASE PARTITION GROUP son opcionales.

Si no se especifica el grupo de particiones de base de datos, se utiliza el grupo de particiones de base de datos por omisión (IBMDEFAULTGROUP) para los espacios de tablas REGULAR, LARGE y USER TEMPORARY. Para los espacios de tablas SYSTEM TEMPORARY, se utiliza el grupo de particiones de base de datos por omisión IBMTEMPGROUP.

PAGESIZE *entero* **[K]**
Define el tamaño de las páginas utilizadas para el espacio de tablas. Los valores válidos de *entero* sin el sufijo K son 4 096, 8 192, 16 384 ó 32 768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE) o el tamaño de página no es el mismo que el de la agrupación de almacenamientos intermedios asociada al espacio de tablas (SQLSTATE 428CB). El valor por omisión es páginas de 4 096 bytes (4K). Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio.

MANAGED BY SYSTEM
Especifica que el espacio de tablas debe estar gestionado por el sistema (SMS).

contenedores-sistema
Especifique los contenedores para un espacio de tablas SMS.

USING *('serie-contenedor',...)*
Para un espacio de tablas SMS, identifica uno o varios contenedores que pertenecerán al espacio de tablas y en el que se almacenarán los datos del espacio de tablas. La *serie-contenedor* no puede sobrepasar los 240 bytes de longitud.

Cada *serie-contenedor* puede ser un nombre de directorio absoluto o relativo. El nombre de directorio, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre de directorio no existe, el gestor de bases de datos lo crea. Cuando se descarta un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos. Si existe el directorio identificado por serie-contenedor, éste no debe contener archivos ni subdirectorios (SQLSTATE 428B2).

El formato de *serie-contenedor* es dependiente del sistema operativo. El sistema operativo especifica los contenedores de la manera habitual. Por ejemplo, una vía de acceso de directorio de Windows empieza por una letra de unidad y un signo ":", mientras que en los sistemas basados en UNIX, una vía de acceso empieza por un signo "/".

Sólo se da soporte simultáneo a recursos remotos (por ejemplo, unidades redirigidas a la LAN o sistemas de archivos montados en NFS), cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o S4100 o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que sólo se da soporte a NEC Storage NS Series con la utilización de una fuente de alimentación ininterrumpible (UPS); se recomienda una UPS continua (en lugar de en espera).

cláusula-en-particiones-base-de-datos
Especifica la partición o particiones en las que se crean los contenedores en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones del grupo de particiones de base de datos que no se han especificado explícitamente en ninguna otra *cláusula-en-particiones-base-de-datos*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, cuando no se especifica la *cláusula-en-particiones-base-de-datos*, los contenedores también se crearán en todas las particiones nuevas que se han añadido a la base de datos.

MANAGED BY DATABASE
Especifica que el espacio de tablas debe estar gestionado por la base de datos (DMS).

contenedores-basedatos
Especifique los contenedores para un espacio de tablas DMS.

USING
Introduce una cláusula-contenedor.

cláusula-contenedor
Especifica los contenedores para un espacio de tablas DMS.

(FILE | DEVICE *'serie-contenedor' número-de-páginas,...***)**
Para un espacio de tablas DMS, identifica uno o varios contenedores que pertenecerán al espacio de tablas y donde se almacenarán los datos del espacio de tablas. Se especifican el tipo del contenedor (FILE o DEVICE) y su tamaño (en páginas de PAGESIZE). El tamaño también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el número de páginas para el contenedor. Se puede especificar una mezcla de contenedores FILE y DEVICE. La *serie-contenedor* no puede sobrepasar los 254 bytes de longitud.

En un contenedor FILE, la *serie-contenedor* debe ser un nombre de archivo relativo o absoluto. El nombre de archivo, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre de directorio no existe, el gestor de bases de datos lo crea. Si el archivo no existe, se creará e inicializará con el tamaño especificado por el gestor de bases de datos. Cuando se descarta un espacio de tablas, se suprimen todos los componentes creados por el gestor de bases de datos.

Nota: Si el archivo existe, se sobregrabará y si es menor a lo especificado, se ampliará. El archivo no se truncará si es más grande de lo que se ha especificado.

Para un contenedor DEVICE, la *serie-contenedor* debe ser un nombre de dispositivo. El dispositivo ya debe existir.

Todos los contenedores deben ser exclusivos entre todas las bases de datos; un contenedor sólo puede pertenecer a un espacio de tablas. El tamaño de los contenedores puede diferir, aunque el rendimiento ideal se consigue cuando todos los contenedores tienen el mismo tamaño. El formato exacto *serie-contenedor* depende del sistema operativo. El sistema operativo especificará los contenedores de la manera habitual.

Sólo se da soporte simultáneo a recursos remotos (por ejemplo, unidades redirigidas a la LAN o sistemas de archivos montados en NFS), cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o S4100 o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que sólo se da soporte a NEC Storage NS Series con la utilización de una fuente de alimentación ininterrumpible (UPS); se recomienda una UPS continua (en lugar de en espera).

cláusula-en-particiones-base-de-datos
Especifica la partición o particiones en las que se crean los contenedores en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones del grupo de particiones de base de datos que no se han especificado explícitamente en ninguna otra *cláusula-en-particiones-base-de-datos*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, cuando no se especifica la *cláusula-en-particiones-base-de-datos*, los contenedores también se crearán en todas las particiones nuevas que se han añadido a la base de datos.

cláusula-en-particiones-base-de-datos
Especifica las particiones en las que se crean los contenedores en una base de datos particionada.

ON DBPARTITIONNUMS
Palabras clave que indican que se han especificado particiones específicas. DBPARTITIONNUM es un sinónimo de DBPARTITIONNUMS.

partición-base-de-datos-número1
Especifique un número de partición de base de datos.

TO *partición-base-de-datos-número2*
Especifique un rango de números de partición. El valor de *partición-base-de-datos-número2* debe ser mayor que o igual al valor de *partición-base-de-datos-número1* (SQLSTATE 428A9). Todas las particiones que se encuentran entre los números de partición especificados, éstos inclusive, se incluirán en las particiones para las que se crean los contenedores si la partición está incluida en el grupo de particiones de base de datos del espacio de tablas.

La partición especificada por el número y todas las particiones que se encuentren en el rango de particiones deben existir en el grupo de particiones de base de datos en el que se define el espacio de tablas (SQLSTATE 42729). Un número de partición sólo puede aparecer de forma explícita o dentro de un rango de, exactamente, una *cláusula-en-particiones-base-de-datos* para la sentencia (SQLSTATE 42613).

EXTENTSIZE *número-de-páginas*
Especifica el número de páginas de PAGESIZE que se grabarán en un

contenedor antes de pasar al siguiente contenedor. El valor del tamaño de extensión también puede especificarse como un valor entero seguido de K (para kilobytes) o de M (para megabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor del tamaño de extensión. El gestor de bases de datos pasa periódicamente por los contenedores a medida que se almacenan datos.

El valor por omisión lo proporciona el parámetro de configuración de base de datos DFT_EXTENT_SZ, que tiene un rango válido de 2 a 256 páginas.

PREFETCHSIZE *número-de-páginas*
Especifica el número de páginas PAGESIZE del espacio de tablas que se leerán cuando se realice la captación previa de datos. El valor del tamaño de captación previa también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor del número de páginas para el tamaño de captación previa. La captación previa lee los datos que una consulta necesita antes que la consulta haga referencia a ellos, por lo que la consulta no necesita esperar a que se efectúen operaciones de E/S.

El valor por omisión lo proporciona el parámetro de configuración DFT_PREFETCH_SZ.

BUFFERPOOL *nombre-agrupalmacinter*
El nombre de la agrupación de almacenamientos intermedios utilizado para las tablas de este espacio de tablas. La agrupación de almacenamientos intermedios debe existir (SQLSTATE 42704). Si no se especifica, se utiliza la agrupación de almacenamientos intermedios por omisión (IBMDEFAULTBP). El tamaño de página de la agrupación de almacenamientos intermedios debe coincidir con el tamaño de página especificado (o el valor por omisión) para el espacio de tablas (SQLSTATE 428CB). Debe haberse definido el grupo de particiones de base de datos del espacio de tablas para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

OVERHEAD *número-de-milisegundos*
Cualquier literal numérico (entero, decimal o de coma flotante) que especifique la actividad general del controlador de E/S y el tiempo de búsqueda y latencia del disco, en milisegundos. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

FILE SYSTEM CACHING o **NO FILE SYSTEM CACHING**
Especifica si las operaciones de E/S se deben almacenar en antememoria o no a nivel del sistema de archivos. El valor por omisión es FILE SYSTEM CACHING.

FILE SYSTEM CACHING
Especifica que todas las operaciones de E/S del espacio de tablas de destino se deben almacenar en antememoria a nivel del sistema de archivos.

NO FILE SYSTEM CACHING
Especifica que todas las operaciones de E/S no se deben almacenar en antememoria a nivel del sistema de archivos.

TRANSFERRATE *número-de-milisegundos*
Cualquier literal numérico (entero, decimal o de coma flotante) que especifique el tiempo para leer una página en la memoria, en milisegundos. El número debe ser el promedio de todos los contenedores que pertenecen al espacio de tablas, si no es el mismo para todos los contenedores. Este valor sirve para determinar el coste de E/S durante la optimización de una consulta.

DROPPED TABLE RECOVERY
Se pueden recuperar las tablas descartadas del espacio de tablas especificado utilizando la opción RECOVER TABLE ON del mandato ROLLFORWARD. Esta cláusula sólo se puede especificar para un espacio de tablas REGULAR (SQLSTATE 42613).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - NODES puede especificarse en lugar de DBPARTITIONNUMS
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
    - LONG puede especificarse en lugar de LARGE
- La elección entre un espacio de tablas gestionado por la base de datos y un espacio de tablas gestionado por el sistema es una elección fundamental que implica ventajas e inconvenientes.
- Cada definición de contenedor requiere 53 bytes más el número de bytes necesario para almacenar el nombre de contenedor. La longitud combinada de todos los nombres de contenedores para el espacio de tablas no puede exceder de 20 480 bytes (SQLSTATE 54034).
- Cuando existe más de un espacio de tablas TEMPORARY en la base de datos, se utilizarán en modalidad rotatoria para equilibrar su utilización.
- En una base de datos particionada, si más de una partición de base de datos reside en el mismo nodo físico, no podrá especificarse el mismo dispositivo o vía de acceso específica para tales particiones des base de datos (SQLSTATE 42730). Para este entorno, especifique una *serie-contenedor* específica para cada partición o bien utilice un nombre de vía de acceso relativa.
- Puede especificar una expresión de partición de base de datos para la sintaxis de la serie de caracteres del contenedor al crear contenedores SMS o DMS. Normalmente, especificará la expresión de partición de base de datos si ha estado utilizando varias particiones de base de datos lógicas en el sistema de bases de datos particionadas. Esto asegura que los nombres de los contenedores sean exclusivos para los nodos (servidores de particiones de bases de datos). Cuando especifica la expresión, el número de partición de base de datos forma parte del nombre del contenedor o, si especifica argumentos adicionales, el resultado del argumento forma parte del nombre del contenedor.

  El argumento " $N" ([blanco]$N) se utiliza para indicar una expresión de partición de base de datos. Una expresión de partición de base de datos puede utilizarse en cualquier punto del nombre del contenedor, y pueden especificarse varias expresiones de partición de base de datos. La expresión de partición de base de datos debe finalizar con un carácter de espacio; lo que siga al espacio se añadirá al nombre del contenedor tras evaluarse la expresión de partición de base de datos. Si no existe ningún carácter de espacio en el nombre del contenedor después de la expresión de partición de base de datos, se da por

supuesto que el resto de la serie de caracteres forma parte de la expresión. El argumento sólo puede utilizarse de una de las formas siguientes:

Tabla 8. Argumentos para la creación de contenedores. Los operadores se evalúan de izquierda a derecha. En los ejemplos, se da por supuesto que el número de partición de base de datos es el 5.

Sintaxis	Ejemplo	Valor
[blanco]$N	" $N"	5
[blanco]$N+[número]	" $N+1011"	1016
[blanco]$N%[número]	" $N%3" [a]	2
[blanco]$N+[número]%[número]	" $N+12%13"	4
[blanco]$N%[número]+[número]	" $N%3+20"	22
[a] % es el módulo.		

Por ejemplo:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
  (device '/dev/rcont $N' 20000)

En un sistema de dos particiones de base de datos, se crearían
 los contenedores siguientes:
  /dev/rcont0  - on DATABASE PARTITION 0
  /dev/rcont1  - on DATABASE PARTITION 1


CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
  (file '/DB2/containers/TS2/container $N+100' 10000)

En un sistema de cuatro particiones de base de datos, se crearían
 los contenedores siguientes:
  /DB2/containers/TS2/container100  - on DATABASE PARTITION 0
  /DB2/containers/TS2/container101  - on DATABASE PARTITION 1
  /DB2/containers/TS2/container102  - on DATABASE PARTITION 2
  /DB2/containers/TS2/container103  - on DATABASE PARTITION 3


CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
      ('/TS3/cont $N%2','/TS3/cont $N%2+2')

En un sistema de dos particiones de base de datos, se crearían
 los contenedores siguientes:
  /TS3/cont0  - On DATABASE PARTITION 0
  /TS3/cont2  - On DATABASE PARTITION 0
  /TS3/cont1  - On DATABASE PARTITION 1
  /TS3/cont3  - On DATABASE PARTITION 1


Si la partición de base de datos = 5, los contenedores:
  '/dbdir/node $N /cont1'
  '/ $N+1000 /file1'
  ' $N%10 /container'
  '/dir/ $N%5+2000 /dmscont'

se crean como:
  '/dbdir/node5/cont1'
  '/1005/file1'
  '5/container'
  '/dir/2000/dmscont'
```

Ejemplos:

Ejemplo 1: Cree un espacio de tablas DMS normal en un sistema basado en UNIX que utiliza 3 dispositivos de 10 000 páginas de 4K cada una. Especifique sus características de E/S.

```
CREATE TABLESPACE PAYROLL
  MANAGED BY DATABASE
  USING (DEVICE'/dev/rhdisk6' 10000,
    DEVICE '/dev/rhdisk7' 10000,
    DEVICE '/dev/rhdisk8' 10000)
  OVERHEAD 12.67
  TRANSFERRATE 0.18
```

Ejemplo 2: Cree un espacio de tablas SMS normal en Windows NT o Windows 2000 que utilice 3 directorios en tres unidades distintas, con un tamaño de extensión de 64 páginas y un tamaño de captación previa de 32 páginas.

```
CREATE TABLESPACE ACCOUNTING
  MANAGED BY SYSTEM
  USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
  EXTENTSIZE 64
  PREFETCHSIZE 32
```

Ejemplo 3: Cree un espacio de tablas DMS temporal en Unix que utiliza 2 archivos de 50.000 páginas cada uno y un tamaño de extensión de 256 páginas.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY DATABASE
  USING (FILE '/tmp/tempspace2.f1' 50000,
    FILE '/tmp/tempspace2.f2' 50000)
  EXTENTSIZE 256
```

Ejemplo 4: Cree un espacio de tablas DMS en el grupo de particiones de base de datos ODDNODEGROUP (particiones 1,3,5) en una base de datos particionada de Unix. En todas las particiones, utilice el dispositivo /dev/rhdisk0 para 10 000 páginas de 4K. Especifique también un dispositivo específico de la partición para cada partición con 40 000 páginas de 4K.

```
CREATE TABLESPACE PLANS
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
  ON DBPARTITIONNUM (1)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
  ON DBPARTITIONNUM (3)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
  ON DBPARTITIONNUM (5)
```

Ejemplos relacionados:

- "tbtemp.sqc -- How to use a declared temporary table (C)"
- "TbTemp.java -- How to use Declared Temporary Table (JDBC)"

CREATE TRANSFORM

La sentencia CREATE TRANSFORM define funciones o métodos de transformación, identificados por un nombre de grupo, que se utilizan para intercambiar valores de tipo estructurado con programas en lenguaje del sistema principal y con funciones y métodos externos.

Invocación:

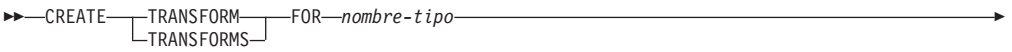
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

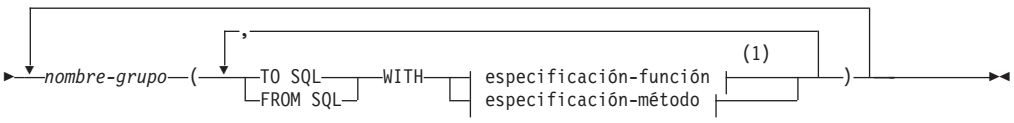
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

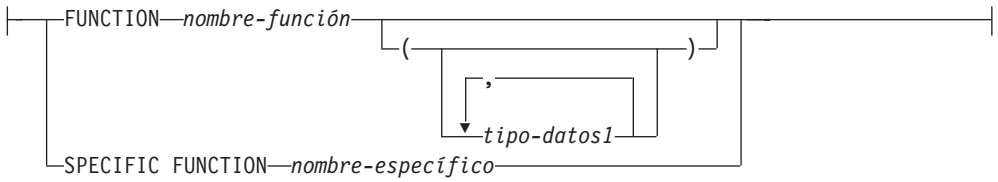
- Autorización SYSADM o DBADM
- Privilegio del usuario que define el tipo que *nombre-tipo* identifica y privilegio EXECUTE para cada función especificada.

Sintaxis:

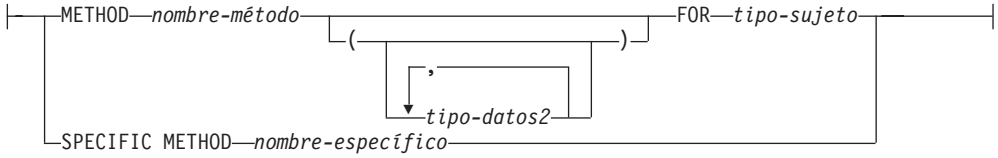




especificación-función:



especificación-método:



Notas:

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

Descripción:

TRANSFORM o **TRANSFORMS**
Indica que se están definiendo uno o más grupos de transformación. Se puede especificar cualquiera de las dos versiones de la palabra clave.

FOR *nombre-tipo*
Especifica un nombre para el tipo estructurado definido por el usuario para el cual se define el grupo de transformación.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un *nombre-tipo* no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para un *nombre-tipo* no calificado. El *nombre-tipo* debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado (SQLSTATE 42809). El tipo estructurado o cualquier otro tipo estructurado incluido en la misma jerarquía de tipos no debe tener transformaciones ya definidas con el nombre-grupo proporcionado (SQLSTATE 42739).

nombre-grupo
Especifica el nombre del grupo de transformación que contiene las funciones o métodos TO SQL y FROM SQL. No es necesario que el nombre sea exclusivo, pero si un grupo de transformación tiene este nombre (con la misma instrucción TO SQL y/o FROM SQL definida), no debe estar definido previamente para el *nombre-tipo* especificado (SQLSTATE 42739). Un *nombre-grupo* debe ser un identificador SQL, con una longitud máxima de 18 caracteres (SQLSTATE 42622) y no puede contener ningún prefijo calificador (SQLSTATE 42601). El *nombre-grupo* no puede comenzar con el prefijo 'SYS', pues éste está reservado para su uso por la base de datos (SQLSTATE 42939).

Como máximo, se puede especificar una de las clases siguientes de funciones FROM SQL y TO SQL para un grupo dado cualquiera (SQLSTATE 42628).

TO SQL
Define la función específica que se utiliza para transformar un valor al formato del tipo estructurado SQL definido por el usuario. La función debe tener todos sus parámetros como tipos de datos incorporados y el tipo devuelto es *nombre-tipo*.

FROM SQL
Define la función específica que se utiliza para transformar un valor a un tipo de datos incorporado representativo del tipo estructurado SQL definido por el usuario. La función debe tener un parámetro del tipo de datos *nombre-tipo* y devolver un tipo de datos incorporado (o conjunto de tipos de datos incorporados).

WITH *especificación-función*
Hay varias maneras de especificar la instancia de función.

Si se especifica FROM SQL, *especificación-función* debe identificar una función que cumpla los requisitos siguientes:

- Existe un parámetro del tipo *nombre-tipo*.
- El tipo de retorno es un tipo incorporado o una fila donde todas sus columnas tienen tipos incorporados.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación FROM SQL que tiene LANGUAGE SQL.

Si se especifica TO SQL, *especificación-función* debe identificar una función que cumpla los requisitos siguientes:

- Todos los parámetros tienen tipos incorporados.
- El tipo de retorno es *nombre-tipo*.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación TO SQL que tiene LANGUAGE SQL.

Si *especificación-función* identifica una función que no cumple estos requisitos (de acuerdo con su utilización como función de transformación FROM SQL o TO SQL), se genera un error (SQLSTATE 428DC).

FUNCTION *nombre-función*
Identifica la función específica por su nombre, y sólo es válido si hay exactamente una función con el *nombre-función*. Para la función identificada puede definirse cualquier número de parámetros.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si hay más de una instancia específica de la función en el esquema nombrado o implícito, se produce un error (SQLSTATE 42725). El algoritmo estándar de selección de funciones no se utiliza.

FUNCTION *nombre-función* **(***tipo-datos1,...***)**
Proporciona la signatura de la función, que identifica de manera exclusiva la función que se debe utilizar. El algoritmo estándar de selección de funciones no se utiliza.

nombre-función
Especifica el nombre de la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

(*tipo-datos1,...***)**
Los tipos de datos que se especifican aquí deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION en las correspondientes posiciones. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la función específica.

Si el tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En su lugar, puede codificarse un conjunto vacío de paréntesis para indicar que esos atributos deben ignorarse al comparar tipos de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE). De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor que se ha definido para n, pues 0<n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE. Observe que el atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la comparación de patrones. Por ejemplo, un CHAR FOR BIT DATA especificado en la signatura coincidiría con una función definida con CHAR solamente.

Si en el esquema nombrado o implícito no existe ninguna función con la signatura especificada, se produce un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-específico*
Identifica la función definida por el usuario, mediante un nombre que se especifica o se toma por omisión al crear la función.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica del esquema nombrado o implícito; de lo contrario se produce un error (SQLSTATE 42704).

WITH *especificación-método*
Existen varias formas de especificar una instancia de método.

Si se especifica FROM SQL, *especificación-método* debe identificar un método que cumpla los requisitos siguientes:

- El tipo que *tipo-sujeto* especifica es del tipo *nombre-tipo*.
- El tipo de retorno es un tipo incorporado o una fila cuyas columnas utilizan tipos incorporados.
- La signatura especifica LANGUAGE SQL o la utilización de otra función o método de transformación FROM SQL que tiene LANGUAGE SQL.

Si se especifica TO SQL, *especificación-método* debe identificar un método que cumpla los requisitos siguientes:

- El tipo que *tipo-sujeto* especifica es del tipo *nombre-tipo*.
- Todos los demás tipos de parámetros, excepto el tipo del propio parámetro implícito, son tipos incorporados.
- El tipo de retorno es *nombre-tipo*.
- El método se ha declarado como SELF AS RESULT.
- La signatura especifica LANGUAGE SQL o la utilización de otra función o método de transformación TO SQL que tiene LANGUAGE SQL.

Si *especificación-método* identifica un método que no cumple estos requisitos (de acuerdo con su utilización como método de transformación FROM SQL o TO SQL), se genera un error (SQLSTATE 428DC).

METHOD *nombre-método* **FOR** *tipo-sujeto*
Identifica el método en particular por el nombre o por el tipo y sólo es válido si existe exactamente un método con las propiedades especificadas. El identificador *nombre-método* es un nombre no calificado. Para el método identificado puede definirse cualquier número de parámetros.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de

precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

Si no existe ningún método con este nombre para el tipo especificado en el esquema implícito, se genera un error (SQLSTATE 42704). Si existe más de una instancia específica del método en el esquema implícito, se genera un error (SQLSTATE 42725). No se utiliza el algoritmo de resolución de método estándar.

METHOD *nombre-método* **(***tipo-datos2,...***) FOR** *tipo-sujeto*
Proporciona la signatura del método, que identifica de forma exclusiva el método que debe utilizarse. No se utiliza el algoritmo de resolución de método estándar.

nombre-método **FOR** *tipo-sujeto*
Especifica el nombre del método. El identificador *nombre-método* es un nombre no calificado. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

(*tipo-datos2,...***)**
Los tipos de datos que se especifican aquí deben coincidir con los tipos de datos que se han especificado en la sentencia de creación de método en las correspondientes posiciones. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar el método especificado.

Si el tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

No es necesario especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. En su lugar, puede codificarse un conjunto vacío de paréntesis para indicar que esos atributos deben ignorarse al comparar tipos de datos.

FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE). Sin embargo, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se ha especificado en la sentencia de creación de método.

No es necesario que un tipo de FLOAT(n) coincida con el valor que se ha definido para n, pues 0<n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE. Observe que el atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la comparación de patrones. Por ejemplo, la especificación de CHAR FOR BIT DATA en la signatura sólo tendría que coincidir con un método definido con CHAR.

Si no existe ningún método con la signatura especificada en el esquema implícito, se genera un error (SQLSTATE 42883).

SPECIFIC METHOD *nombre-específico*
Identifica el método definido por el usuario en particular, utilizando un nombre específico que se ha especificado o que se ha tomado por omisión al crearse el método.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de método en el esquema especificado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

Normas:

- El tipo o tipos incorporados que se devuelven desde la función o método FROM SQL deben corresponder directamente al tipo o tipos incorporados que son parámetros de la función o método TO SQL. Esto es una consecuencia lógica de la relación inversa que existe entre estas dos funciones. Si no existe esta relación entre la transformación FROM y la transformación TO, se genera un error (SQLSTATE -3).

Notas:

- Cuando no se especifica un grupo de transformación en un programa de aplicación (utilizando la opción de precompilación o de enlace TRANSFORM GROUP para el SQL estático o utilizando la sentencia SET CURRENT DEFAULT TRANSFORM GROUP para el SQL dinámico), las funciones o métodos de transformación del grupo de transformación 'DB2_PROGRAM' se utilizan (si se han definido) cuando el programa de aplicación recupera o envía variables del lenguaje principal que están basadas en el tipo estructurado definido por el usuario que *nombre-tipo* identifica. Cuando se recupera un valor de tipo de datos *nombre-tipo*, se invoca la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado que devuelve la función o método de transformación. De forma similar, cuando se envía una variable del lenguaje principal que se asignará a un valor del tipo de datos *nombre-tipo*, la transformación TO SQL se invoca para transformar el valor del tipo de datos incorporado en el valor del tipo de datos estructurado. Si la transformación TO SQL es un método, se crea un objeto del tipo dinámico adecuado y el método de transformación TO SQL se invoca con el objeto que acaba de crearse como argumento del sujeto. Si no se especifica un grupo de transformación definido por el usuario o si no se ha definido un grupo 'DB2_PROGRAM' (para el tipo estructurado dado), se genera un error (SQLSTATE 42741).
- La representación del tipo de datos incorporado para una variable del lenguaje principal de tipo estructurado debe asignarse:
  - desde el resultado de la función de transformación FROM SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando normas de asignación para la recuperación) y
  - al parámetro de la función de transformación TO SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando normas de asignación de memoria).

  Si una variable del lenguaje principal no ofrece compatibilidad de asignación con el tipo que necesita la función o método de transformación aplicable, se genera un error (para el enlace de entrada: SQLSTATE 42821; para el enlace de salida: SQLSTATE 42806). Para obtener información acerca de los errores que resultan de las asignaciones de series de caracteres, consulte “Asignaciones de series de caracteres”.
- Las funciones o métodos de transformación que se han identificado en el grupo de transformación por omisión denominado 'DB2_FUNCTION' se utilizan

siempre que una función o método definido por el usuario, no escrito en SQL, se invoca mediante la utilización del tipo de datos *nombre-tipo* como parámetro o tipo de retorno. Esto es aplicable cuando la función o método no especifica la cláusula TRANSFORM GROUP. Cuando se invoca la función o método con un argumento del tipo de datos *nombre-tipo*, se ejecuta la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado que la función o método de transformación devuelve. De forma similar, cuando el tipo de datos de retorno de la función o método es un tipo de datos *nombre-tipo*, se invoca la transformación TO SQL para transformar el valor del tipo de datos incorporado que se ha devuelto desde el programa de la función externa en el valor de tipo estructurado.

Si la transformación TO SQL es un método y la transformación TO SQL se utiliza para transformar el resultado de una invocación de método cuyo método se ha declarado como SELF AS RESULT, se crea un objeto del tipo dinámico del sujeto. Si el método no se ha declarado como SELF AS RESULT, se crea un objeto del tipo declarado del resultado del método más específico que puede enviarse. De lo contrario, se crea un objeto del tipo declarado del resultado de la función o método resuelto. El método de transformación TO SQL se invoca con el objeto que acaba de crearse como argumento del sujeto, junto con los argumentos del tipo base que se han devuelto desde la invocación de la función o método real.

- Si un tipo estructurado contiene un atributo que también es un tipo estructurado, las funciones o métodos de transformación asociados deben expandir (o ensamblar) de forma recursiva todos los tipos estructurados anidados. Esto significa que los resultados o los parámetros de las funciones o métodos de transformación sólo se componen del conjunto de tipos incorporados que representa a todos los atributos base del tipo estructurado del sujeto (incluidos todos sus tipos estructurados anidados). No existe ninguna "aplicación en cascada" de las funciones o métodos de transformación para manejar los tipos estructurados anidados.
- Las funciones o métodos identificados en esta sentencia se resuelven, en función de las normas anteriormente descritas, durante la ejecución de esta sentencia. Cuando estas funciones se utilizan (implícitamente) en sentencias de SQL subsiguientes, no son objeto de otro proceso de resolución. Las funciones o métodos de transformación definidos en esta sentencia se registran exactamente como se han resuelto en esta sentencia. Sin embargo, cuando se invoca un método de transformación, se aplican los algoritmos para REFER (Envío dinámico de métodos).
- Cuando se crean o eliminan atributos o subtipos de un tipo determinado, también deben modificarse las funciones de transformación correspondientes al tipo estructurado definido por el usuario.
- Para un grupo de transformación determinado, las transformaciones FROM SQL y TO SQL pueden especificarse en la misma cláusula de *nombre-grupo*, en cláusulas de *nombre-grupo* distintas o en sentencias CREATE TRANSFORM distintas. La única restricción es que una designación de transformación FROM SQL o TO SQL determinada no puede volver a definirse sin eliminar primero la definición de grupo existente. Esto le permite, por ejemplo, definir primero una transformación FROM SQL para un grupo determinado y definir posteriormente la transformación TO SQL correspondiente para el mismo grupo.

Ejemplos:

Ejemplo 1: Cree dos grupos de transformación que asocien el tipo estructurado definido por el usuario POLYGON a funciones de transformación adaptadas para C y Java respectivamente.

```
CREATE TRANSFORM FOR POLYGON
  mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
             TO SQL WITH FUNCTION myxform_structsql)
  myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
             TO SQL WITH FUNCTION myxform_javasql)
```

Ejemplo 2: Cree dos grupos de transformación que asocien el tipo estructurado definido por el usuario POLYGON a métodos de transformación adaptados para C y Java respectivamente.

```
CREATE TRANSFORM FOR POLYGON
  mystruct1 (FROM SQL WITH METHOD myxform_sqlstruct FOR POLYGON,
             TO SQL WITH METHOD myxform_structsql FOR POLYGON)
  myjava1   (FROM SQL WITH METHOD myxform_sqljava FOR POLYGON,
             TO SQL WITH METHOD myxform_javasql FOR POLYGON)
```

Ejemplo 3: Cree un grupo de transformación para un tipo que no está en el esquema actual.

```
CREATE TRANSFORM FOR NEWTON.POLYGON
  mystruct1 (FROM SQL WITH METHOD myxform_sqlstruct FOR NEWTON.POLYGON,
             TO SQL WITH METHOD myxform_structsql FOR NEWTON.POLYGON)
```

Información relacionada:

- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

CREATE TRIGGER

La sentencia CREATE TRIGGER define un activador en una base de datos.

Invocación:

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

Autorización:

Cuando se crea el activador, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio ALTER para la tabla en la que se define el activador BEFORE o AFTER
- Privilegio CONTROL para la vista en la que se define el activador INSTEAD OF
- Privilegio del definidor de la vista en la que se define el activador INSTEAD OF
- Privilegio ALTERIN para el esquema de la tabla o de la vista en la que se define el activador

y uno de estos:

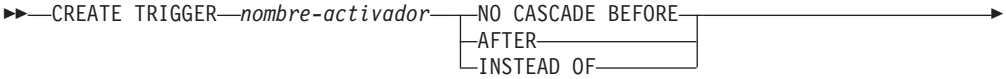
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del activador no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema del activador hace referencia a un esquema existente.

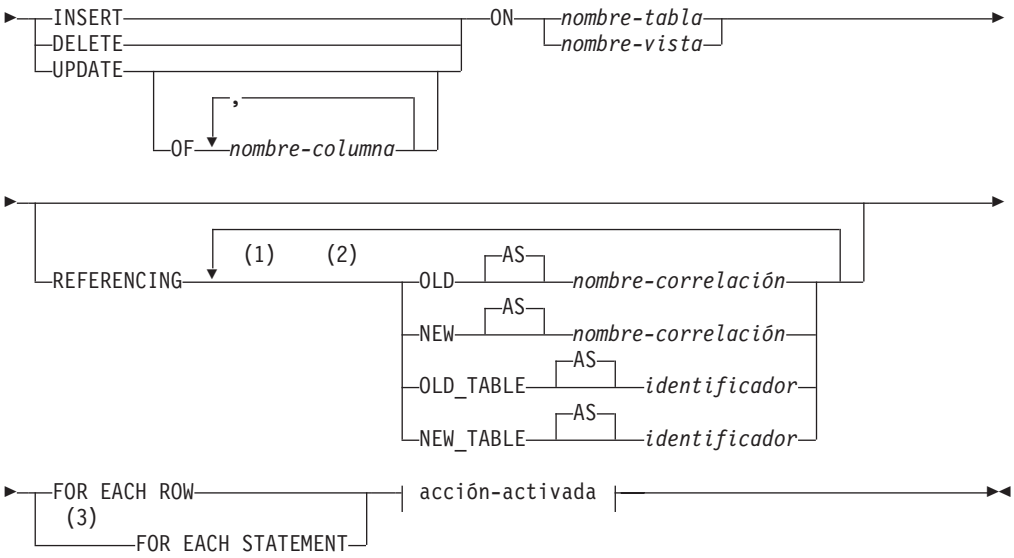
Si el ID de autorización de la sentencia no tiene la autorización SYSADM ni DBADM, los privilegios que tiene el ID de autorización de la sentencia (sin tener en cuenta los privilegios PUBLIC ni de grupo) deben incluir todos los siguientes elementos, siempre que exista el activador:

- Privilegio SELECT para la tabla donde está definido el activador, si se especifica cualquier tabla o variable de transición.
- Privilegio SELECT para cualquier tabla o vista referenciada en la condición de la acción activada.
- Los privilegios necesarios para invocar las sentencias de SQL activadas especificadas.

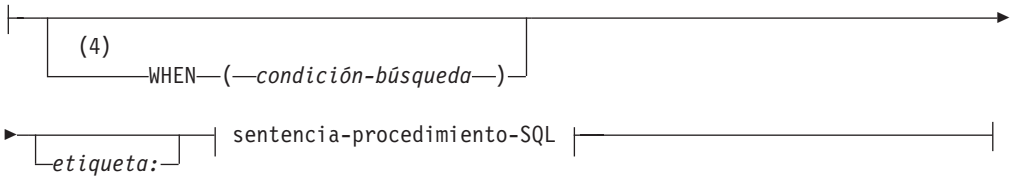
Si la persona que define un activador sólo puede crear el activador porque tiene autorización SYSADM, entonces se le otorga la autorización DBAM explícita para crear el activador.

Sintaxis:

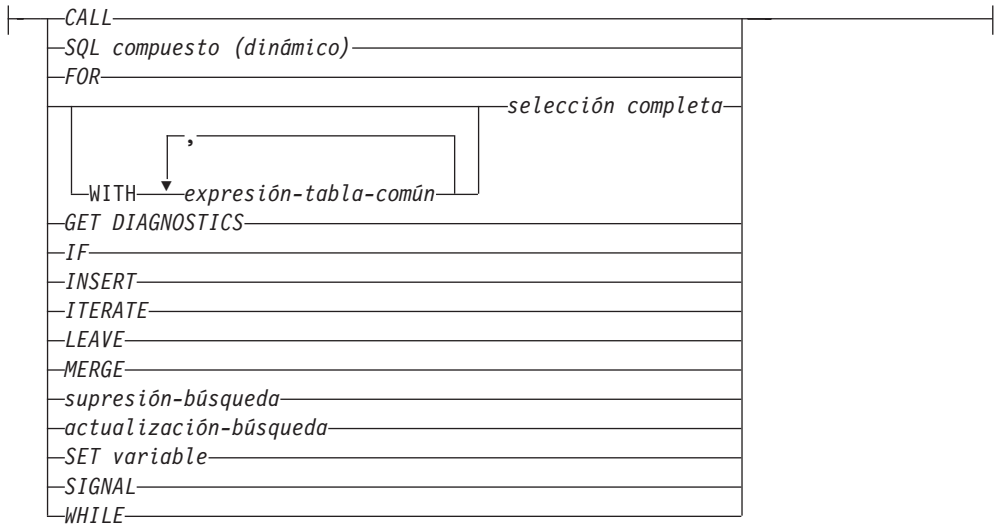




acción-activada:



sentencia-procedimiento-SQL:



Notas:

1 Puede especificarse OLD y NEW sólo una vez cada uno.

2 OLD_TABLE y NEW_TABLE sólo pueden especificarse, cada uno de ellos, una vez, y sólo para los activadores AFTER o para los activadores INSTEAD OF.

3 FOR EACH STATEMENT no puede especificarse para activadores BEFORE o para activadores INSTEAD OF.

4 La condición WHEN no puede especificarse para los activadores INSTEAD OF.

Descripción:

nombre-activador
Indica el nombre del activador. El nombre, incluido en el nombre de esquema implícito o explícito, no debe identificar un activador que ya esté descrito en el catálogo (SQLSTATE 42710). Si se especifica un nombre compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS' (SQLSTATE 42939).

NO CASCADE BEFORE
Especifica que la acción activada asociada se debe aplicar antes de aplicar a la base de datos los cambios ocasionados por la actualización real de la tabla sujeto. También especifica que la acción activada del activador no provocará la activación de otros activadores.

AFTER
Especifica que la acción activada asociada se debe aplicar después de que los cambios ocasionados por la actualización real y la tabla sujeto se apliquen a la base de datos.

INSTEAD OF
Especifica que la acción activada asociada sustituye a la acción que corresponde a la vista sujeto. Sólo está permitido un activador INSTEAD OF para cada tipo de operación de una vista sujeto determinada (SQLSTATE 428FP).

INSERT
Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique una operación INSERT a la tabla sujeto o a la vista sujeto.

DELETE
Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique una operación DELETE a la tabla sujeto o a la vista sujeto.

UPDATE
Especifica que la acción activada que se asocia al activador va a ejecutarse siempre que se aplique la operación UPDATE a la tabla sujeto o a la vista sujeto, de acuerdo con las columnas especificadas o implícitas.

Si no se especifica la lista opcional *nombre-columna*, estarán implicadas todas las columnas de la tabla. Por lo tanto, la omisión de la lista *nombre-columna* supone la activación del activador cuando se actualiza cualquier columna de la tabla.

OF *nombre-columna*,...
Cada *nombre-columna* especificada debe ser una columna de la tabla base (SQLSTATE 42703). Si el activador es un activador anterior, el *nombre-columna* especificado no puede ser una columna generada distinta de la columna de identidad (SQLSTATE 42989). Ningún *nombre-columna* deberá aparecer más de una vez en la lista *nombre-columna* (SQLSTATE 42711). El activador sólo se podrá activar al actualizar una columna que conste en la lista *nombre-columna*. Esta cláusula no puede especificarse para un activador INSTEAD OF (SQLSTATE 42613).

ON

nombre-tabla
Designa la tabla sujeto de la definición del activador BEFORE o del activador AFTER. El nombre debe especificar una tabla base o un alias que se resuelva dando como resultado una tabla base (SQLSTATE 42704 ó 42809). El nombre no debe especificar una tabla de catálogo (SQLSTATE 42832), una tabla de consultas materializadas (SQLSTATE 42997), una tabla temporal declarada (SQLSTATE 42995) o un apodo (SQLSTATE 42809).

nombre-vista
Designa la vista sujeto de la definición del activador INSTEAD OF. El nombre debe especificar una vista sin tipo o un alias que se resuelva dando como resultado una vista sin tipo (SQLSTATE 42704 ó 42809). El nombre no debe especificar una vista de catálogo (SQLSTATE 42832). El nombre no debe especificar una vista que se haya definido utilizando WITH CHECK OPTION (una vista simétrica) o una vista en la que se haya definido una vista simétrica, directa o indirectamente (SQLSTATE 428FQ).

REFERENCING
Especifica los nombres de correlación para las *variables de transición* y los nombres de tabla para las *tablas de transición*. Los nombres de correlación identifican una fila determinada del conjunto de filas que se ven afectadas por la operación SQL activador. Los nombres de tabla identifican todo el conjunto de filas afectadas. Cada fila afectada por la operación SQL activador está disponible para la acción activada mediante la calificación de las columnas con *nombres-correlación* especificados de la siguiente manera:

OLD AS *nombre-correlación*
Especifica un nombre de correlación que identifica el estado de fila anterior a la operación SQL activador.

NEW AS *nombre-correlación*
Especifica un nombre de correlación que identifica el estado de la fila tal como la ha modificado la operación SQL activador y cualquier sentencia SET de un activador BEFORE que ya se ha ejecutado.

La acción activada dispone del conjunto completo de filas afectadas por la operación SQL activador mediante la utilización de un nombre de tabla temporal que se especifica de la siguiente manera:

OLD_TABLE AS *identificador*
Especifica un nombre de tabla temporal que identifica el conjunto de filas afectadas antes de la operación SQL activador.

NEW_TABLE AS *identificador*
Especifica un nombre de tabla temporal que identifica las filas afectadas tal como las ha modificado la operación SQL activador y cualquier sentencia SET de un activador BEFORE que ya se ha ejecutado.

Las siguientes normas se aplican a la cláusula REFERENCING:

- Ninguno de los nombres de correlación OLD y NEW y los nombres de OLD_TABLE y NEW_TABLE pueden ser idénticos (SQLSTATE 42712).
- Para un activador solamente se puede especificar un *nombre-correlación* OLD y uno NEW (SQLSTATE 42613).
- Para un activador, sólo puede especificarse un *identificador* OLD_TABLE y NEW_TABLE (SQLSTATE 42613).
- El *nombre-correlación* OLD y el *identificador* OLD_TABLE sólo pueden utilizarse si el suceso del activador es una operación DELETE o una operación UPDATE (SQLSTATE 42898). Si la operación es DELETE, el

nombre-correlación OLD captura el valor de la fila suprimida. Si la operación es UPDATE, captura el valor de la fila antes de la operación UPDATE. Lo mismo se aplica al *identificador* OLD_TABLE y al conjunto de filas afectadas.

- El *nombre-correlación* NEW y el *identificador* NEW_TABLE sólo pueden utilizarse si el suceso del activador es una operación INSERT o una operación UPDATE (SQLSTATE 42898). En ambas operaciones, el valor de NEW captura el nuevo estado de la fila como lo proporciona la operación original y modificado por cualquier activador BEFORE que se haya ejecutado hasta ese momento. Ocurre exactamente lo mismo con el *identificador* NEW_TABLE y el conjunto de filas afectadas.
- Los *identificador*es OLD_TABLE y NEW_TABLE no se pueden definir para un activador BEFORE (SQLSTATE 42898).
- Los *nombres-correlación* no se pueden definir para un activador FOR EACH STATEMENT (SQLSTATE 42899).
- Las tablas de transición no pueden modificarse (SQLSTATE 42807).
- El total de las referencias a las columnas de la tabla de transición y a las variables de transición de la acción activada no puede sobrepasar el límite del número de columnas de una tabla o la suma de sus longitudes no puede exceder la longitud máxima de una fila de una tabla (SQLSTATE 54040).
- El ámbito de cada *nombre-correlación* y de cada *identificador* es la totalidad de la definición del activador.

FOR EACH ROW
Especifica que la acción activada va a aplicarse una vez para cada fila de la tabla sujeto o de la vista sujeto que se vea afectada por la operación de SQL activador.

FOR EACH STATEMENT
Especifica que la acción activada se debe aplicar una vez para toda la sentencia. Este tipo de granularidad de activador no puede especificarse para un activador BEFORE o para un activador INSTEAD OF (SQLSTATE 42613). Si se especifica, se activará un activador UPDATE o DELETE, aunque no exista ninguna fila que se vea afectada por la sentencia UPDATE o DELETE activador.

acción-activada
Especifica la acción que se debe realizar cuando se activa un activador. Una acción activada se compone de una *sentencia-procedimiento-SQL* y de una condición opcional para la ejecución de la *sentencia-procedimiento-SQL*.

WHEN (*condición-búsqueda***)**
Especifica una condición verdadera, falsa o desconocida. La *condición-búsqueda* proporciona la posibilidad de determinar si se debe ejecutar o no una determinada acción activada.

La acción asociada se realiza sólo si la condición de búsqueda especificada es verdadera. Si se omite la cláusula WHEN, la *sentencia-procedimiento-SQL* asociada se ejecuta siempre.

La cláusula WHEN no puede especificarse para los activadores INSTEAD OF (SQLSTATE 42613).

etiqueta:
Especifica la etiqueta de una sentencia de procedimiento SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden

utilizar la misma etiqueta. Varias sentencias de control SQL admiten la especificación de una lista de sentencias de procedimiento SQL.

Sólo la sentencia FOR, la sentencia WHILE y la sentencia compuesta dinámica pueden incluir una etiqueta.

sentencia-procedimiento-SQL
Especifica la sentencia de SQL que debe formar parte de la acción activada. No se da soporte a una operación de actualización de búsqueda, supresión de búsqueda, inserción o fusión de apodos en SQL compuesto.

La *sentencia-procedimiento-SQL* no debe contener una sentencia no soportada (SQLSTATE 42987).

La *sentencia-procedimiento-SQL* no puede hacer referencia a una variable de transición no definida (SQLSTATE 42703), a un objeto federado (SQLSTATE 42997) o a una tabla temporal declarada (SQLSTATE 42995).

La *sentencia-procedimiento-SQL* en un activador BEFORE no puede:

- Ser una sentencia CALL que invoque un procedimiento definido con MODIFIES SQL DATA, ni una sentencia MERGE (SQLSTATE 42987)
- Hacer referencia a una tabla de consultas materializadas definida con REFRESH IMMEDIATE (SQLSTATE 42997)
- Hacer referencia a una columna generada que no sea la columna de identidad de la variable de transición NEW (SQLSTATE 42989).

Notas:

- Al añadir un activador a una tabla que ya contiene filas, no provocará la activación de ninguna acción activada. Así pues, si el activador tiene como objetivo imponer las restricciones de la tabla, es posible que las filas existentes no cumplan dichas restricciones.
- Si los sucesos para dos activadores tienen lugar simultáneamente (por ejemplo, si tienen el mismo suceso, tiempo de activación y tablas sujeto), el primer activador creado es el primero en ejecutarse.
- Si se añade una columna a la tabla sujeto cuando ya se han definido los activadores, se aplican las siguientes normas:
  - Si se trata de un activador UPDATE que se ha especificado sin una lista de columnas explícita, cualquier actualización de una columna nueva provocará la activación del activador.
  - La columna no estará visible en la acción activada de cualquier activador definido con anterioridad.
  - Las tablas de transición OLD_TABLE y NEW_TABLE no contendrán esta columna. Por este motivo, el resultado de ″SELECT *″ en una tabla de transición no contendrá la columna añadida.
- Si se añade una columna a cualquier tabla a la que se haga referencia en una acción activada, la nueva columna no estará visible para la acción activada.
- El resultado de una selección completa especificada en una *sentencia-procedimiento-SQL* no está disponible dentro ni fuera del activador.
- Un procedimiento que se llama desde una sentencia compuesta activada no debe emitir las sentencias COMMIT ni ROLLBACK (SQLSTATE 42985).
- No se da soporte a un procedimiento que contiene una referencia a un apodo en una sentencia UPDATE de búsqueda, una sentencia DELETE de búsqueda o una sentencia INSERT (SQLSTATE 25000).
- ***Restricciones de acceso a las tablas:***

Si un procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del procedimiento puede acceder a una tabla que la sentencia compuesta que ha invocado el procedimiento esté modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del procedimiento puede modificar una tabla que la sentencia compuesta que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

- Un activador BEFORE DELETE definido en una tabla implicada en un ciclo de restricciones de referencia en cascada no debe incluir referencias a la tabla en la que está definido ni a ninguna otra tabla modificada en cascada durante la evaluación del ciclo de restricciones de integridad de referencia. El resultado de tal activador son datos dependientes y, por lo tanto, tal vez no produzcan resultados coherentes.

  En su forma más simple, significa que un activador BEFORE DELETE de una tabla con una restricción de referencia a si misma y una norma de supresión CASCADE no debe incluir ninguna referencia a la tabla en la *acción-activada*.
- La creación de un activador hace que se marquen determinados paquetes como no válidos:
  - Si se crea un activador UPDATE sin una lista de columnas explícita, se invalidan los paquetes que utilizan la actualización en la tabla o la vista de destino.
  - Si se crea un activador UPDATE con una lista de columnas, los paquetes que utilizan la actualización en la tabla de destino sólo se invalidan si el paquete también utiliza la actualización en como mínimo una columna de la lista *nombre-columna* de la sentencia CREATE TRIGGER.
  - Si se crea un activador INSERT, se invalidan los paquetes que utilizan la inserción en una tabla o vista de destino.
  - Si se crea un activador de supresión, los paquetes que hagan uso de la supresión en la tabla o vista de destino se invalidarán.
- Un paquete permanece invalidado hasta que el programa de aplicación se enlaza explícitamente, se vuelve a enlazar o se ejecuta y el gestor de bases de datos lo vuelve a enlazar de manera automática.
- ***Activadores no operativos***: Un *activador no operativo* es un activador que ya no está disponible y que, por lo tanto, nunca se activará. Un activador deja de ser operativo si:
  - se revoca un privilegio que el creador del activador debe tener para que se ejecute el activador
  - se elimina un objeto, por ejemplo una tabla, una vista o un alias, del que depende la acción activada
  - deja de estar operativa una vista de la que depende la acción activada
  - se elimina un alias que es la tabla sujeto del activador.

  En otras palabras, un activador no operativo es aquél en el que se ha eliminado la definición de un activador a consecuencia de las normas en cascada de las sentencias DROP y REVOKE. Por ejemplo, cuando se elimina una vista, deja de estar operativo cualquier activador con una *sentencia-procedimiento-SQL* definida utilizando dicha vista.

  Cuando un activador deja de ser operativo, todos los paquetes con sentencias que realicen operaciones y que estuvieran activando el activador quedarán marcados como no válidos. Cuando el paquete se vuelve a enlazar (explícita o implícitamente), se ignora por completo el activador no operativo. De igual forma, las aplicaciones con sentencias de SQL dinámico que realicen operaciones

y que estuvieran activando el activador también pasarán por alto por completo todos los activadores que no sean operativos.

El nombre del activador puede seguirse especificando en las sentencias DROP TRIGGER y COMMENT ON TRIGGER.

Es posible volver a crear un activador no operativo emitiendo una sentencia CREATE TRIGGER y utilizando el texto de definición del activador no operativo. Este texto de definición de activador se almacena en la columna TEXT de la vista de catálogo SYSCAT.TRIGGERS. Observe que no es necesario eliminar de manera explícita el activador no operativo para volver a crearlo. La emisión de una sentencia CREATE TRIGGER con el mismo *nombre-activador* que un activador no operativo hará que se sustituya dicho activador no operativo con un aviso (SQLSTATE 01595).

Los activadores no operativos se señalan con una X en la columna VALID de la vista de catálogo SYSCAT.TRIGGERS.

- ***Errores que se producen al ejecutar activadores***: Los errores que se producen durante la ejecución de las sentencias de SQL activadas se devuelven por medio de la utilización de SQLSTATE 09000 a menos que se considere que el error es grave. Si el error es grave, se devuelve el SQLSTATE de error grave. El campo SQLERRMC de la SQLCA de un error no grave incluirá el nombre de activador, el SQLCODE, el SQLSTATE y tantos símbolos de la anomalía como quepan.

  La *sentencia-procedimiento-SQL* puede incluir una sentencia SIGNAL SQLSTATE o una función RAISE_ERROR. En ambos casos, el SQLSTATE que se devuelve es el que se especifica en la sentencia SIGNAL SQLSTATE o la condición RAISE_ERROR.
- La creación de un activador con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Antes de ejecutar cualquier activador BEFORE, el gestor de bases de datos crea un valor para una columna de identidad. Por lo tanto, el activador BEFORE puede acceder al valor de identidad generado.
- Después de ejecutar todos activador BEFORE, el gestor de bases de datos crea un valor para una columna generada por expresión. Por lo tanto, los activadores BEFORE no pueden acceder al valor generado por la expresión.
- ***Activadores y tablas con tipo***: Un activador BEFORE o AFTER puede asociarse a una tabla con tipo en cualquier nivel de una jerarquía de tabla. Si una sentencia de SQL activa varios activadores, éstos se ejecutarán en el orden en que fueron creados, aunque estén asociados a tablas diferentes de la jerarquía de tablas con tipo.

  Cuando se activa un activador, sus variables de transición (OLD, NEW, OLD_TABLE y NEW_TABLE) pueden contener filas de subtablas. Sin embargo, sólo contendrán columnas definidas en la tabla a la que están asociadas.

  Efectos de las sentencias INSERT, UPDATE y DELETE:

  – Activadores de fila: Cuando se utiliza una sentencia de SQL para insertar, actualizar o suprimir una fila de tabla, la sentencia activa activadores de fila asociados a la tabla más específica donde reside la fila, y los activadores asociados a todas las supertablas de esa tabla. Esta norma se cumple siempre, cualquiera que sea la forma en que la sentencia de SQL accede a la tabla. Por ejemplo, al emitir un mandato UPDATE EMP, algunas de las filas actualizadas pueden estar en la subtabla MGR. Para las filas de EMP, se activan los activadores de fila asociados a EMP y a sus supertablas. Para las filas de MGR, se activan los activadores de fila asociados a MGR y a sus supertablas.

– Activadores de sentencia: Las sentencias INSERT, UPDATE o DELETE activan activadores de sentencia asociados a las tablas (y a sus supertablas) que podrían estar afectados por la sentencia. Esta norma se cumple siempre, con independencia de si hay realmente filas afectadas por la sentencia en esas tablas. Por ejemplo, en un mandato INSERT INTO EMP, se activan los activadores de sentencia para EMP y sus supertablas. Otro ejemplo: En un mandato UPDATE EMP o DELETE EMP, se activan los activadores para EMP y sus supertablas y subtablas, aunque no se haya actualizado ni suprimido ninguna fila de subtabla. Del mismo modo, un mandato UPDATE ONLY (EMP) o DELETE ONLY (EMP) activará activadores de sentencia para EMP y sus supertablas, pero no activadores de sentencia para subtablas.

Efectos de las sentencias DROP TABLE: Una sentencia DROP TABLE ("eliminar tabla") no activa ningún activador asociado a la tabla que se elimina. En cambio, si la tabla eliminada es una subtabla, todas las filas de la tabla eliminada son elegibles para ser suprimidas de sus supertablas. Por lo tanto, para una tabla T:

– Activadores de fila: DROP TABLE T activa activadores de supresión de filas que están asociados a todas las supertablas de T, para cada fila de T.
– Activadores de sentencia: DROP TABLE T activa activadores de supresión de sentencias que están asociados a todas las supertablas de T, sin importar si T contiene o no alguna fila.

Acciones sobre vistas: Para predecir qué activadores son activados por una acción sobre una vista, utilice la definición de la vista para convertir esa acción en una acción sobre tablas base. Por ejemplo:

1. Una sentencia de SQL ejecuta UPDATE V1, donde V1 es una vista con tipo y V2 es una subvista de ella. Suponga que V1 deriva de la tabla T1, y V2 deriva de la tabla T2. Esta sentencia podría potencialmente afectar a filas de T1, T2 y de sus subtablas, por lo tanto se activan activadores de sentencia para T1 y T2 y para todas sus subtablas y supertablas.
2. Una sentencia de SQL ejecuta UPDATE V1, donde V1 es una vista con tipo y V2 es una subvista de ella. Suponga que V1 está definida como SELECT ... FROM ONLY(T1) y V2 está definida como SELECT ... FROM ONLY(T2). Debido a que la sentencia no puede afectar a filas de subtablas de T1 y T2, se activan activadores de sentencia para T1 y T2 y para sus supertablas, pero no para sus subtablas.
3. Una sentencia de SQL ejecuta UPDATE ONLY(V1), donde V1 es una vista con tipo que está definida como SELECT ... FROM T1. La sentencia puede potencialmente afectar a T1 y a sus subtablas. Por lo tanto, se activan activadores de sentencia para T1 y para todas sus subtablas y supertablas.
4. Una sentencia de SQL ejecuta UPDATE ONLY(V1), donde V1 es una vista con tipo que está definida como SELECT ... FROM ONLY(T1). En este caso, T1 es la única tabla que puede verse afectada por la sentencia, aunque V1 tenga subvistas y T1 tenga subtablas. Por lo tanto, se activan activadores de sentencia sólo para T1 y sus supertablas.

| • ***Sentencia y activadores MERGE***: La sentencia MERGE puede ejecutar
| operaciones de actualización, supresión e inserción. Los activadores UPDATE,
| DELETE o INSERT aplicables se activan para la sentencia MERGE cuando se
| ejecuta una operación de actualización, supresión o inserción.

Ejemplos:

Ejemplo 1: Cree dos activadores que den como resultado un seguimiento automático del número de empleados que gestiona una empresa. Los activadores interactuarán con las tablas siguientes:

La tabla EMPLOYEE con estas columnas: ID, NAME, ADDRESS y POSITION.

La tabla COMPANY_STATS con estas columnas: NBEMP, NBPRODUCT y REVENUE.

El primer activador aumenta el número de empleados cada vez que se da de alta a una persona; es decir, cada vez que se inserta una nueva fila en la tabla EMPLOYEE.

```
CREATE TRIGGER NEW_HIRED
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

El segundo activador reduce el número de empleados cada vez que un empleado se marcha de la compañía; es decir, cada vez que se suprime una fila de la tabla EMPLOYEE:

```
CREATE TRIGGER FORMER_EMP
  AFTER DELETE ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

Ejemplo 2: Cree un activador que asegure que siempre que se actualice un registro de piezas, se lleven a cabo la siguiente comprobación y acción (si es necesaria):
Si la cantidad disponible es inferior al 10% de la cantidad máxima de existencias, se emitirá una petición de entrega solicitando el número de artículos de la pieza en cuestión sea la cantidad máxima en existencias menos la cantidad disponible.

El activador interactuará con la tabla PARTS con estas columnas: PARTNO, DESCRIPTION, ON_HAND, MAX_STOCKED y PRICE.

ISSUE_SHIP_REQUEST es una función definida por el usuario que envía un formulario de pedido de piezas adicionales a la empresa adecuada.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
  BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
  END
```

Ejemplo 3: Cree un activador que emita un error cuando se produzca una actualización que daría como resultado un aumento de salario mayor que el diez por ciento del salario actual.

```
CREATE TRIGGER RAISE_LIMIT
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING NEW AS N OLD AS O
  FOR EACH ROW
  WHEN (N.SALARY > 1.1 * O.SALARY)
         SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT='Salary increase>10%'
```

Ejemplo 4: Suponga que una aplicación registra y hace un seguimiento de los cambios en los precios de las existencias. Esta base de datos contiene dos tablas CURRENTQUOTE y QUOTEHISTORY.

```
Tablas: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
          QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)
```

Al actualizar la columna QUOTE de CURRENTQUOTE, el nuevo precio debería copiarse, con una indicación horaria, en la tabla QUOTEHISTORY. Asimismo, la columna STATUS de CURRENTQUOTE debería actualizarse para reflejar si las existencias:

1. son un valor en alza;
2. están en valor máximo del año;
3. son un valor a la baja;
4. están en el valor mínimo del año;
5. son un valor estable.

Las sentencias CREATE TRIGGER que realizan este cometido son las siguientes:

- Definición del activador para determinar el estado:

```
CREATE TRIGGER STOCK_STATUS
  NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
  FOR EACH ROW
  BEGIN ATOMIC
     SET NEWQUOTE.STATUS =
       CASE
          WHEN NEWQUOTE.QUOTE >
                (SELECT MAX(QUOTE) FROM QUOTEHISTORY
                WHERE SYMBOL = NEWQUOTE.SYMBOL
                AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
             THEN 'Alto'
          WHEN NEWQUOTE.QUOTE <
(SELECT MIN(QUOTE) FROM QUOTEHISTORY
                WHERE SYMBOL = NEWQUOTE.SYMBOL
                AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
             THEN 'Low'
          WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
             THEN 'En alza'
          WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
             THEN 'A la baja'
          WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
             THEN 'Estable'
        END;
  END
```

- Definición del activador para registrar un cambio en la tabla QUOTEHISTORY:

```
CREATE TRIGGER RECORD_HISTORY
  AFTER UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE
  FOR EACH ROW
  BEGIN ATOMIC
    INSERT INTO QUOTEHISTORY
      VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
  END
```

Información relacionada:

- "SQL compuesto (Dinámico)" en la página 130

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"
- "tbtrig.sqc -- How to use a trigger on a table (C)"
- "tbtrig.sqC -- How to use a trigger on a table (C++)"
- "trigsql.sqb -- How to use a trigger on a table (MF COBOL)"
- "TbTrig.java -- How to use triggers (JDBC)"
- "TbTrig.sqlj -- How to use triggers (SQLj)"

CREATE TYPE (Estructurado)

La sentencia CREATE TYPE define un tipo estructurado definido por el usuario. Un tipo estructurado definido por el usuario puede incluir cero o más atributos. Un tipo estructurado puede ser un subtipo que permita que los atributos se hereden de un supertipo. La ejecución satisfactoria de la sentencia genera métodos para recuperar y actualizar valores de atributos. La ejecución satisfactoria de la sentencia también genera funciones para crear instancias de un tipo estructurado usado en una columna, convertir entre el tipo de referencia y su tipo de representación, y para dar soporte a los operadores de comparación (=, <>, <, <=, > y >=) para el tipo de referencia.

La sentencia CREATE TYPE también define especificaciones de método para métodos definidos por el usuario, para su uso con el tipo estructurado definido por el usuario.

Invocación:

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

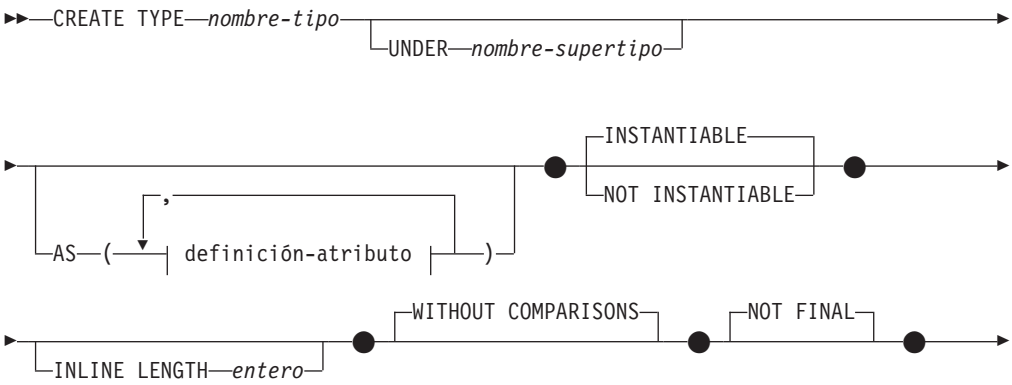
Autorización:

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

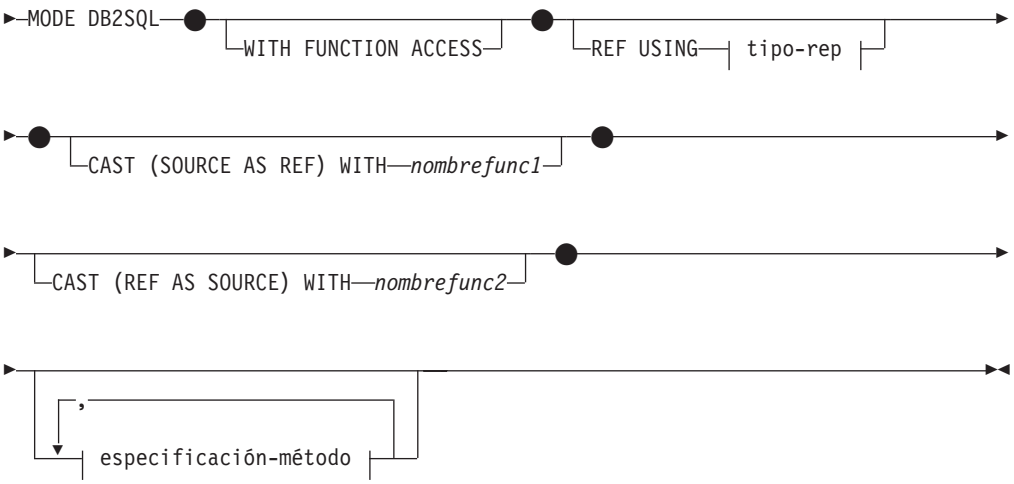
- Autorización SYSADM o DBADM
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo no hace referencia a un esquema existente.
- Privilegio CREATEIN para el esquema si el nombre de esquema del tipo hace referencia a un esquema existente.

Si se especifica UNDER y el ID de autorización de la sentencia no es el mismo que el definidor del tipo raíz de la jerarquía de tipos, entonces es necesaria la autorización SYSADM o DBADM.

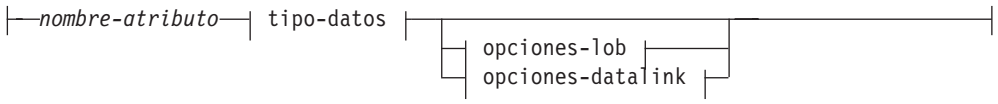
Sintaxis:



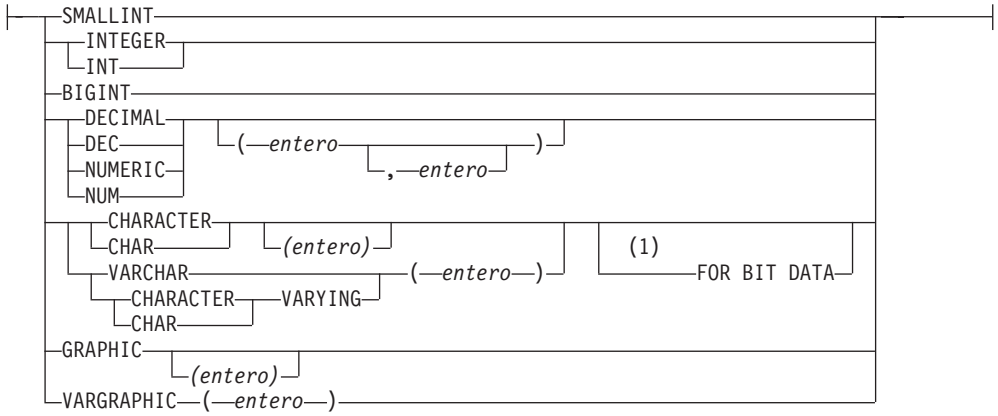
CREATE TYPE (Estructurado)



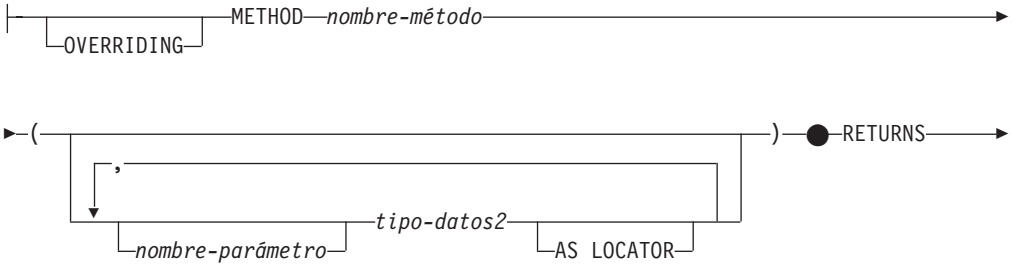
definición-atributo:

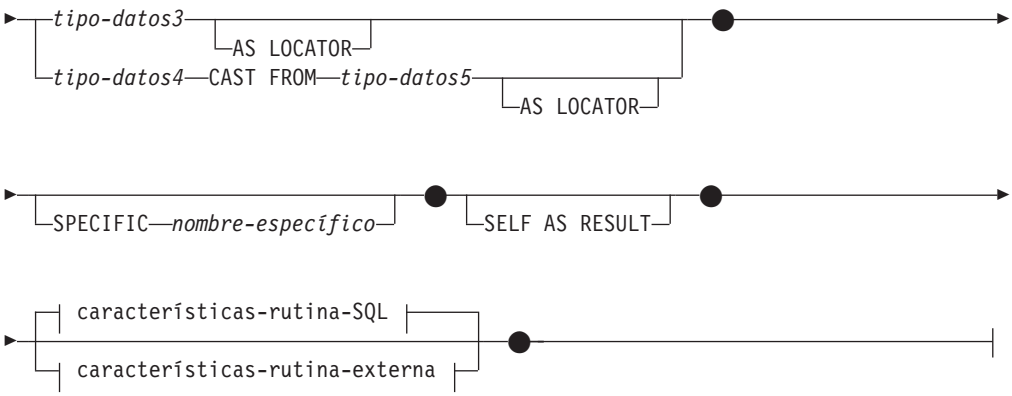


tipo-rep:

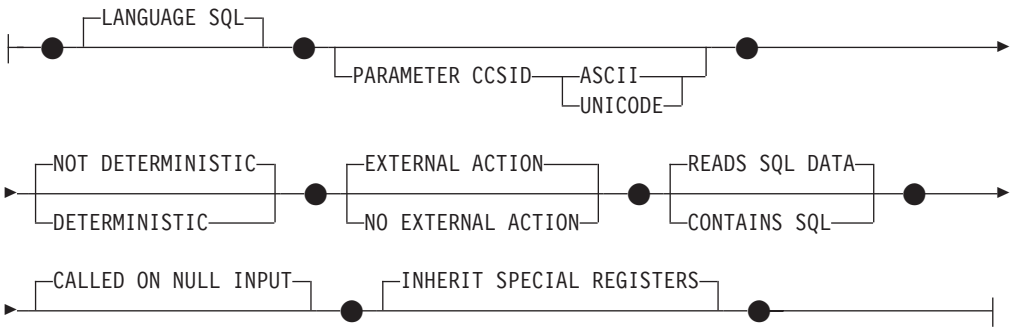


especificación-método:

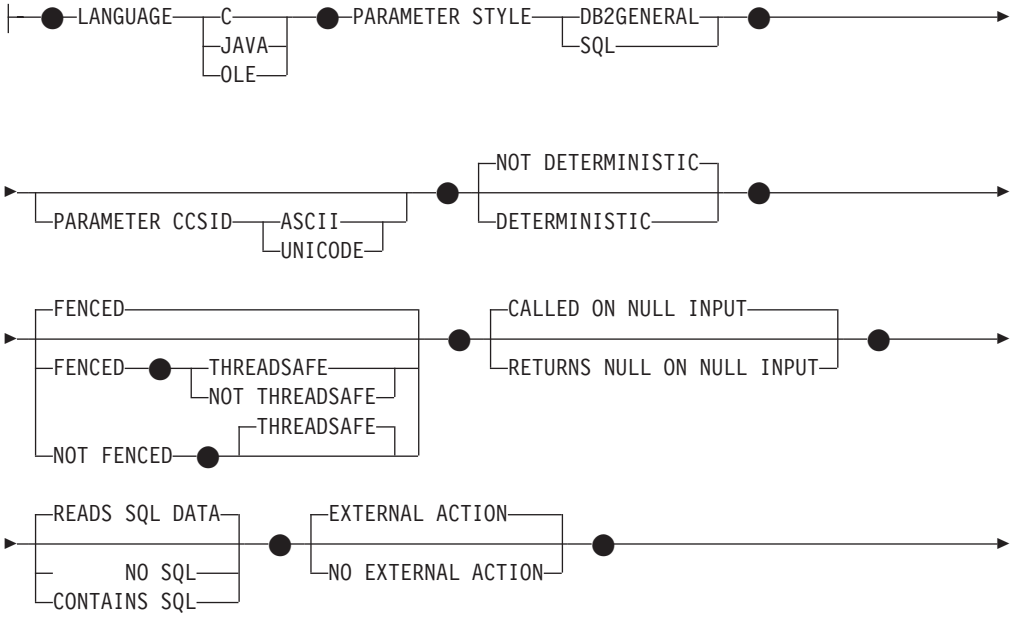


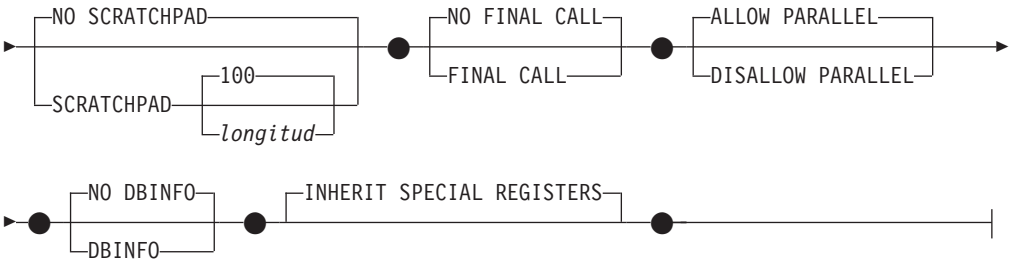


características-rutina-SQL:



características-rutina-externa:





Notas:

1 La cláusula FOR BIT DATA se puede especificar en cualquier orden con las restricciones de columna siguientes.

Descripción:

nombre-tipo
Indica el tipo. El nombre (incluido el calificador implícito o explícito) no debe designar ningún otro tipo (incorporado, estructurado o diferenciado) ya descrito en el catálogo. El nombre no calificado no debe ser el mismo que el de un tipo de datos incorporado o BOOLEAN (SQLSTATE 42918). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

El nombre de esquema (implícito o explícito) no debe ser mayor que 8 bytes (SQLSTATE 42622).

Algunos nombres utilizados como palabras clave en predicados están reservados para su uso por el sistema, y no pueden utilizarse como *nombre-tipo* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Si se especifica un *nombre-tipo* compuesto de dos partes, el nombre de esquema no puede empezar por 'SYS'; de lo contrario se devuelve un error (SQLSTATE 42939).

UNDER *nombre-supertipo*
Especifica que este tipo estructurado es un subtipo por debajo del *nombre-supertipo* especificado. El *nombre-supertipo* debe identificar un tipo estructurado existente (SQLSTATE 42704). Si *nombre-supertipo* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. El tipo estructurado incluye todos los atributos del supertipo seguidos de los atributos adicionales que se proporcionan en la *definición-atributo*.

definición-atributo
Define los atributos del tipo estructurado.

nombre-atributo
El nombre de un atributo. El *nombre-atributo* no puede ser el mismo que el de otro atributo de este tipo estructurado o un supertipo de este tipo estructurado (SQLSTATE 42711).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como un *nombre-atributo* (SQLSTATE 42939). Los nombres son SOME, ANY,

ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

tipo-datos
El tipo de datos del atributo. Es uno de los tipos de datos que aparecen en la lista de "CREATE TABLE" que no sea LONG VARCHAR, LONG VARGRAPHIC o un tipo diferenciado basado en LONG VARCHAR o LONG VARGRAPHIC (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". Si el tipo de datos del atributo es un tipo de referencia, el tipo destino de la referencia debe ser un tipo estructurado existente o se crea mediante esta sentencia (SQLSTATE 42704).

Si un tipo estructurado está definido con un atributo de tipo DATALINK, sólo puede utilizarse de manera efectiva como tipo de datos para una tabla con vista o vista con tipo (SQLSTATE 01641).

Para evitar las definiciones de tipo que, durante la ejecución, permitirían que una instancia del tipo contuviera directa o indirectamente otra instancia del mismo tipo o uno de sus subtipos, un tipo no puede definirse de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP).

opciones-lob
Especifica las opciones asociadas a tipos LOB (o tipos diferenciados basados en tipos LOB). Para obtener una descripción detallada de *opciones-lob*, consulte "CREATE TABLE".

opciones-datalink
Especifica las opciones asociadas a tipos DATALINK (o tipos diferenciados basados en tipos DATALINK). Para obtener una descripción detallada de *opciones-datalink*, consulte "CREATE TABLE".

Tenga en cuenta que, si no se especifican opciones para un tipo DATALINK o un tipo diferenciado cuyo origen sea DATALINK, los valores por omisión serán las opciones LINKTYPE URL y NO LINK CONTROL.

INSTANTIABLE o **NOT INSTANTIABLE**
Determina si se puede crear una instancia del tipo estructurado. El no poder crear una instancia de un tipo estructurado tiene estas consecuencias:

- no se genera ninguna función constructora para un tipo para el que no se pueden crear instancias
- el tipo que no admite creación de instancias no puede utilizarse como tipo de una tabla o vista (SQLSTATE 428DP)
- el tipo que no admite creación de instancias se puede utilizar como tipo de una columna (sólo pueden insertarse en la columna valores nulos o instancias de subtipos que permiten la creación de instancias.

Para crear instancias de un tipo que no permite crear instancias, se deben crear subtipos que permiten crear instancias. Si se especifica NOT INSTANTIABLE, no se puede crear ninguna instancia del nuevo tipo.

INLINE LENGTH *entero*
Esta opción indica el tamaño máximo, en bytes, de una instancia de columna de tipo estructurado para su almacenamiento "inline" con el resto de valores de la fila de una tabla. Las instancias de un tipo estructurado o de sus subtipos

que son mayores que la longitud ″inline″ especificada se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB.

Si el valor especificado de INLINE LENGTH es menor que el tamaño del resultado de la función constructora para el tipo recién creado (32 bytes más 10 bytes por atributo) y menor que 292 bytes, se produce un error (SQLSTATE 429B2). Observe que el número de atributos incluye todos los atributos heredados a partir del supertipo del tipo.

El valor INLINE LENGTH del tipo, ya sea especificado explícitamente o tomado por omisión, es la longitud ″inline″ por omisión de las columnas que hacen uso del tipo estructurado. Este valor por omisión se puede alterar temporalmente por otro valor al crear la tabla.

El valor INLINE LENGTH no es aplicable cuando el tipo estructurado se utiliza como tipo de una tabla con tipo.

El valor por omisión de INLINE LENGTH para un tipo estructurado es calculado por el sistema. En la fórmula mostrada más abajo se utilizan los términos siguientes:

atributo corto
: designa un atributo que tiene uno de los tipos de datos siguientes: SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, FLOAT, DATE o TIME. También comprende los tipos diferenciados o tipos de referencia basados en esos tipos.

atributo no corto
: designa un atributo perteneciente a cualquiera de los tipos de datos restantes o a los tipos diferenciados basados en esos tipos de datos.

El sistema calcula la longitud ″inline″ por omisión de este modo:

1. Determina las necesidades adicionales de espacio para atributos no cortos, mediante la fórmula siguiente:

   espacio_para_atributos_no_cortos = SUM(*longitudAtributo* + n)

   n se define como:

   - 0 bytes para atributos de tipo estructurado anidado
   - 2 bytes para atributos que no son de LOB
   - 9 bytes para atributos de LOB

   longitudAtributo está basado en el tipo de datos especificado para el atributo, tal como se describe en la Tabla 9.
2. Calcula la longitud total ″inline″ por omisión, mediante esta fórmula:

   longitud_por_omisión(*tipo_estructurado*) = (*número_de_atributos* * 10) + 32 + *espacio_para_atributos_no_cortos*

   número_de_atributos es el número total de atributos del tipo estructurado, incluidos los atributos que se heredan del supertipo del tipo. En cambio, *número_de_atributos* no incluye ningún atributo definido para cualquier subtipo del *tipo_estructurado*.

Tabla 9. Contajes de bytes para los tipos de datos de atributos

Tipo de datos de atributo	Contaje de bytes
DECIMAL	Parte entera de (p/2)+1, donde p es la precisión
CHAR (n)	n

Tabla 9. Contajes de bytes para los tipos de datos de atributos (continuación)

Tipo de datos de atributo	**Contaje de bytes**	
VARCHAR (n)	n	
GRAPHIC (n)	n * 2	
VARGRAPHIC (n)	n * 2	
TIMESTAMP	10	
DATALINK(n)	n + 54	
Tipo LOB	Cada atributo LOB tiene un descriptor de LOB, en la instancia del tipo estructurado, que apunta a la ubicación del valor real. El tamaño del descriptor varía de acuerdo con la longitud máxima definida para el atributo LOB	
	Longitud máxima de LOB	Tamaño del descriptor de LOB
	1 024	72
	8 192	96
	65 536	120
	524 000	144
	4 190 000	168
	134 000 000	200
	536 000 000	224
	1 070 000 000	256
	1 470 000 000	280
	2 147 483 647	316
Tipo diferenciado	Longitud del tipo de fuente del tipo diferenciado	
Tipo de referencia	Longitud del tipo de datos incorporado en el que está basado el tipo de referencia.	
Tipo estructurado	longitud_en-línea(*tipo_atributo*)	

WITHOUT COMPARISONS
Indica que no se da soporte a funciones de comparación para las instancias del tipo estructurado.

NOT FINAL
Indica que el tipo estructurado puede utilizarse como supertipo.

MODE DB2SQL
Esta cláusula es obligatoria y permite la invocación directa de la función constructora para el tipo.

WITH FUNCTION ACCESS
Indica que todos los métodos del tipo y de sus subtipos, incluidos los métodos que se creen en el futuro, se pueden acceder utilizando la notación funcional. Esta cláusula sólo puede especificarse para el tipo raíz de una jerarquía de tipos estructurados (la cláusula UNDER no se especifica) (SQLSTATE 42613). Esta cláusula se proporciona para permitir el uso de la notación funcional para aquellas aplicaciones que prefieren esta forma de notación respecto a la notación de invocación de método.

REF USING *tipo-rep*
Define el tipo de datos incorporados utilizados como representación (tipo de datos subyacente) para el tipo de referencia de este tipo estructurado y de

todos sus subtipos. Esta cláusula sólo puede especificarse para el tipo raíz de una jerarquía de tipos estructurados (la cláusula UNDER no se especifica) (SQLSTATE 42613). El *tipo-rep* no puede ser un tipo LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK o un tipo estructurado y debe tener una longitud menor que o igual a 32 672 bytes (SQLSTATE 42613).

Si esta cláusula no se especifica para el tipo raíz de una jerarquía de tipos estructurados, entonces se supone REF USING VARCHAR(16) FOR BIT DATA.

CAST (SOURCE AS REF) WITH *nombrefunc1*
Define el nombre de la función, generada por el sistema, que convierte un valor cuyo tipo de datos es *tipo-rep* al tipo de referencia de este tipo estructurado. No debe especificarse un nombre de esquema como parte de *nombrefunc1* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc1* es *nombre-tipo* (el nombre del tipo estructurado). El esquema no debe contener ya una signatura de función que coincida con *nombrefunc1(tipo-rep)*.

CAST (REF AS SOURCE) WITH *nombrefunc2*
Define el nombre de la función, generada por el sistema, que convierte un valor de tipo de referencia para este tipo estructurado al tipo de datos *tipo-rep*. No se debe especificar un nombre de esquema como parte de *nombrefunc2* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc2* es *tipo-rep* (el nombre del tipo de representación).

especificación-método
Define los métodos correspondientes a este tipo. Para utilizar un método es necesario proporcionarle un cuerpo mediante una sentencia CREATE METHOD (SQLSTATE 42884).

OVERRIDING
Especifica que el método que está definiéndose altera temporalmente a un método de un supertipo del tipo que está definiéndose. La alteración temporal permite volver a implementar métodos en los subtipos y, por lo tanto, ofrece funciones más específicas. La alteración temporal no recibe soporte para los siguientes tipos de métodos:

- Métodos de tabla y fila
- Métodos externos declarado con PARAMETER STYLE JAVA
- Métodos que puede utilizarse como predicados en una extensión de índice
- Métodos de mutación o de observador generados por el sistema

Si se intenta alterar temporalmente uno de estos métodos, se generará un error (SQLSTATE 42745).

Si un método es un método de alteración temporal válido, deberá existir un método original para uno de los supertipos adecuados del tipo que está definiéndose y deberán existir las relaciones siguientes entre el método de alteración temporal y el método original:

- El nombre de método del método que está definiéndose y el método original son equivalentes.
- El método que está definiéndose y el método original tienen el mismo número de parámetros.
- El tipo de datos de cada parámetro del método que está definiéndose y el tipo de datos de los parámetros correspondientes del método original son idénticos. Este requisito excluye al parámetro SELF implícito.

Si no existe un método original con estas características, se devolverá un error (SQLSTATE 428FV).

El método de alteración temporal hereda los siguientes atributos del método original:

- Lenguaje
- Indicación de determinismo
- Indicación de acción externa
- Una indicación que especifique si este método debe o no llamarse si alguno de sus argumentos es el valor nulo
- Conversión del resultado (si se ha especificado en el método original)
- Indicación SELF AS RESULT
- La indicación de acceso a los datos de SQL o CONSTAINS SQL
- Para los métodos externos:
  - Estilo del parámetro
  - Indicación del localizador de los parámetros y del resultado (si se ha especificado en el método original)
  - Indicación FENCED, SCRATCHPAD, FINAL CALL, ALLOW PARALLEL y DBINFO
  - Indicación INHERIT SPECIAL REGISTER y THREADSAFE

nombre-método
Designa el método que se está definiendo. Debe ser un identificador SQL no calificado (SQLSTATE 42601). El nombre de método está calificado implícitamente por el esquema utilizado para CREATE TYPE.

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como un *nombre-método* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

En general, puede utilizarse el mismo nombre para más de un método si existe alguna diferencia en la signatura de los métodos.

nombre-parámetro
Designa el nombre del parámetro. Este nombre no puede ser SELF, que es el nombre del parámetro sujeto implícito de un método (SQLSTATE 42734). Si método es un método SQL, todos sus parámetros deben tener nombres (SQLSTATE 42629). Si el método que está declarándose altera temporalmente a otro método, el nombre del parámetro deberá ser exactamente el mismo nombre que el del correspondiente parámetro del método alterado temporalmente; de lo contrario, se devolverá un error (SQLSTATE 428FV).

tipo-datos2
Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el método espera recibir. No se permiten más de 90 parámetros, incluido el parámetro implícito SELF. Si se excede este límite, se produce un error (SQLSTATE 54023).

Se pueden utilizar las especificaciones y abreviaturas de tipos de datos SQL que puedan especificarse como tipo-columna en una sentencia CREATE TABLE y que tengan una correspondencia en el lenguaje utilizado para escribir la método. Consulte las secciones específicas del lenguaje del manual ″Application Development Guide″ para conocer

detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal con respecto a las funciones y métodos definidos por el usuario.

Nota: Si el tipo de datos SQL en cuestión es un tipo estructurado, no existe una correlación por omisión con un tipo de datos de lenguaje principal. Se debe utilizar una función de transformación definida por el usuario para crear una correlación entre el tipo estructurado y el tipo de datos de lenguaje principal.

DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).

Se puede especificar REF, pero no tiene un ámbito definido. Dentro del cuerpo del método, se puede utilizar un tipo de referencia en una expresión de vía de acceso sólo si primero se convierte el tipo para que tenga un ámbito. Similarmente, una referencia devuelta por un método se puede utilizar en una expresión de vía de acceso sólo si primero se convierte para que tenga un ámbito.

AS LOCATOR
Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB al método, en lugar de pasarle el valor real. Esto reduce considerablemente el número de bytes que se pasan al método y puede también mejorar el rendimiento, especialmente cuando el método sólo necesite unos pocos bytes.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Si el método que está declarándose altera temporalmente a otro método, la indicación AS LOCATOR del parámetro deberá coincidir exactamente con la indicación AS LOCATOR del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

Si el método que está declarándose altera temporalmente a otro método, la indicación FOR BIT DATA de cada parámetro deberá coincidir exactamente con la indicación FOR BIT DATA del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

RETURNS
Esta cláusula obligatoria identifica el resultado del método.

tipo-datos3
Especifica el tipo de datos del resultado del método. En este caso, son válidas las mismas consideraciones que para los parámetros de métodos, descritas anteriormente bajo tipo-datos2.

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB desde el método, en lugar del valor real.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si el método altera temporalmente a otro método, *tipo-datos3* debe ser un subtipo del tipo de datos del resultado del método alterado temporalmente si este tipo de datos es un tipo estructurado; de lo contrario, ambos tipos de datos deben ser idénticos (SQLSTATE 428FV).

tipo-datos4 **CAST FROM** *tipo-datos5*
Especifica el tipo de datos del resultado del método.

Esta cláusula se utiliza para devolver a la sentencia invocadora un tipo de datos diferente que el tipo de datos devuelto por el método. El *tipo-datos5* debe ser convertible al parámetro *tipo-datos4*. Si no es convertible, se produce un error (SQLSTATE 42880).

Debido a que la longitud, precisión o escala de *tipo-datos4* puede inferirse de *tipo-datos5*, no es necesario (pero está permitido) especificar la longitud, precisión o escala de los tipos con parámetros especificados para *tipo-datos4*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo, VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601), pues el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

No puede especificarse un tipo diferenciado como tipo para tipo-datos5 (SQLSTATE 42815).

La operación de conversión del tipo de datos también está sujeta a comprobaciones durante la ejecución que pueden dar lugar a errores de conversión.

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB desde el método, en lugar del valor real.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse la cláusula FOR BIT DATA (SQLSTATE 428FV).

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia del método que se está definiendo. Este nombre específico puede utilizarse al crear el cuerpo del método o al eliminar el método. No puede utilizarse nunca para invocar el método. La forma no calificada de *nombre-específico* es un identificador SQL (cuya longitud máxima es 18). La forma calificada es un nombre de esquema

seguido de un punto y un identificador SQL. El nombre (incluido el calificador implícito o explícito) no debe designar otro nombre de método específico que exista en el servidor de aplicaciones; de lo contrario se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser el mismo que un *nombre-método* existente.

Si no se especifica ningún calificador, se emplea el calificador que se utilizó para *nombre-tipo*. Si se especifica un calificador, debe ser el mismo que el calificador explícito o implícito de *nombre-tipo*, de lo contrario se produce un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

SELF AS RESULT
Identifica este método como un método de conservación de tipo, lo que significa lo siguiente:

- El tipo de retorno declarado debe ser el mismo que el tipo sujeto declarado (SQLSTATE 428EQ).
- Cuando una sentencia de SQL se compila y su resolución da un tipo conservador del método, el tipo estático del resultado del método es el mismo que el tipo estático del argumento sujeto.
- El método debe implementarse de forma que el tipo dinámico del resultado sea el mismo que el tipo dinámico del argumento sujeto (SQLSTATE 2200G), y el resultado no puede ser NULL (SQLSTATE 22004).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

características-rutina-SQL
Especifica las características del cuerpo del método que se definirán para este tipo utilizando CREATE METHOD.

LANGUAGE SQL
Esta cláusula se utiliza para indicar que el método está escrito en SQL mediante una sola sentencia RETURN. El cuerpo del método se especifica utilizando la sentencia CREATE METHOD.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al método de SQL y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE
Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

NOT DETERMINISTIC o **DETERMINISTIC**
Esta cláusula opcional especifica si el método siempre devuelve los mismos resultados para valores de argumento determinados (DETERMINISTIC) o si el método depende de ciertos valores de estado que afectan a los

resultados (NOT DETERMINISTIC). Es decir, un método DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. Se debe especificar NOT DETERMINISTIC, de forma explícita o implícita, si el cuerpo del método accede a un registro especial, o invoca otra rutina no determinista (SQLSTATE 428C2).

EXTERNAL ACTION o **NO EXTERNAL ACTION**
Esta cláusula opcional especifica si el método realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que el método no produce ningún efecto externo, se especifica EXTERNAL ACTION. Por ejemplo: enviar un mensaje, hacer sonar una alarma o grabar un registro en un archivo.

READS SQL DATA o **CONTAINS SQL**
Indica qué tipo de sentencias de SQL se pueden ejecutar. Debido a que la sentencia de SQL soportada es la sentencia RETURN, la distinción está relacionada con el hecho de si la expresión es una subconsulta o no.

READS SQL DATA
Indica que el método puede ejecutar sentencias de SQL que no modifican datos SQL (SQLSTATE 42985). No se pueden utilizar apodos en la sentencia de SQL (SQLSTATE 42997).

CONTAINS SQL
Indica que el método puede ejecutar sentencias de SQL que no leen ni modifican datos SQL (SQLSTATE 42985).

CALLED ON NULL INPUT
Esta cláusula opcional indica que debe invocarse el método definido por el usuario aunque contenga argumentos nulos. Puede devolver un valor nulo o un valor normal (no nulo). Sin embargo, corresponde al método comprobar si existen valores de argumento nulos.

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del método heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para un método que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno en el que se ha abierto el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

características-rutina-externa

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje para el está escrito el cuerpo del método definido por el usuario.

C Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera una función escrita en C. El método definido por el usuario debe cumplir el convenio de invocación y enlace del lenguaje C, tal como está definido por el prototipo C estándar de ANSI.

JAVA
Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera un método de una clase Java.

OLE
Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera un método expuesto por un objeto de automatización OLE. El método debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

Sólo se da soporte a LANGUAGE OLE para métodos definidos por el usuario almacenados en Sistemas operativos Windows de 32 bits. THREADSAFE no puede especificarse para los métodos que se han definido con LANGUAGE OLE (SQLSTATE 42613).

PARAMETER STYLE
Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros al método y obtener el resultado del método.

DB2GENERAL
Se utiliza para especificar los convenios para pasar parámetros a métodos externos y obtener sus resultados; estos métodos están definidos como tales en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

SQL
Se utiliza para especificar los convenios para pasar parámetros a métodos externos y obtener su resultado; estos métodos cumplen los convenios de invocación y enlace del lenguaje C o son métodos expuestos por objetos de automatización OLE. Esta opción debe especificarse cuando se utiliza LANGUAGE C o LANGUAGE OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al método externo y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE
Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

Esta cláusula no se puede especificar con LANGUAGE OLE (SQLSTATE 42613).

DETERMINISTIC o **NOT DETERMINISTIC**
Esta cláusula opcional especifica si el método siempre devuelve los mismos resultados para valores de argumento determinados (DETERMINISTIC) o si el método depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un método DETERMINISTIC siempre debe devolver el mismo resultado para invocaciones sucesivas con entradas de datos idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

Un ejemplo de método NOT DETERMINISTIC sería un método que devuelve al azar un número de identificación de un empleado de un departamento. Un ejemplo de método DETERMINISTIC sería un método que calcula el área de un polígono.

FENCED o **NOT FENCED**
Esta cláusula especifica si el método se considera ″seguro″ (NOT FENCED) o no (FENCED) para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos.

Si un método se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios) para que el método no pueda acceder a ellos. La mayoría de los métodos tienen la posibilidad de ejecutarse como FENCED o NOT FENCED. En general, un método que se ejecute como FENCED no funcionará tan bien como otro método similar que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para métodos no probados debidamente puede comprometer la integridad de DB2. DB2 dispone de algunos mecanismos para hacer frente a la mayoría de los tipos de errores involuntarios más habituales que pueden producirse, pero no puede garantizar la integridad completa cuando se utilizan métodos NOT FENCED definidos por el usuario.

Para un método con LANGUAGE OLE o NOT THREADSAFE, sólo puede especificarse FENCED (SQLSTATE 42613).

Si el método es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

Para registrar un método como NOT FENCED, se necesita autorización SYSADM, autorización DBADM o una autorización especial (CREATE_NOT_FENCED_ROUTINE).

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

Si el método se ha definido con un LANGUAGE distinto de OLE:

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

Para los métodos FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si el método se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para los métodos NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o **CALLED ON NULL INPUT**
Esta cláusula opcional sirve para evitar la invocación de un método externo si cualquiera de los argumentos no sujetos es nulo.

Si se especifica RETURNS NULL ON NULL INPUT y al ejecutar el método alguno de sus argumentos es nulo, no se invoca el método y el resultado es el valor nulo.

Si se especifica CALLED ON NULL INPUT, se invoca el método con independencia de si hay argumentos nulos. Puede devolver un valor nulo o un valor normal (no nulo). Sin embargo, corresponde al método comprobar si existen valores de argumento nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

Existen dos casos para los que no se tiene en cuenta esta especificación:

- Si el argumento sujeto es nulo. En este caso el método no se ejecuta y el resultado es nulo.
- Si el método está definido para no tener parámetros. En este caso la condición de argumento nulo no puede producirse.

NO SQL, CONTAINS SQL, READS SQL DATA
Indica si el método debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que el método no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL
Indica que el método puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ningún método devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en el método pueden incluirse algunas sentencias de SQL

que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ningún método devuelven un error distinto (SQLSTATE 38003 ó 42985).

EXTERNAL ACTION o **NO EXTERNAL ACTION**
Esta cláusula opcional especifica si el método realiza alguna acción que cambia el estado de un objeto no gestionado por el gestor de bases de datos. Para evitar las optimizaciones basadas en que el método no produce ningún efecto externo, se especifica EXTERNAL ACTION.

NO SCRATCHPAD o **SCRATCHPAD** *longitud*
Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para un método externo. Es muy recomendable que los métodos sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que el método "guarde el estado" entre una llamada y la siguiente.

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación del método definido por el usuario, se asigna memoria para que el método externo utilice una memoria de trabajo. Esta memoria de trabajo tiene las características siguientes:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para cada referencia al método externo contenida en la sentencia de SQL.

Por tanto, si el método X de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarán tres memorias de trabajo.

```
SELECT A, X..(A) FROM TABLEB
   WHERE X..(A) > 103 OR X..(A) < 19
```

Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si el método se ejecuta en varias particiones, se asigna una memoria de trabajo en cada partición donde se procesa el método, una para cada referencia al método contenida en la sentencia de SQL. De manera similar, si la consulta se ejecuta con el paralelismo de intrapartición habilitado, pueden asignarse más de tres memorias de trabajo.

La memoria de trabajo es permanente. Su contenido se conserva entre una llamada al método externo y la siguiente. Los cambios hechos en la memoria de trabajo por el método externo en una llamada seguirán allí en la llamada siguiente. El gestor de bases de datos inicializa las memorias de trabajo al principio de la ejecución de cada sentencia de SQL. El gestor de bases de datos puede restaurar las memorias de trabajo al comienzo de la ejecución de cada subconsulta. El sistema emite una llamada final antes de restaurar una memoria de trabajo si se especifica la opción FINAL CALL.

La memoria de trabajo puede utilizarse como punto central de los recursos del sistema (por ejemplo, la memoria) que podría adquirir el método externo. El método podría adquirir la memoria en la primera llamada, conservar su dirección en la memoria de trabajo y acceder a ella en llamadas posteriores.

En el caso en que se adquiere el recurso del sistema, también debe especificarse la palabra clave FINAL CALL; esto provoca una llamada

especial en el final de la sentencia para permitir que el método externo libere los recursos del sistema adquiridos.

Si se especifica SCRATCHPAD, en cada invocación del método definido por el usuario se pasa un argumento adicional al método externo que indica la dirección de la memoria de trabajo.

Si se especifica NO SCRATCHPAD, no se asignará ni pasará ninguna memoria de trabajo al método externo.

NO FINAL CALL o **FINAL CALL**
Esta cláusula opcional especifica si debe realizarse una llamada final a un método externo. La finalidad de esta llamada final es permitir que el método externo libere los recursos que ha adquirido del sistema. Junto con la palabra clave SCRATCHPAD, está cláusula puede ser útil en situaciones donde el método externo adquiere recursos del sistema, tales como memoria, y los fija en la memoria de trabajo.

Si se especifica FINAL CALL, durante la ejecución se pasa al método externo un argumento adicional que especifica el tipo de llamada. Los tipos de llamadas son:

- Llamada normal: Se pasan argumentos SQL y se espera la devolución de un resultado.
- Primera llamada: es la primera llamada al método externo para esta referencia específica al método contenida en esta sentencia de SQL específica. La primera llamada es una llamada normal.
- Llamada final: es una llamada final al método externo para permitir que el método libere recursos. La llamada final no es una llamada normal. Esta llamada final tiene lugar en los momentos siguientes:
  - Fin de sentencia: Esta situación se produce cuando se cierra el cursor, en el caso de sentencias basadas en cursores, o cuando la sentencia termina su ejecución de otra manera.
  - Fin de transacción: Este caso se produce cuando no se da un fin de sentencia normal. Por ejemplo, cuando por alguna razón la lógica de una aplicación ignora el cierre del cursor.

  Si se produce una operación de confirmación mientras está abierto un cursor definido como WITH HOLD, se realiza una llamada final en el cierre subsiguiente del cursor o al final de la aplicación.

Si se especifica NO FINAL CALL, no se pasa al método externo ningún argumento que especifique el tipo de llamada, y no se realiza ninguna llamada final.

ALLOW PARALLEL o **DISALLOW PARALLEL**
Esta cláusula opcional especifica si, para una referencia individual al método, puede paralelizarse la invocación del método. En general, las invocaciones de la mayoría de los métodos escalares pueden paralelizarse, pero puede haber métodos (tales como los que dependen de una sola copia de una memoria de trabajo) que no puedan. Si se especifica ALLOW PARALLEL o DISALLOW PARALLEL para un método, DB2 aceptará esta especificación.

Deben tenerse en cuenta las cuestiones siguientes al determinar qué palabra clave es la adecuada para el método:

- ¿Son todas las invocaciones del método completamente independientes las unas de las otras? En caso afirmativo, especifique ALLOW PARALLEL.
- ¿Se actualiza la memoria de trabajo con cada invocación del método, proporcionando valores que son de interés para la siguiente invocación (el incremento de un contador, por ejemplo)? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Realiza el método alguna acción externa que deba producirse en una sola partición? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique ALLOW PARALLEL.

En cualquier caso, el cuerpo de cada método externo debe estar en un directorio que sea accesible en todas las particiones de la base de datos.

El diagrama de sintaxis indica que el valor por omisión es ALLOW PARALLEL. Sin embargo, el valor por omisión es DISALLOW PARALLEL si en la sentencia se especifican una o más de las opciones siguientes:

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO o **DBINFO**

Esta cláusula opcional especifica si se pasará (DBINFO) o no (NO DBINFO) al método cierta información específica conocida por DB2, en forma de argumento adicional de invocación. NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613). Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si se especifica DBINFO, al método se le envía una estructura que contiene la siguiente información:

- Nombre de base de datos - el nombre de la base de datos actualmente conectada.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, sin tener en cuenta los métodos anidados existentes entre este método y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - exactamente bajo las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - si la referencia a método es la parte derecha de una cláusula SET en una sentencia UPDATE o es un elemento de la lista VALUES de una sentencia INSERT, este dato es el nombre no calificado de la tabla que se está actualizando o insertando; en otro caso, está en blanco.
- Nombre de columna - si se dan las mismas condiciones que para el Nombre de tabla, este dato es el nombre de la columna que se está actualizando o insertando; en otro caso, está en blanco.

- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos invocador del método.
- Plataforma - contiene el tipo de plataforma del servidor.
- Números de columna del resultado del método de tabla - no es aplicable a métodos.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales del método heredarán sus valores iniciales de la sentencia que realiza la llamada. Para los cursores, los valores iniciales se heredan del valor de tiempo que corresponde a la apertura del cursor.

Al proceso que realiza la llamada del método no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Algunos registros especiales como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, nunca heredan valores del proceso que realiza la llamada.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - Se admite la sintaxis siguiente:
      - NOT VARIANT puede especificarse en lugar de DETERMINISTIC
      - VARIANT puede especificarse en lugar de NOT DETERMINISTIC
      - NULL CALL puede especificarse en lugar de CALLED ON NULL INPUT
      - NOT NULL CALL puede especificarse en lugar de RETURNS NULL ON NULL INPUT
    - Se acepta la sintaxis siguiente como comportamiento por omisión para los métodos externos:
      - ASUTIME NO LIMIT
      - NO COLLID
      - PROGRAM TYPE SUB
      - STAY RESIDENT NO
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode si no se especifica PARAMETER CCSID UNICODE
    - Se acepta la sintaxis siguiente como comportamiento por omisión para los métodos de SQL:
      - CCSID UNICODE en una base de datos Unicode
      - CCSID ASCII en una base de datos que no es Unicode
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - PARAMETER STYLE DB2SQL puede especificarse en lugar de PARAMETER STYLE SQL
- La creación de un tipo estructurado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.

- Un subtipo estructurado definido sin atributos define un subtipo que hereda todos sus atributos de un supertipo. Si no se especifica una cláusula UNDER ni ningún otro atributo, el tipo es un tipo raíz de una jerarquía de tipos, sin ningún atributo.
- La adición de un nuevo subtipo a una jerarquía de tipos puede causar la invalidación de paquetes. Un paquete puede quedar invalidado si depende de un supertipo del nuevo tipo. Dicha dependencia es el resultado de la utilización de un predicado TYPE o de una especificación TREAT.
- Un tipo estructurado puede tener 4082 atributos como máximo (SQLSTATE 54050).
- Una especificación de método no puede tener la misma signatura que una función (cuando se compara el primer tipo-parámetro de la función con el tipo-sujeto del método).
- Ningún método original puede alterar temporalmente a otro método, o ser alterado temporalmente por un método original (SQLSTATE 42745). Además, en una relación de alteración temporal no puede haber una función y un método. Esto significa que si se ha considerado que la función es un método, siendo su primer parámetro el sujeto S, no deberá alterar temporalmente a otro método de ningún supertipo de S y no podrá alterarlo temporalmente otro método de ningún subtipo de S (SQLSTATE 42745).
- La creación de un tipo estructurado genera automáticamente un conjunto de funciones y métodos que pueden utilizarse con el tipo. Todas las funciones y métodos se generan en el mismo esquema que el tipo estructurado. Si la signatura de la función o método generados entra en conflicto con o altera temporalmente la signatura de una función existente en este esquema, la sentencia falla (SQLSTATE 42710). Las funciones o métodos generados no pueden eliminarse sin eliminar el tipo estructurado (SQLSTATE 42917). Se generan las funciones y métodos siguientes:
  - Funciones
    - Comparaciones de referencia

      Se generan seis funciones de comparación denominadas =, <>, <, <=, >, >= para el tipo de referencia REF(*nombre-tipo*). Cada una de estas funciones toma dos parámetros del tipo REF(*nombre-tipo*) y devuelve el valor de verdadero, falso o desconocido. Los operadores de comparación para REF(*nombre-tipo*) se han definido para que tengan el mismo comportamiento que los operadores de comparación para el tipo de datos subyacente de REF(*nombre-tipo*). (Todas las referencias de una jerarquía de tipo tienen el mismo tipo de representación de referencia. Esto permite que REF(S) y REF(T) puedan compararse, siempre que S y T tengan un supertipo común. Debido a que la exclusividad de la columna de OID sólo se aplica dentro de una jerarquía de tabla, es posible que un valor de REF(T) de una jerarquía de tabla sea "igual" a un valor de REF(T) de otra jerarquía de tabla, aunque hagan referencia a filas distintas.)

      El ámbito del tipo de referencia no se tiene en cuenta en la comparación.
    - Funciones de conversión

      Se generan dos funciones de conversión para convertir entre el tipo de referencia generado REF(*nombre-tipo*) y el tipo de datos subyacente de este tipo de referencia.

      - El nombre de la función para convertir el tipo subyacente al tipo de referencia es el *nombrefunc1* implícito o explícito.

        El formato de esta función es:

        ```
        CREATE FUNCTION nombrefunc1 (tipo-rep)
          RETURNS REF(nombre-tipo) ...
        ```

- El nombre de la función para convertir el tipo de referencia al tipo subyacente del tipo de referencia es el *nombrefunc2* implícito o explícito.

  El formato de esta función es:

  ```
  CREATE FUNCTION nombrefunc2 ( REF(nombre-tipo) )
    RETURNS tipo-rep ...
  ```

  Para algunos tipos-rep, existen funciones de conversión adicionales generadas con *nombrefunc1* para manejar las conversiones desde constantes.

  - Si *tipo-rep* es SMALLINT, la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 (INTEGER)
      RETURNS REF(nombre-tipo)
    ```

  - Si *tipo-rep* es CHAR(n), la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 ( VARCHAR(n))
      RETURNS REF(nombre-tipo)
    ```

  - Si *tipo-rep* es GRAPHIC(n), la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 (VARGRAPHIC(n))
      RETURNS REF(nombre-tipo)
    ```

  El nombre de esquema del tipo estructurado debe incluirse en la vía de acceso de SQL para que estos operadores y funciones de conversión se utilicen satisfactoriamente en las sentencias de SQL.

- Función constructora

  La función constructora se genera para permitir la creación de una nueva instancia del tipo. Esta nueva instancia tendrá el valor nulo para todos los atributos del tipo, incluidos los atributos que se heredan de un supertipo.

  El formato de la función constructora generada es:

  ```
  CREATE FUNCTION nombre-tipo ( )
    RETURNS nombre-tipo
   ...
  ```

  Si se especifica NOT INSTANTIABLE, no se genera ninguna función constructora. Si el tipo estructurado tiene atributos de tipo DATALINK, la invocación de la función constructora no surte efecto (SQLSTATE 428ED).

– Métodos

- Métodos observadores

  Se define un método observador para cada atributo del tipo estructurado. Para cada atributo, el método observador devuelve el tipo del atributo. Si el sujeto es nulo, el método observador devuelve un valor nulo del tipo de atributo.

  Por ejemplo, los atributos de una instancia del tipo estructurado ADDRESS se pueden observar utilizando `C1..STREET`, `C1..CITY`, `C1..COUNTRY` y `C1..CODE` .

  La signatura del método observador generado es como si se hubiera ejecutado la sentencia siguiente:

  ```
  CREATE TYPE  nombre-tipo
      ...
    METHOD nombre-atributo()
      RETURNS tipo-atributo
  ```

  donde *nombre-tipo* es el nombre del tipo estructurado.

- Métodos mutadores

Se define un método mutador preservador del tipo para cada atributo del tipo estructurado. Utilice métodos mutadores para cambiar atributos dentro de una instancia de un tipo estructurado. Para cada atributo, el método mutador devuelve una copia del sujeto modificada por la asignación del argumento al atributo mencionado de la copia.

Por ejemplo, una instancia del tipo estructurado ADDRESS se puede mutar utilizando `C1..CODE('M3C1H7')`. Si el sujeto es nulo, el método mutador emite un error (SQLSTATE 2202D).

La signatura del método mutador generado es como si se hubiera ejecutado la sentencia siguiente:

```
CREATE TYPE  nombre-tipo
        ...
     METHOD nombre-atributo (tipo-atributo)
     RETURNS nombre-tipo
```

Si el tipo de datos del atributo es SMALLINT, REAL, CHAR o GRAPHIC, se genera un método mutador adicional para dar soporte a la mutación utilizando constantes:

- Si *tipo-atributo* es SMALLINT, el mutador adicional da soporte a un argumento de tipo INTEGER.
- Si *tipo-atributo* es REAL, el mutador adicional da soporte a un argumento de tipo DOUBLE.
- Si *tipo-atributo* es CHAR, el mutador adicional da soporte a un argumento de tipo VARCHAR.
- Si *tipo-atributo* es GRAPHIC, el mutador adicional da soporte a un argumento de tipo VARGRAPHIC.

- Si el tipo estructurado se utiliza como tipo de columna, la longitud máxima de una instancia del tipo es 1 GB durante la ejecución (SQLSTATE 54049).

- Cuando se crea un nuevo subtipo para un tipo estructurado existente (para utilizarlo como tipo de columna), se deben reexaminar y actualizar según sea necesario las funciones de transformación existentes que dan soporte a los tipos estructurados asociados. Tanto si el nuevo tipo está en la misma jerarquía que un tipo determinado o en la jerarquía de un tipo anidado, es probable que la función de transformación asociada a este tipo deba modificarse para incluir algunos o todos los nuevos atributos originados por el nuevo subtipo. En términos generales, puesto que es el conjunto de funciones de transformación que se asocia a un tipo determinado (o jerarquía de tipo) que permite a UDF y a la aplicación cliente acceder al tipo estructurado, las funciones de transformación deben escribirse de modo que se dé soporte a *todos* los atributos de una jerarquía compuesta determinada (es decir, incluyendo el cierre transitivo de todos los subtipos y sus tipo estructurados anidados).

  Cuando se crea un nuevo subtipo de un tipo existente, se invalidan todos los paquetes que dependen de los métodos que se han definido en los supertipos del tipo que está creándose y que pueden elegirse para la alteración temporal.

- ***Restricciones de acceso a las tablas***

  Si un método se ha definido como READS SQL DATA, ninguna sentencia del método podrá acceder a una tabla que la sentencia que ha invocado el método está modificando (SQLSTATE 57053). Por ejemplo, supongamos que el método BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE DEPTINFO SET SALARY = SALARY + EMP..BONUS(), no podrá leerse ninguna sentencia de SQL del método BONUS desde la tabla EMPLOYEE.

- ***Privilegios***

  - El usuario que define el tipo definido por el usuario siempre recibe el privilegio EXECUTE y WITH GRANT OPTION para todos los métodos y funciones que se generan automáticamente para el tipo estructurado. El privilegio EXECUTE no se otorgará para ninguno de los métodos explícitamente especificados en la sentencia CREATE TYPE hasta que se haya definido un cuerpo de método mediante la utilización de la sentencia CREATE METHOD. El usuario que define el tipo definido por el usuario no dispone del derecho para eliminar la especificación de método mediante la utilización de la sentencia ALTER TYPE. A PUBLIC se otorga el privilegio EXECUTE para todas las funciones que automáticamente se generan durante CREATE DISTINCT TYPE.
  - Cuando en una sentencia de SQL se utiliza un método externo, el usuario que define el método debe disponer de privilegio EXECUTE para cualquiera de los paquetes que el método utiliza.
- En un entorno de bases de datos particionadas, el uso de SQL en funciones o métodos externos definidos por el usuario no recibe soporte (SQLSTATE 42997).
- Sólo las rutinas que se han definido como NO SQL pueden utilizarse para definir una extensión de índice (SQLSTATE 428F8).
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.

Ejemplos:

Ejemplo 1: Cree un tipo para Department.

```
CREATE TYPE DEPT AS
   (DEPT NAME     VARCHAR(20),
      MAX_EMPS INT)
      REF USING INT
   MODE DB2SQL
```

Ejemplo 2: Cree una jerarquía de tipos compuesta de un tipo para los empleados y de un subtipo para los directores.

```
CREATE TYPE EMP AS
  (NAME      VARCHAR(32),
  SERIALNUM INT,
  DEPT      REF(DEPT),
  SALARY    DECIMAL(10,2))
  MODE DB2SQL

CREATE TYPE MGR UNDER EMP AS
  (BONUS     DECIMAL(10,2))
  MODE DB2SQL
```

Ejemplo 3: Cree una jerarquía de tipos para direcciones. Las direcciones están destinadas a utilizarse como tipos de columnas. La longitud ″inline″ no se especifica, por lo que DB2 calculará una longitud por omisión. Dentro de la definición del tipo de dirección se encapsulará un método externo que calcula el grado de proximidad de la dirección con una dirección de entrada proporcionada. El cuerpo de la sentencia se crea mediante la sentencia CREATE METHOD.

```
CREATE TYPE address_t AS
  (STREET     VARCHAR(30),
  NUMBER     CHAR(15),
  CITY       VARCHAR(30),
  STATE      VARCHAR(10))
  NOT FINAL
  MODE DB2SQL
    METHOD SAMEZIP (addr address_t)
    RETURNS INTEGER
```

```
    LANGUAGE SQL
    DETERMINISTIC
    CONTAINS SQL
    NO EXTERNAL ACTION

    METHOD DISTANCE (address_t)
    RETURNS FLOAT
    LANGUAGE C
    DETERMINISTIC
    PARAMETER STYLE SQL
    NO SQL
    NO EXTERNAL ACTION

CREATE TYPE germany_addr_t UNDER address_t AS
  (FAMILY_NAME VARCHAR(30))
  NOT FINAL
  MODE DB2SQL

CREATE TYPE us_addr_t UNDER address_t AS
  (ZIP VARCHAR(10))
  NOT FINAL
  MODE DB2SQL
```

Ejemplo 4: Creación de un tipo que tenga atributos de tipo estructurado anidados.

```
CREATE TYPE PROJECT AS
  (PROJ_NAME  VARCHAR(20),
   PROJ_ID    INTEGER,
   PROJ_MGR   MGR,
   PROJ_LEAD  EMP,
   LOCATION   ADDR_T,
   AVAIL_DATE DATE)
   MODE DB2SQL
```

Información relacionada:

- "Predicado básico" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE TABLE" en la página 341
- "SET PATH" en la página 744
- "Registros especiales" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)"

CREATE TYPE MAPPING

La sentencia CREATE TYPE MAPPING crea una correlación entre los siguientes tipos de datos:

- El tipo de datos de una columna de una tabla o una vista de fuente de datos que se va a definir para una base de datos federada
- Un tipo de datos correspondiente que ya está definido para la base de datos federada

La correlación puede asociar el tipo de datos de base de datos federada con un tipo de datos de:

- Una fuente de datos especificada
- Un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular

Una correlación de tipo de datos sólo debe crearse si una correlación existente no es adecuada.

Si se pueden aplicar varias correlaciones de tipos al crear un apodo o una tabla (DDL transparente), se aplicará la más reciente.

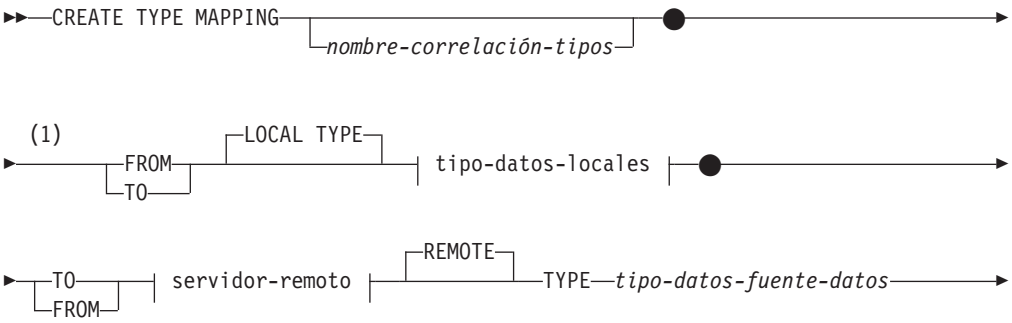
Invocación:

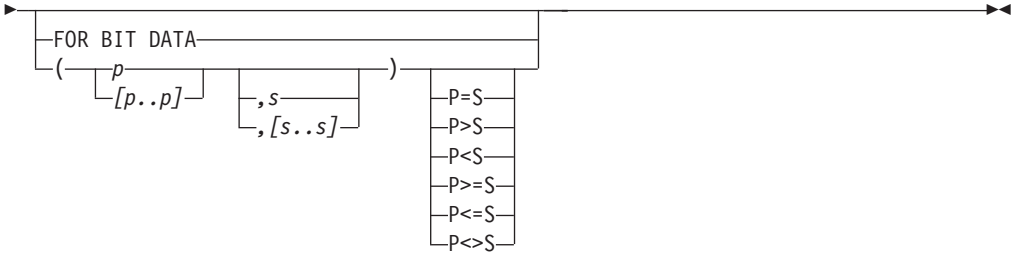
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

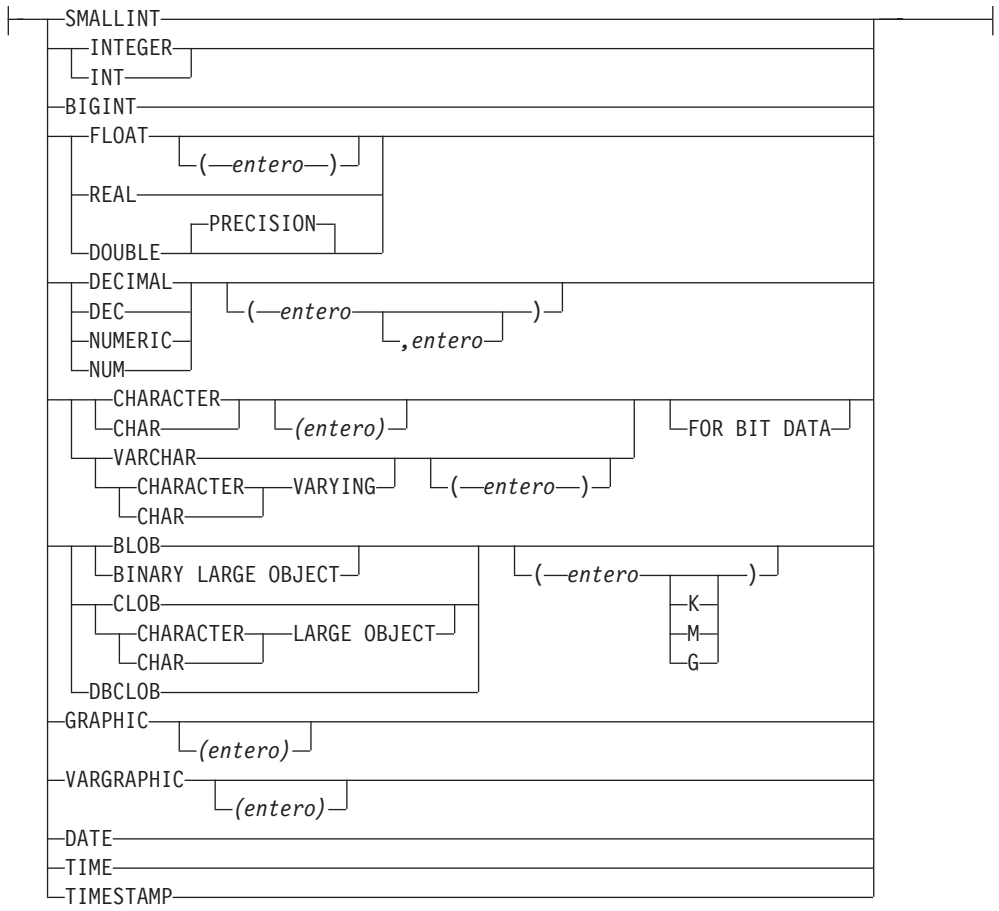
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM.

Sintaxis:

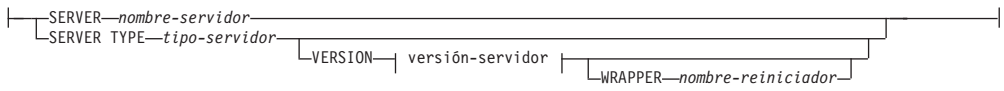




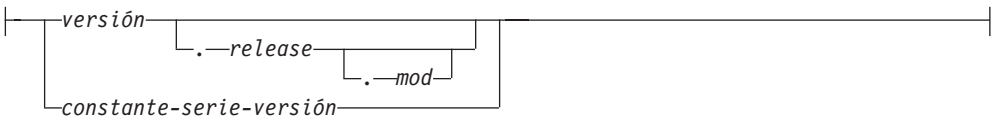
tipo-datos-locales:



servidor-remoto:



versión-servidor:



Notas:

1 En la sentencia CREATE TYPE MAPPING, deberán existir una palabra clave TO y una palabra clave FROM.

Descripción:

nombre-correlación-tipos
Designa la correlación de tipos de datos. El nombre no debe identificar ninguna correlación de datos que ya esté descrita en el catálogo. Se genera un nombre exclusivo si no se especifica *nombre-correlación-tipos*.

FROM o **TO**
Especifica una correlación de tipos invertida o en avance.

FROM
Especifica una correlación de tipos en avance cuando va seguida del *tipo-datos-local* o una correlación de tipos invertida cuando va seguida del *servidor-remoto*.

TO
Especifica una correlación de datos en avance cuando va seguida del *servidor-remoto* o una correlación de tipo de datos invertida cuando va seguida del *tipo-datos-local*.

tipo-datos-locales
Identifica un tipo de datos que se define en una base de datos federada. Si se especifica el *tipo-datos-local* sin un nombre de esquema, el nombre de tipo se resuelve buscando los esquemas en la vía de acceso de SQL.

Se pueden utilizar paréntesis vacíos para los tipos de datos con parámetros. Un tipo de datos con parámetros es cualquiera de los tipos de datos que se puede definir con una longitud, una escala o una precisión específica. Si se especifican paréntesis vacíos en una correlación de tipos en avance como, por ejemplo, CHAR(), la longitud se determina a partir de la longitud de la columna de la tabla remota. Si se especifican paréntesis vacíos en una correlación de tipo invertida, la correlación de tipos se aplica al tipo de datos con cualquier longitud. Si se omiten los paréntesis, se utilizará la longitud por omisión para el tipo de datos (consulte la descripción de la sentencia CREATE TABLE).

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

El *tipo-datos-local* no puede ser LONG VARCHAR, LONG VARGRAPHIC, DATALINK ni un tipo definido por el usuario (SQLSTATE 42611).

SERVER *nombre-servidor*
Nombra la fuente de datos para la que se define *tipo-datos-fuente-datos*.

SERVER TYPE *tipo-servidor*
Identifica el tipo de fuente de datos para el que se define el *tipo-datos-fuente-datos*.

VERSION
Identifica la versión de la fuente de datos para el que se define el *tipo-datos-fuente-datos*.

versión
Especifica el número de versión. El valor debe ser un entero.

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

mod
Especifica el número de la modificación del release indicado por *release*. El valor debe ser un entero.

constante-serie-versión
Especifica la designación completa de la versión. La *constante-serie-versión* puede ser un solo valor (por ejemplo, '8i') o puede estar formada por los valores concatenados de *versión*, *release* y, si se aplica, *mod* (por ejemplo, '8.0.3').

WRAPPER *nombre-reiniciador*
Especifica el nombre del reiniciador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión que *tipo-servidor* y *versión-servidor* indican.

TYPE *tipo-datos-fuente-datos*
Especifica el tipo de datos de fuente de datos que se está correlacionando con el tipo de datos local.

Se pueden utilizar paréntesis vacíos para los tipos de datos con parámetros. Si se especifican paréntesis vacíos en una correlación de tipos en avance como, por ejemplo, CHAR(), la correlación de tipos se aplica al tipo de datos con cualquier longitud. Si se especifican paréntesis vacíos en una correlación de tipo de datos invertida, la longitud se determina a partir de la longitud de columna especificada en el DDL transparente. Si se omiten los paréntesis, se utiliza la longitud por omisión para el tipo de datos.

El *tipo-datos-fuente-datos* debe ser un tipo de datos incorporado. No están permitidos los tipos definidos por el usuario.

Si se especifica el *nombre-servidor* con una correlación de tipos o los servidores existentes se ven afectados por la correlación de tipos, *tipo-datos-fuente-datos*, *p* y *s* se verifican al crear la correlación de tipos (SQLSTATE 42611).

p Si se especifica *p*, el tipo de correlación sólo afecta al tipo de datos cuya longitud o precisión sea igual a *p*.

[p1..p2]
Sólo para la correlación de tipos en avance. Para un tipo de datos decimal, *p1* y *p2* especifican el número mínimo y máximo de dígitos que un valor puede tener. Para tipos de datos de serie, *p1* y *p2* especifican el número mínimo y máximo de caracteres que un valor puede tener. En todos los casos, el máximo debe ser igual o exceder del mínimo; y ambos números deben ser válidos con respecto al tipo de datos.

s Si se especifica *s*, sólo el tipo de datos cuya escala sea igual a *s* se ve afectado por la correlación de tipos.

[s1..s2]
Sólo para la correlación de tipos en avance. Para un tipo de datos decimal, *s1* y *s2* especifican el número mínimo y máximo de dígitos permitidos a la derecha

de la coma decimal. El máximo debe ser igual o superior al mínimo y ambos números deben ser válidos con respecto al tipo de datos.

P [operando] **S**
El tipo de datos decimal, P *[operando]* S especifica una comparación entre la precisión y el número de dígitos permitidos a la derecha de la coma decimal. Por ejemplo, el operando = indica que se aplica la correlación de tipos si la precisión y el número de dígitos permitidos en la fracción decimal son iguales.

FOR BIT DATA
Indica si *tipo-datos-fuente-datos* es para los datos de bits. Estas palabras clave son necesarias si la columna del tipo de la fuente de datos contiene valores binarios. El gestor de base de datos determinará este atributo si no se especifica para un tipo de datos de caracteres.

Notas:

- Una sentencia CREATE TYPE MAPPING de una unidad de trabajo determinada (UOW) no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- Cuando se pueden aplicar varias correlaciones de tipos, se utilizará la más reciente. Puede recuperar la hora de creación para una correlación de tipos consultando la columna CREATE_TIME de la vista del catálogo SYSCAT.TYPEMAPPINGS.

Ejemplos:

Ejemplo 1: Cree una correlación de tipos en avance entre el tipo de datos DATE de Oracle y el tipo de datos SYSIBM.DATE. Para todos los apodos que se creen después de haber definido esta correlación, las columnas del tipo de datos DATE de Oracle se correlacionarán con columnas DB2 del tipo de datos DATE.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
  FROM LOCAL TYPE SYSIBM.DATE
  TO SERVER TYPE ORACLE
  REMOTE TYPE DATE
```

Ejemplo 2: Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.DECIMAL(10,2) y el tipo de datos NUMBER([10..38],2) en la fuente de datos ORACLE1. Si hay una columna en la tabla Oracle del tipo de datos

NUMBER(11,2), se correlacionará con una columna de datos de tipo DECIMAL(10,2), porque 11 está entre 10 y 38.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  FROM LOCAL TYPE SYSIBM.DECIMAL(10,2)
   TO SERVER ORACLE1
  REMOTE TYPE NUMBER([10..38],2)
```

Ejemplo 3: Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.VARCHAR(*p*) y el tipo de datos CHAR(*p*) de Oracle en la fuente de datos ORACLE1 (*p* es cualquier longitud). Si hay una columna en la tabla Oracle del tipo de datos CHAR(10), se correlacionará con una columna del tipo de datos VARCHAR(10).

```
CREATE TYPE MAPPING MY_ORACLE_CHAR
  FROM LOCAL TYPE SYSIBM.VARCHAR()
   TO SERVER ORACLE1
  REMOTE TYPE CHAR()
```

Ejemplo 4: Cree una correlación de tipos invertida del tipo de datos NUMBER(10,2) de Oracle en la fuente de datos ORACLE2 y el tipo de datos SYSIBM.DECIMAL(10,2). Si utiliza un DDL transparente para crear una tabla Oracle y especifica una columna de tipo de datos DECIMAL(10,2), DB2 creará la tabla Oracle con una columna de tipo de datos NUMBER(10,2).

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  TO LOCAL TYPE SYSIBM.DECIMAL(10,2)
  FROM SERVER ORACLE2
  REMOTE TYPE NUMBER(10,2)
```

Información relacionada:

- "CREATE TABLE" en la página 341

CREATE USER MAPPING

La sentencia CREATE USER MAPPING define una correlación entre un ID de autorización que utiliza una base de datos federada y el ID de autorización y la contraseña que se deben utilizar en una fuente de datos especificada.

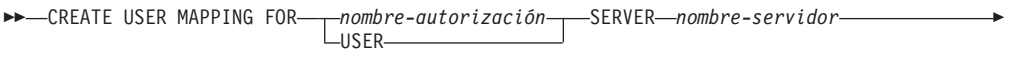
Invocación:

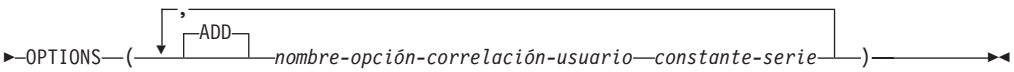
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

Si el ID de autorización de la sentencia es diferente del nombre de autorización que se correlaciona con la fuente de datos, los privilegios del ID de autorización de la sentencia deben incluir las autorizaciones SYSADM o DBADM. Por lo demás, si el ID de autorización y el nombre de autorización coinciden, no se requieren privilegios ni autorizaciones.

Sintaxis:





Descripción:

nombre-autorización
Especifica el nombre de autorización bajo el cual un usuario o una aplicación se conecta a una base de datos federada. El *nombre-autorización* se correlaciona con la opción de correlación de usuarios REMOTE_AUTHID.

USER
El valor del registro especial USER. Cuando se especifica USER, el ID de autorización que emite la sentencia CREATE USER MAPPING se correlaciona con la opción de correlación de usuarios REMOTE_AUTHID.

SERVER *nombre-servidor*
Nombra el objeto de servidor para la fuente de datos a la que el *nombre-autorización* puede acceder. El *nombre-servidor* es el nombre local para el servidor remoto que se ha registrado en la base de datos federada.

OPTIONS
Indica las opciones que están habilitadas cuando se crea la correlación de usuarios.

ADD
Habilita una o varias opciones de correlación de usuarios.

nombre-opción-correlación-usuario
Especifica el nombre de la opción.

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

Notas:

- Las correlaciones de usuarios sólo se necesitan para las siguientes fuentes de datos: la familia de productos DB2, Documentum, Informix, Microsoft SQL Server, ODBC, Oracle, Sybase y Teradata.
- Siempre se requiere la opción REMOTE_PASSWORD para una correlación de usuarios.

Ejemplos:

Ejemplo 1: Registre una correlación de usuarios en el objeto servidor de fuente de datos DB2 para z/OS y OS/390 SERVER390. Correlacione el nombre de autorización para la base de datos federada local con el ID de usuario y contraseña para SERVER390. El nombre de autorización es RSPALTEN. El ID de usuario para SERVER390 es SYSTEM. La contraseña para SERVER390 es MANAGER.

```
CREATE USER MAPPING FOR RSPALTEN
  SERVER SERVER390
  OPTIONS
  (REMOTE_AUTHID 'SYSTEM',
  REMOTE_PASSWORD 'MANAGER')
```

Ejemplo 2: Registre una correlación de usuarios para el objeto de servidor de fuente de datos Oracle ORACLE1. MARCR es el nombre de autorización para la base de datos federada local y el ID de usuario para ORACLE1. Puesto que el nombre de autorización y el ID de usuario son iguales, no es necesario especificar la opción REMOTE_AUTHID en la correlación de usuarios. La contraseña para MARCR en ORACLE1 es NZXCZY.

```
CREATE USER MAPPING FOR MARCR
    SERVER ORACLE1
    OPTIONS
  (REMOTE_PASSWORD 'NZXCZY')
```

Información relacionada:

- "User mapping options for federated systems" en la publicación *Federated Systems Guide*

CREATE VIEW

La sentencia CREATE VIEW crea una vista para una o más tablas, vistas o apodos.

Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM, o bien
- Para cada tabla, vista o apodo identificado en cualquier selección completa:
  - Privilegio CONTROL para esa tabla o vista, o bien
  - Privilegio SELECT para esa tabla o vista

  y como mínimo uno de los elementos siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la vista no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema de la vista hace referencia a un esquema existente.

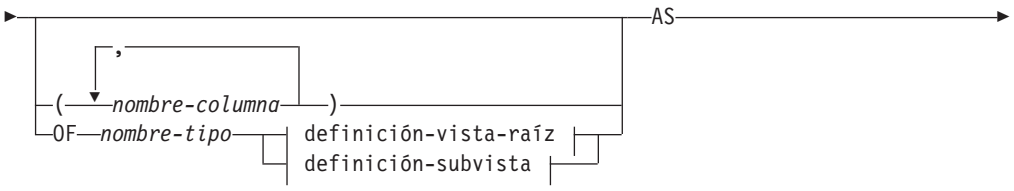
  Si se crea una subvista, el ID de autorización de la sentencia debe:
  - Ser el mismo que el definidor de la tabla raíz de la jerarquía de tablas.
  - Tener SELECT WITH GRANT en la tabla principal de la subvista o el privilegio SELECT sobre la supervista no debe haberse otorgado a un usuario que no sea el definidor de la vista.

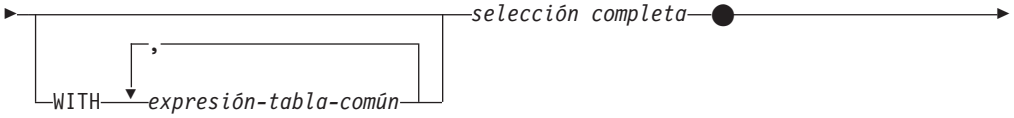
No se tienen en cuenta los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE VIEW.

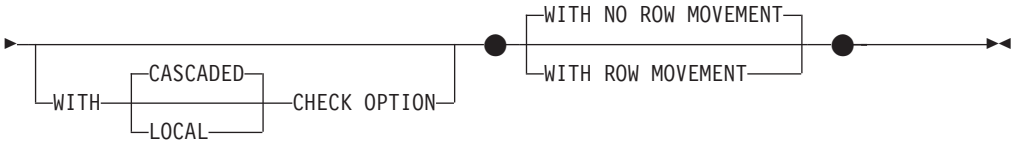
Los privilegios no se tienen en cuenta al definir una vista en un apodo de base de datos federada. Los requisitos de autorización de la fuente de datos para la tabla o vista referenciada por el apodo se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización remoto distinto.

Si el definidor de una vista sólo puede crear la vista porque tiene autorización SYSADM, se le otorgará la autorización explícita DBADM con el propósito de crear la vista.

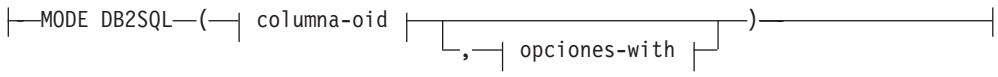
Sintaxis:



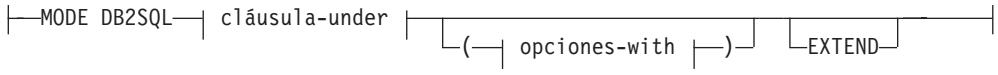




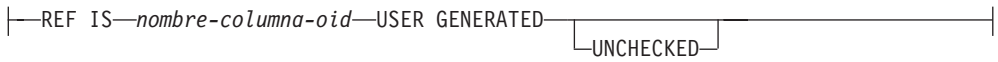
definición-vista-raíz:



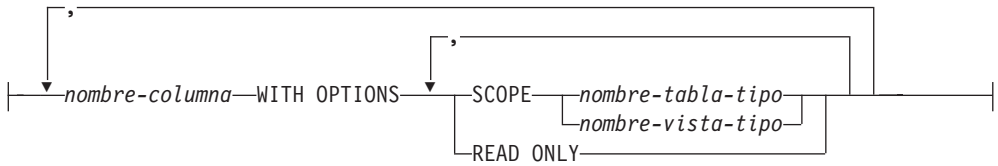
definición-subvista:



columna-oid:



opciones-with:



cláusula-under:

Descripción:

nombre-vista
Indica el nombre de la vista. El nombre (incluido el calificador implícito o

explícito) no debe designar una tabla, vista, apodo ni alias que se haya descrito en el catálogo. El calificador no debe ser SYSIBM, SYSCAT, SYSFUN ni SYSSTAT (SQLSTATE 42939).

El nombre puede ser el mismo que el nombre de una vista no operativa (consulte el apartado "Vistas no operativas" en la página 475). En tal caso, la nueva vista especificada en la sentencia CREATE VIEW sustituirá la vista no operativa. El usuario recibirá un aviso (SQLSTATE 01595) cuando se sustituya una vista no operativa. No se devuelve ningún aviso si la aplicación se ha enlazado con la opción de enlace lógico SQLWARN establecida en NO.

nombre-columna
Indica los nombres de las columnas de la vista. Si se especifica una lista de nombres de columna, ésta debe constar de tantos nombres como columnas haya en la tabla de resultados de la selección completa. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifica la lista de nombres de columnas, las columnas de la vista heredarán los nombres de las columnas de la tabla de resultados de la selección completa.

Debe especificarse una lista de nombres de columna si la tabla de resultados de la selección completa tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es una columna derivada de una constante, función, expresión u operación de conjuntos que no se designa utilizando la cláusula AS de la lista de selección.

OF *nombre-tipo*
Especifica que las columnas de la vista están basadas en los atributos del tipo estructurado identificado por *nombre-tipo*. Si se especifica *nombre-tipo* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). El nombre de tipo debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se pueda crear una instancia (SQLSTATE 428DP).

MODE DB2SQL
Esta cláusula se utiliza para especificar la modalidad de la vista con tipo. En este momento es la única modalidad válida a la que se da soporte.

UNDER *nombre-supervista*
Indica que la vista es una subvista de *nombre-supervista*. La supervista debe ser una vista existente (SQLSTATE 42704) y la vista debe estar definida mediante un tipo estructurado que sea el supertipo inmediato de *nombre-tipo* (SQLSTATE 428DB). El nombre de esquema de *nombre-vista* y *nombre-supervista* debe ser el mismo (SQLSTATE 428DQ). La vista identificada por *nombre-supervista* no debe tener ninguna subvista existente ya definida mediante *nombre-tipo* (SQLSTATE 42742).

Entre las columnas de la vista, se incluye la columna de identificador de objeto de la supervista con su tipo modificado para que sea REF(*nombre-tipo*), seguida de columnas basadas en los atributos de *nombre-tipo* (recuerde que el tipo incluye los atributos de su supertipo).

INHERIT SELECT PRIVILEGES
Cualquier usuario o grupo que sostenga un privilegio SELECT sobre la supervista recibirá un privilegio equivalente sobre la subvista recién creada. Se considera que el definidor de la subvista es el otorgante de este privilegio.

columna-OID
Define la columna de identificador de objeto para la vista con tipo.

REF IS *nombre-columna-OID* **USER GENERATED**
Especifica que en la vista se define una columna de identificador de objeto (OID) como la primera columna. Se necesita un OID para la vista raíz de una jerarquía de vistas (SQLSTATE 428DX). La vista debe ser una vista con tipo (debe estar presente la cláusula OF) que no sea una subvista (SQLSTATE 42613). El nombre de la columna se define como *nombre-columna-OID* y no puede ser el mismo que el nombre de cualquier atributo del tipo estructurado *nombre-tipo* (SQLSTATE 42711). La primera columna especificada en *selección completa* debe ser de tipo REF(*nombre-tipo*) (puede ser necesario que se convierta para que tenga el tipo adecuado). Si no se especifica UNCHECKED, la vista debe basarse en una columna sin posibilidad de nulos, en la que se asegura la unicidad mediante un índice (clave primaria, restricción de unicidad, índice de unicidad o columna OID). Esta columna vendrá referida como la *columna de identificador de objeto* o *columna de OID*. Las palabras clave USER GENERATED indican que el usuario debe proporcionar el valor inicial de la columna de OID cuando inserte una fila. Una vez que se haya insertado una fila, la columna de OID no podrá actualizarse (SQLSTATE 42808).

UNCHECKED
Define la columna identificadora de objeto de la definición de vista con tipo para asumir la unicidad aunque el sistema no pueda probar esta unicidad. Esto es así para utilizarlo con tablas o vistas que se están definiendo en un jerarquía de vistas con tipo donde el usuario sabe que los datos se ajustan a esta norma de unicidad pero no se ajustan a las normas que permiten al sistema probar esta unicidad. La opción UNCHECKED es obligatoria para jerarquías de vistas que comprenden varias jerarquías, o tablas o vistas preexistentes. Cuando se especifica UNCHECKED, el usuario debe asegurarse de que cada fila de la vista tiene un OID exclusivo. Si no se asegura esta propiedad y una vista contiene valores OID duplicados, puede producirse un error en una expresión de vía de acceso o en un operador DEREF donde intervenga uno de los valores OID no exclusivos (SQLSTATE 21000).

opciones-with
Define opciones adicionales que se aplican a las columnas de una vista con tipo.

nombre-columna **WITH OPTIONS**
Especifica el nombre de la columna para la que se especifican las opciones adicionales. El *nombre-columna* debe corresponderse con el nombre de un atributo definido en (no heredado por) el *nombre-tipo* de la vista. La columna debe ser un tipo de referencia (SQLSTATE 42842). No puede corresponderse con una columna que también exista en la supervista (SQLSTATE 428DJ). Sólo puede aparecer un nombre de columna en una cláusula WITH OPTIONS SCOPE de la sentencia (SQLSTATE 42613).

SCOPE
Identifica el ámbito de la columna de tipo de referencia. Debe especificarse un ámbito para cualquier columna que vaya a utilizarse como operando izquierdo de un operador de eliminación de referencia o como argumento de la función DEREF.

La especificación del ámbito de una columna de tipo de referencia puede diferirse a una sentencia ALTER VIEW subsiguiente (si no se hereda el ámbito) para permitir que se defina la vista o tabla de destino, normalmente en el caso de tablas y vistas que se hacen referencia mutuamente. Si no se especifica ningún ámbito para una columna de tipo

de referencia de la vista y la columna de la vista o tabla subyacente tenía ámbito, la columna de tipo de referencia hereda el ámbito de la columna subyacente. La columna permanece sin ámbito si la columna de la vista o tabla subyacente no tenía ámbito. Consulte 473 para obtener más información sobre el ámbito y las columnas de tipo de referencia.

nombre-tabla-tipo
El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

READ ONLY
Identifica la columna como columna de sólo lectura. Esta opción se utiliza para hacer que una columna sea de sólo lectura, de manera que las definiciones de las subvistas puedan especificar una expresión para la misma columna que sea de sólo lectura implícitamente.

AS
Identifica la definición de la vista.

WITH *expresión-tabla-común*
Define una expresión de tabla común para utilizarla con la selección completa que va a continuación. No puede especificarse una expresión de tabla común cuando se define una vista con tipo.

selección completa
Define la vista. En todo momento, la vista consta de las filas que se generarían si se ejecutara la sentencia SELECT. La selección completa no debe hacer referencia a variables del lenguaje principal, a marcadores de parámetros ni a tablas temporales declaradas. En cambio, una vista parametrizada se puede crear como función de tabla SQL.

La selección completa no puede incluir una sentencia de cambio de datos de SQL en la cláusula FROM (SQLSTATE 428FL).

Para subvistas y vistas con tipo: La *selección completa* debe ajustarse a las normas siguientes, de lo contrario se devolverá un error (SQLSTATE 428EA a no ser que se especifique lo contrario).

- La selección completa no debe incluir referencias a las funciones DBPARTITIONNUM o HASHEDVALUE, a las funciones no deterministas o a funciones que se han definido para que exista una acción externa.
- El cuerpo de la vista debe consistir en una sola subselección o en UNION ALL de dos o más subselecciones. Supongamos que cada una de las subselecciones que participan directamente en el cuerpo de la vista se llama una *rama* de la vista. Una vista puede tener una o más ramas.

- La cláusula FROM de cada rama debe consistir de una sola tabla o vista (no necesariamente de tipo), llamada la tabla o vista *subyacente* de esa rama.
- La tabla o vista subyacente de cada rama debe estar en una jerarquía separada (es decir, una vista no puede tener varias ramas con sus tablas o vistas subyacentes en la misma jerarquía).
- Ninguna de las ramas de una definición de vista con tipo puede que especifique GROUP BY o HAVING.
- Si el cuerpo de la vista contiene UNION ALL, entonces la vista raíz de la jerarquía debe especificar la opción UNCHECKED para su columna OID.

Para una jerarquía de vistas y subvistas: BR1 y BR2 son las ramas que aparecen en las definiciones de vistas en la jerarquía. T1 debe ser la tabla subyacente o vista de BR1 y T2 debe ser la tabla o vista subyacente de BR2. Entonces:

- Si T1 y T2 no están en la misma jerarquía, entonces la vista raíz de la jerarquía de vistas debe especificar la opción UNCHECKED para su columna OID.
- Si T1 y T2 están en la misma jerarquía, entonces BR1 y BR2 deben contener predicados o cláusulas ONLY que sean suficientes para garantizar que sus conjuntos de filas no están unidos.

Para subvistas con tipo definidas utilizando EXTEND AS: Para cada rama en el cuerpo de la subvista:

- La tabla subyacente de cada rama debe ser una subtabla (no necesariamente la correspondiente) de alguna tabla subyacente de la supervista inmediata.
- Las expresiones de la lista SELECT deben poder asignarse a las columnas no heredadas de la subvista (SQLSTATE 42854).

Para subvistas con tipo definidas utilizando AS sin EXTEND:

- Para cada rama del cuerpo de la subvista, las expresiones en la lista SELECT deben poder asignarse a los tipos declarados de las columnas heredadas y no heredadas de la subvista (SQLSTATE 42854).
- La expresión OID de cada rama de una jerarquía de la subvista debe ser equivalente (excepto para la conversión del tipo de datos) a la expresión OID en la rama en la misma jerarquía en la vista raíz.
- La expresión para una columna no definida (implícita o explícitamente) como READ ONLY en una supervista debe ser equivalente a todas las ramas en la misma jerarquía subyacente en sus subvistas.

WITH CHECK OPTION

Especifica la restricción según la cual cada fila que se haya insertado o actualizado a través de la vista debe ajustarse a la definición de dicha vista. Una fila que no se ajusta a la definición de la vista es una fila que no cumple las condiciones de búsqueda de la vista.

WITH CHECK OPTION no debe especificarse si se da alguna de las condiciones siguientes:

- La vista es de sólo lectura (SQLSTATE 42813). Si se ha especificado WITH CHECK OPTION para una vista que puede actualizarse y que no admite inserciones, la restricción se aplicará sólo a las actualizaciones.
- La vista hace referencia a la función NODENUMBER o PARTITION, a una función no determinista o a una función que tiene una acción externa (SQLSTATE 42997).
- Un apodo es el destino de actualización de la vista.

- Una vista que tiene definido un activador INSTEAD OF es el destino de actualización de la vista (SQLSTATE 428FQ).

Si se omite WITH CHECK OPTION, la definición de la vista no se utilizará en la comprobación de ninguna operación de inserción o de actualización que utilicen esa vista. Si la vista depende directa o indirectamente de otra vista que incluya WITH CHECK OPTION, durante las operaciones de inserción y actualización es posible que se siga produciendo algún tipo de comprobación. Dado que no se utiliza la definición de la vista, se podrían insertar o actualizar filas a través de la vista que no se ajustasen a la definición de la vista.

CASCADED
La restricción WITH CASCADED CHECK OPTION en una vista *V* significa que *V* hereda las condiciones de búsqueda como restricciones de cualquier vista actualizable de la que *V* depende. Además, todas las vistas actualizables que dependen de *V* también están sometidas a estas restricciones. De esta forma, las condiciones de búsqueda de *V* y todas las vistas de las que *V* depende se unen mediante AND para formar una restricción que se aplica a las inserciones o actualizaciones de *V* o de cualquier vista dependiente de *V*.

LOCAL
La restricción WITH LOCAL CHECK OPTION en una vista *V* significa que la condición de búsqueda de *V* se aplica como restricción a las inserciones o actualizaciones de *V* o de cualquier vista que sea dependiente de *V*.

La diferencia entre CASCADED y LOCAL se explica en el ejemplo siguiente. Tome en consideración las siguientes vistas actualizables (sustituyendo Y en las cabeceras de columna de la tabla que sigue):

```
V1 definida en la tabla T
V2 definida en V1 WITH Y CHECK OPTION
V3 definida en V2
V4 definida en V3 WITH Y CHECK OPTION
V5 definida en V4
```

La tabla siguiente muestra las condiciones de búsqueda con las que se comparan las filas insertadas o actualizadas:

	Y es LOCAL	**Y es CASCADED**
V1 se comprueba con:	ninguna vista	ninguna vista
V2 se comprueba con:	V2	V2, V1
V3 se comprueba con:	V2	V2, V1
V4 se comprueba con:	V2, V4	V4, V3, V2, V1
V5 se comprueba con:	V2, V4	V4, V3, V2, V1

Tome en consideración la siguiente vista actualizable que muestra el efecto de WITH CHECK OPTION utilizando la opción CASCADED por omisión:

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

La siguiente sentencia INSERT que utiliza *V1* será satisfactoria porque *V1* no tiene WITH CHECK OPTION y *V1* no depende de ninguna otra vista que lo tenga.

```
INSERT INTO V1 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V2* dará lugar a error porque *V2* tiene WITH CHECK OPTION y la inserción generaría una fila que no se ajustaría con la definición de *V2*.

```
INSERT INTO V2 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* dará lugar a un error aun no teniendo WITH CHECK OPTION porque *V3* depende de *V2*, que sí tiene especificado WITH CHECK OPTION (SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* será satisfactoria aunque no se ajuste a la definición de *V3* (*V3* no tiene WITH CHECK OPTION); se ajusta a la definición de *V2*, que sí tiene WITH CHECK OPTION.

```
INSERT INTO V3 VALUES(200)
```

WITH NO ROW MOVEMENT o **WITH ROW MOVEMENT**
Especifica la acción a emprender para una vista UNION ALL actualizable cuando se actualiza una fila de modo que viola una restricción de comprobación en la tabla subyacente. El valor por omisión es WITH NO ROW MOVEMENT.

WITH NO ROW MOVEMENT
Especifica que se debe devolver un error (SQLSTATE 23513) si se actualiza una fila de modo que viola una restricción de comprobación sobre la tabla subyacente.

WITH ROW MOVEMENT
Especifica que se debe mover una fila actualizada a la tabla subyacente adecuada, aunque viole una restricción de comprobación en dicha tabla.

El movimiento de filas implica la supresión de las filas que violan la restricción de comprobación y la inserción de dichas filas de nuevo en la vista. La cláusula WITH ROW MOVEMENT sólo se puede especificar para vistas UNION ALL cuyas columnas sean actualizables (SQLSTATE 429BJ). Si se inserta una fila (quizás después de la activación de un activador) en la misma tabla subyacente desde la que se ha suprimido, se devuelve un error (SQLSTATE 23524). Una vista definida utilizando la cláusula WITH ROW MOVEMENT no debe contener operaciones UNION ALL anidadas, excepto en la selección completa más externa (SQLSTATE 429BJ).

Notas:

- ***Compatibilidades:***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - La palabra clave FEDERATED puede especificarse entre las palabras clave CREATE y VIEW. Sin embargo, se ignora la palabra clave FEDERATED porque ya no se devuelve un aviso si se utilizan objetos federados en la definición de la vista.
- La creación de una vista con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Las columnas de la vista heredan el atributo NOT NULL WITH DEFAULT de la tabla o vista base excepto cuando las columnas derivan de una expresión. Al insertar o actualizar una fila en una vista actualizable, se comprueba

comparándola con las restricciones (clave primaria, integridad de referencia y control) si es que hay alguna definida en la tabla base.

- No se puede crear una nueva vista si ésta utiliza en su definición una vista no operativa. (SQLSTATE 51024).
- Esta sentencia no permite utilizar tablas temporales declaradas (SQLSTATE 42995).
- ***Vistas que pueden suprimirse:*** Una vista *podrá suprimirse* si para la vista se ha definido un activador INSTEAD OF para la operación de supresión o si se dan todas las condiciones siguientes:
  - cada cláusula FROM de la selección completa externa identifica una sola tabla base (sin cláusula OUTER), una vista suprimible (sin cláusula OUTER), una expresión de tabla anidada suprimible o una expresión de tabla común suprimible (no puede identificar un apodo)
  - la selección completa externa no incluye una cláusula VALUES
  - la selección completa externa no incluye una cláusula GROUP BY ni una cláusula HAVING
  - la selección completa externa no incluye funciones de columna en la lista de selección
  - la selección completa externa no incluye operaciones SET (UNION, EXCEPT o INTERSECT) a excepción de UNION ALL
  - las tablas base en los operandos de una UNION ALL no deben ser iguales y cada operando debe poderse suprimir
  - la lista de selección de la selección completa externa no incluye DISTINCT
  - la cláusula FROM de la selección completa externa no incluye una *referencia a tabla de cambio de datos*
- ***Vistas que pueden actualizarse:*** Una columna de una vista *podrá actualizarse* si para la vista se ha definido un activador INSTEAD OF para la operación de actualización o si se dan todas las condiciones siguientes:
  - la vista puede suprimirse (con independencia de la existencia de un activador INSTEAD OF para la supresión), la resolución de la columna da como resultado una columna de una tabla base (sin utilizarse una operación de eliminación de referencia) y no se ha especificado la opción READ ONLY
  - todas las columnas correspondientes de los operandos de una UNION ALL tienen tipos de datos que coinciden exactamente (incluyendo longitud o precisión y escala) y valores por omisión que coinciden, si la selección completa de la vista incluye una UNION ALL

  Una vista puede actualizarse si puede actualizarse *cualquier* columna de la vista.
- ***Vistas insertables:***
  - Una vista podrá insertarse si para la vista se ha definido un activador INSTEAD OF para la operación de inserción o si, como mínimo, una columna de la vista puede actualizarse (con independencia de si existe un activador INSTEAD OF para la actualización) y la selección completa de la vista no incluye UNION ALL.
  - Una fila determinada podrá insertarse en una vista (incluyendo UNION ALL) si, y sólo si, cumple las restricciones de comprobación de exactamente una de las tablas base subyacentes.
  - Para su inserción en una vista que incluya columnas que no pueden actualizarse, esas columnas deberán omitirse de la lista de columnas.
- ***Vistas de sólo lectura:*** Una vista es de *sólo lectura* si *no* puede suprimirse, actualizarse o insertarse.

La columna READONLY de la vista de catálogo SYSCAT.VIEWS indica si una vista es de sólo lectura sin tenerse en cuenta los activadores INSTEAD OF.

- Las expresiones de tablas comunes y las expresiones de tablas anidadas siguen el mismo conjunto de normas para determinar si pueden suprimirse, actualizarse, insertarse o si son de sólo lectura.
- ***Vistas no operativas:*** Una *vista no operativa* es una vista que ya no está disponible para las sentencias de SQL. Una vista deja de ser operativa si:
  - Se revoca un privilegio del que depende la definición de dicha vista.
  - Se elimina un objeto, como una tabla, un apodo, un alias o una función, del que depende la definición de dicha vista.
  - Otra vista, de la cual depende la definición de la vista, deja de ser operativa.
  - Una vista que es la supervista de la definición de vista (la subvista) se vuelve no operativa.

  En otras palabras, una vista no operativa es aquélla en la que se ha eliminado involuntariamente la definición de vista. Por ejemplo, al eliminar un alias, cualquier vista definida con ese alias deja de ser operativa. También se considerarán no operativas todas las vistas dependientes, y los paquetes que dependen de la misma ya no se consideran válidos.

  Hasta que no se vuelva a crear o eliminar explícitamente una vista no operativa, no podrán compilarse las sentencias que utilicen esa vista (SQLSTATE 51024), exceptuando las sentencias CREATE ALIAS, CREATE VIEW, DROP VIEW y COMMENT ON TABLE. Hasta que no se haya eliminado explícitamente la vista no operativa, no se puede utilizar su nombre calificado para crear otra tabla o alias (SQLSTATE 42710).

  Una vista no operativa puede volver a crearse emitiendo una sentencia CREATE VIEW mediante el texto de definición de la vista no operativa. Este texto de definición de vista se almacena en la columna TEXT del catálogo SYSCAT.VIEWS. Cuando se vuelve a crear una vista no operativa, es necesario otorgar de forma explícita todos los privilegios que otros necesitan en dicha vista, debido al hecho de que se suprimen todos los registros de autorizaciones en una vista si la vista se marca como no operativa. Observe que no es necesario eliminar de manera explícita la vista no operativa para volverla a crear. La emisión de una sentencia CREATE VIEW que utilice el mismo *nombre-vista* que una vista no operativa hará que se sustituya la vista no operativa, y la sentencia CREATE VIEW emitirá un aviso (SQLSTATE 01595).

  Las vistas no operativas se indican mediante una X en la columna VALID de la vista de catálogo SYSCAT.VIEWS y una X en la columna STATUS de la vista de catálogo SYSCAT.TABLES.

- ***Privilegios:***

  El definidor de una vista siempre recibe el privilegio SELECT para la vista, así como el derecho a eliminar la vista. El definidor de una vista obtendrá el privilegio CONTROL para la vista sólo si tiene el privilegio CONTROL para todas las tablas base, las vistas o los apodos identificados en la selección completa o si tiene autorización SYSADM o DBADM.

  El definidor de la vista recibe los privilegios INSERT, UPDATE, UPDATE a nivel de columna o DELETE para la vista si ésta no es de sólo lectura y el definidor tiene los privilegios correspondientes para los objetos subyacentes.

  Para una vista definida WITH ROW MOVEMENT, el definidor adquiere el privilegio UPDATE sobre la vista sólo si el definidos tiene el privilegio UPDATE sobre todas las columnas de la vista, así como los privilegios INSERT y DELETE sobre todas las tablas o vistas subyacentes.

El definidor de una vista sólo adquiere privilegios si los privilegios de los cuales aquéllos derivan ya existen cuando se crea la vista. El definidor debe tener estos privilegios directamente o porque PUBLIC tiene este privilegio. Los privilegios no se tienen en cuenta al definir una vista en un apodo de servidor federado. Sin embargo, al utilizar una vista en un apodo, el ID de autorización de usuario debe tener los privilegios de selección válidos en la tabla o vista a la que el apodo hace referencia en la fuente de datos. De lo contrario, se emite un error. No se tienen en cuenta los privilegios pertenecientes a grupos de los que forma parte el definidor.

Cuando se crea una subvista, se otorgan automáticamente sobre la misma los privilegios SELECT sostenidos sobre la supervista inmediata.

- ***Columnas REF y de ámbito:***

  Cuando seleccione una columna de tipo de referencia en la selección completa de una definición de vista, tenga en cuenta el tipo de destino y el ámbito que sean necesarios.

  – Si el tipo de destino y el ámbito necesarios son los mismos que los de la vista o tabla subyacente, ya puede seleccionar la columna.
  – Si se debe cambiar el ámbito, utilice la cláusula WITH OPTIONS SCOPE para definir la vista o tabla con ámbito necesarias.
  – Si se debe cambiar el tipo de destino de la referencia, se debe convertir la columna primero en el tipo de representación de la referencia y después en el tipo de referencia nuevo. En este caso, el ámbito se puede especificar en la conversión hacia el tipo de referencia o mediante la cláusula WITH OPTIONS SCOPE. Por ejemplo, suponga que selecciona la columna Y definida como REF(TYP1) SCOPE TAB1. Desea que se defina como REF(VTYP1) SCOPE VIEW1. El elemento de la lista de selección sería el siguiente:

    ```
    CAST(CAST(Y AS VARCHAR(16) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
    ```

- ***Columnas de identidad:*** Se considera que una columna de una vista es una columna de identidad si el elemento de la columna correspondiente en la selección completa de la definición de vista es el nombre de una columna de identidad de una tabla, o el nombre de una columna de una vista que, directa o indirectamente, se correlaciona con el nombre de una columna de identidad de una tabla base.

  En todos los demás casos, las columnas de una vista no obtendrán el atributo de identidad. Por ejemplo:

  – La lista de selección de la definición de vista contiene varias instancias del nombre de una columna de identidad (es decir, se selecciona la misma columna más de una vez)
  – la definición de la vista incluye una operación de unión
  – una columna de la definición de la vista incluye una expresión que hace referencia a una columna de identidad
  – la definición de la vista incluye una operación UNION

  Cuando se inserta en una vista para la que la lista de selección de la definición de vista incluye, directa o indirectamente, el nombre de una columna de identidad de una tabla base, son aplicables las mismas normas que si la sentencia INSERT hiciera referencia directamente a la columna de identidad de la tabla base.

- ***Vistas federadas:*** Una vista federada es una vista que incluye una referencia a un apodo situado en algún lugar de la selección completa. La presencia de este apodo cambia el modelo de autorización utilizado para la vista cuando posteriormente se hace referencia a la misma en una consulta.

Cuando se crea la vista, no se realiza ninguna comprobación de privilegios para determinar si el definidor de la vista tiene acceso a la tabla de fuente de datos subyacente o a la vista de un apodo. El control de privilegios para referencias a tablas o vistas de la base de datos federada se lleva a cabo como siempre, siendo necesario que el definidor de la vista tenga al menos el privilegio SELECT sobre esos objetos.

Cuando posteriormente se hace referencia en una consulta, a una vista federada, los apodos que resultan de las consultas en la fuente de datos y el ID de autorización que ha emitido la consulta (o el ID de autorización remota con el que se correlaciona) deben tener los privilegios necesarios para acceder a la tabla o vista de la fuente de datos. El ID de autorización que emite la consulta que hace referencia a la vista federada no es obligatorio que tenga ningún privilegio adicional en vista o tablas (no federadas) que existen en el servidor federado.

- ***ROW MOVEMENT, activadores y restricciones:*** Cuando se actualiza una vista definida mediante la cláusula WITH ROW MOVEMENT, la secuencia de operaciones de activador y restricciones es la siguiente:
  1. Los activadores BEFORE UPDATE se activan para todas las filas que se van a actualizar, incluidas las filas que se van a mover.
  2. La operación de actualización se procesa.
  3. Las restricciones se procesan para todas las filas actualizadas.
  4. Loa activadores AFTER UPDATE (tanto de nivel de fila como de nivel de sentencia) se activan en el orden de creación, para todas las filas que satisfacen las restricciones tras la operación de actualización. Debido a que es una sentencia UPDATE, todos los activadores de nivel de sentencia UPDATE se activan para todas las tablas subyacentes.
  5. Los activadores BEFORE DELETE se activan para todas las filas que no satisfacían las restricciones tras la operación de actualización (son las filas que se van a mover).
  6. La operación de supresión se procesa.
  7. Se procesan las restricciones correspondientes a todas las filas suprimidas.
  8. Los activadores AFTER DELETE (tanto de nivel de fila como de nivel de sentencia) se activan en orden de creación, para todas las filas suprimidas. Los activadores de nivel de sentencia se activan sólo para las tablas que intervienen en la operación de supresión.
  9. Los activadores BEFORE INSERT se activan para todas las filas que se van a insertar (es decir, las filas que se van a mover). Las nuevas tablas de transición correspondientes a activadores BEFORE INSERT contienen los datos de entrada proporcionados por el usuario.
  10. La operación de inserción se procesa.
  11. Las restricciones se procesan para todas las filas insertadas.
  12. Los activadores AFTER INSERT (tanto de nivel de fila como de nivel de sentencia) se activan en el orden de creación para todas las filas insertadas. Los activadores de nivel de sentencia se activan sólo para las tablas que intervienen en la operación de inserción.
- ***Vistas UNION ALL anidadas:*** Una vista definida con UNION ALL y basada, directa o indirectamente, en una vista que también está definida con UNION ALL no se puede actualizar si alguna de las vistas se ha definido utilizando la cláusula WITH ROW MOVEMENT (SQLSTATE 429BK).

Ejemplos:

Ejemplo 1: Cree una vista denominada MA_PROJ en la tabla PROJECT que sólo contenga las filas con un número de proyecto (PROJNO) que empiece por las letras 'MA'.

```
CREATE VIEW MA_PROJ  AS SELECT *
  FROM PROJECT
   WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Ejemplo 2: Cree una vista como la del ejemplo 1, pero seleccione sólo las columnas para el número de proyecto (PROJNO), nombre de proyecto (PROJNAME) y empleado encargado del proyecto (RESPEMP).

```
CREATE VIEW MA_PROJ
  AS SELECTPROJNO, PROJNAME, RESPEMP
  FROM PROJECT
  WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Ejemplo 3: Cree una vista como la del ejemplo 2, pero en la vista, llame a la columna para el empleado encargado del proyecto IN_CHARGE.

```
CREATE VIEW MA_PROJ
  (PROJNO, PROJNAME, IN_CHARGE)
  AS SELECTPROJNO, PROJNAME, RESPEMP
  FROM PROJECT
  WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Nota: Aunque sólo se cambie uno de los nombres de columna, los nombres de las tres columnas de la vista deben listarse entre los paréntesis que siguen a MA_PROJ.

Ejemplo 4: Cree una vista llamada PRJ_LEADER que contenga las cuatro primeras columnas (PROJNO, PROJNAME, DEPTNO, RESPEMP) de la tabla PROJECT junto con el apellido (LASTNAME) de la persona que es responsable del proyecto (RESPEMP). Obtendremos el nombre de la tabla EMPLOYEE emparejando EMPNO de EMPLOYEE con RESPEMP de PROYECT.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
  FROM PROJECT, EMPLOYEE
  WHERE RESPEMP = EMPNO
```

Ejemplo 5: Cree una vista como en el ejemplo 4, pero que además de mostrar las columnas PROJNO, PROJNAME, DEPTNO, RESPEMP y LASTNAME, que también muestre la paga total (SALARY + BONUS + COMM) del empleado responsable. Asimismo, seleccione sólo aquellos proyectos cuyo empleo de personal principal (PRSTAFF) sea mayor que 1.

```
CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
  FROM PROJECT, EMPLOYEE
  WHERE RESPEMP = EMPNO
  AND PRSTAFF > 1
```

Puede evitarse la especificación de la lista de nombres de columna especificando el nombre de la expresión SALARY+BONUS+COMM como TOTAL_PAY en la selección completa.

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
              LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
    FROM PROJECT, EMPLOYEE
    WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

Ejemplo 6: Dado el conjunto de tablas y vistas mostrado en el diagrama siguiente:

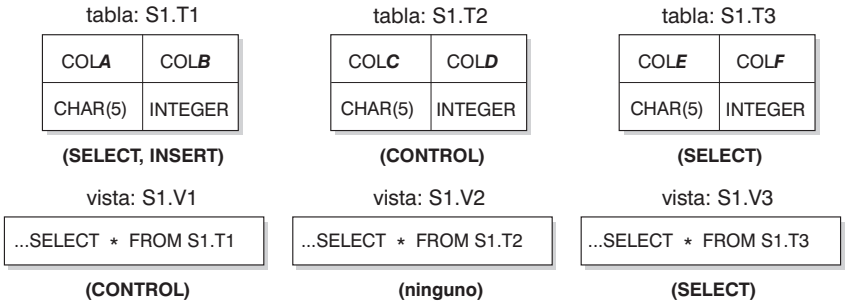


Figura 1. Tablas y vistas para el ejemplo 6

Se han otorgado al usuario ZORPIE (que no tiene la autorización DBADM ni SYSADM) los privilegios que se muestran entre corchetes debajo de cada objeto:

1. ZORPIE obtendrá el privilegio CONTROL en la vista que cree con:

   ```
   CREATE VIEW VA AS SELECT * FROM S1.V1
   ```

   porque tiene CONTROL en S1.V1. (Algún usuario que disponga de autorización DBADM o SYSADM deberá otorgar a ZORPIE el privilegio CONTROL para S1.V1.) No importa qué privilegios tiene, si tiene alguno, para la tabla base subyacente.
2. ZORPIE no estará autorizada a crear la vista:

   ```
   CREATE VIEW VB AS SELECT * FROM S1.V2
   ```

   como ZORPIE no tiene ni CONTROL ni SELECT en S1.V2. No importa el hecho de que tenga CONTROL en la tabla base principal (S1.T2).
3. ZORPIE obtendrá el privilegio CONTROL en la vista que cree con:

   ```
   CREATE VIEW VC (COLA, COLB, COLC, COLD)
     AS SELECT * FROM S1.V1, S1.T2
     WHERE COLA = COLC
   ```

   porque la selección completa de ZORPIE.VC hace referencia a la vista S1.V1 y a la tabla S1.T2 y dispone de CONTROL en ambas. Observe que la vista VC es de sólo lectura, por lo tanto ZORPIE no obtiene los privilegios INSERT, UPDATE ni DELETE.
4. ZORPIE obtendrá el privilegio SELECT en la vista que cree con:

   ```
   CREATE VIEW VD (COLA,COLB, COLE, COLF)
     AS SELECT * FROM S1.V1, S1.V3
     WHERE COLA = COLE
   ```

   porque la selección completa de ZORPIE.VD hace referencia a las dos vistas S1.V1 y S1.V3, en una de las cuales sólo tiene el privilegio SELECT, y en la otra tiene el privilegio CONTROL. Se le otorga el menor de los dos privilegios, SELECT, en ZORPIE.VD.
5. ZORPIE obtendrá el privilegio INSERT, UPDATE y DELETE con GRANT OPTION y el privilegio SELECT en la vista VE en la siguiente definición de vista.

   ```
   CREATE VIEW VE
      AS SELECT * FROM S1.V1
     WHERE COLA > ANY
            (SELECT COLE FROM S1.V3)
   ```

Los privilegios de ZORPIE en VE se determinan principalmente de acuerdo con sus privilegios en S1.V1. Como sólo se hace referencia a S1.V3 en una subconsulta, sólo necesita el privilegio SELECT en S1.V3 para crear la vista VE. La persona que define la vista sólo obtiene CONTROL en la vista si tiene CONTROL en todos los objetos a los que se hace referencia en la definición de vista. ZORPIE no tiene CONTROL en S1.V3, en consecuencia no obtiene CONTROL en VE.

Información relacionada:

- "CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 259
- "Consultas de SQL" en la publicación *Consulta de SQL, Volumen 1*

CREATE WRAPPER

La sentencia CREATE WRAPPER registra un reiniciador en un servidor federado. Un reiniciador es un mecanismo mediante el cual un servidor federado puede interactuar con determinados tipos de fuentes de datos.

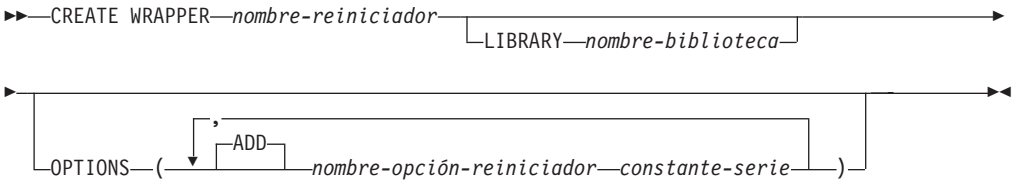
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

nombre-reiniciador
Nombra el reiniciador. Puede ser:

- Un nombre predefinido. Si se especifica un nombre predefinido, el servidor federado asigna automáticamente un valor por omisión a *nombre-biblioteca*.
- Un nombre suministrado por el usuario. Si se proporciona un nombre suministrado por el usuario, también se debe especificar el *nombre-biblioteca* adecuado que se debe utilizar con ese reiniciador y sistema operativo.

LIBRARY *nombre-biblioteca*
Nombra el archivo que contiene el módulo de biblioteca del reiniciador.

El nombre de biblioteca puede especificarse como un nombre de vía de acceso absoluta o, simplemente, puede especificarse el nombre base (sin la vía de acceso). Si sólo se especifica el nombre base, la biblioteca debe residir en el subdirectorio lib (UNIX) o en el subdirectorio bin (Windows) de la vía de acceso de instalación de DB2.El *nombre-biblioteca* se debe especificar entre comillas simples.

La opción LIBRARY sólo es necesaria si se utiliza un *nombre-reiniciador* proporcionado por el usuario. Esta opción no debe utilizarse cuando se proporciona un *nombre-reiniciador* predefinido.

OPTIONS (ADD *nombre-opción-reiniciador constante-serie*, **...)**
Las opciones de reiniciador se utilizan para configurar el reiniciador o para definir cómo DB2 utiliza el reiniciador. El *nombre-opción-reiniciador* es el nombre de la opción. La *constante-serie* especifica el valor para la opción de reiniciador. La *constante-serie* se debe especificar entre comillas simples. Algunas opciones de reiniciador pueden utilizarlas todos los reiniciadores y otras son específicas de un reiniciador en particular.

Ejemplos:

Ejemplo 1: Registre el reiniciador NET8 en un servidor federado para acceder a fuentes de datos Oracle. *NET8* es el nombre predefinido para uno de los dos reiniciadores que puede utilizar para acceder a las fuentes de datos Oracle.

```
CREATE WRAPPER NET8
```

Ejemplo 2: Registre un reiniciador en un servidor federado DB2 que utiliza el sistema operativo Linux para acceder a fuentes de datos ODBC. Asigne el nombre *odbc* al reiniciador que se registra en la base de datos federada. La vía de acceso completa de la biblioteca que contiene el Gestor de controladores ODBC se define en la opción de reiniciador *MODULE '/usr/lib/odbc.so'*.

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

Ejemplo 3: Registre un reiniciador en un servidor federado DB2 que utiliza el sistema operativo Windows para acceder a fuentes de datos ODBC. El nombre de biblioteca para el reiniciador de ODBC es *'db2rcodbc.dll'*.

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

Ejemplo 4: Registre un reiniciador en un servidor federado DB2 que utiliza el sistema operativo AIX para acceder a fuentes de datos Entrez. Designe *entrez_wrapper* como el nombre del reiniciador. En los servidores federados AIX, *libdb2lsentrez.a* es el archivo de biblioteca para el reiniciador Entrez. La opción EMAIL se necesita cuando el reiniciador Entrez está registrado en el servidor federado.

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a'
  OPTIONS (EMAIL 'jeff@someplace.com')
```

DECLARE CURSOR

La sentencia DECLARE define un cursor.

Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización:

El término "sentencia SELECT del cursor" se utiliza para especificar las normas de autorización. La sentencia SELECT del cursor es una de las siguientes:

- La sentencia-select preparada que se identifica por el *nombre-sentencia*
- La *sentencia-select* especificada

Por cada tabla o vista que esté identificada (ya sea directamente o mediante un alias) en la sentencia SELECT del cursor, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Autorización SYSADM o DBADM.
- Para cada tabla o vista identificada en la *sentencia-select*:
  - Privilegio SELECT para la tabla o vista, o bien
  - Privilegio CONTROL para la tabla o vista.

Si la *sentencia-select* contiene una sentencia de cambio de datos de SQL, los requisitos de autorización de la sentencia también se aplican a la sentencia DECLARE CURSOR.

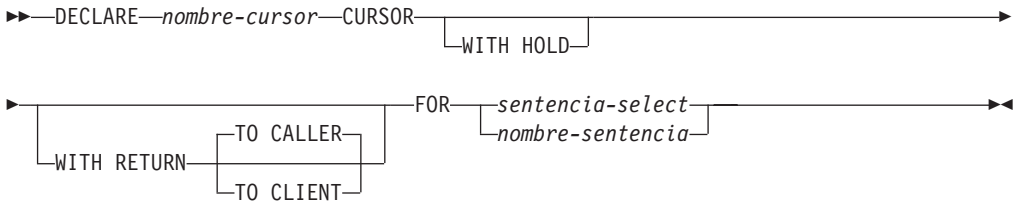
Si se especifica *nombre-sentencia*:

- El ID de autorización de la sentencia es el ID de autorización de ejecución.
- El control de autorizaciones se realiza al preparar la sentencia-select.
- El cursor no puede abrirse si la sentencia-select no está preparada correctamente.

Si se especifica *sentencia-select*:

- No se comprueban los privilegios GROUP.
- El ID de autorización de la sentencia es el ID de autorización especificado durante la preparación del programa.

Sintaxis:



Descripción:

nombre-cursor
Especifica el nombre del cursor que se ha creado al ejecutar el programa fuente. Este nombre no debe ser el mismo que el de ningún otro cursor que esté declarado en el programa fuente. El cursor deberá abrirse para poder utilizarlo.

WITH HOLD
Mantiene recursos en varias unidades de trabajo. El efecto del atributo del cursor WITH HOLD es el siguiente:

- En las unidades de trabajo que finalizan con COMMIT:
  - Los cursores abiertos definidos con WITH HOLD permanecen abiertos. El cursor se sitúa antes de la siguiente fila lógica de la tabla de resultados.

    Si se emite la sentencia DISCONNECT después de la sentencia COMMIT para una conexión con cursores WITH HOLD, los cursores mantenidos deben cerrarse explícitamente, porque si no se supondrá que la conexión ha realizado cierto trabajo (simplemente por tener abiertos cursores WITH HOLD aun sin haber emitido sentencias de SQL) y fallará la sentencia DISCONNECT.
  - Se liberan todos los bloqueos, excepto los bloqueos que protegen la posición actual del cursor, de los cursores WITH HOLD abiertos. Los bloqueos mantenidos incluyen los bloqueos sobre la tabla y para los entornos paralelos, los bloqueos sobre las filas en las que los cursores están situados actualmente. Los bloqueos sobre paquetes y secciones SQL dinámico (si las hay) se mantienen.
  - Las operaciones válidas en los cursores definidos WITH HOLD que están inmediatamente a continuación de una petición COMMIT son:
    - FETCH: Lee la siguiente fila del cursor.
    - CLOSE: Cierra el cursor.
  - UPDATE y DELETE CURRENT OF CURSOR sólo son válidas para aquellas filas que se recuperan dentro de la misma unidad de trabajo.
  - Se liberan los localizadores de LOB.
  - El conjunto de filas modificado por:
    - Una sentencia de cambio de datos
    - Rutinas que modifican datos SQL incorporados dentro de cursores WITH HOLD abiertos

    se confirma.
- Para las unidades de trabajo que finalizan con ROLLBACK:
  - Se cierran todos los cursores abiertos.
  - Se liberan todos los bloqueos adquiridos durante la unidad de trabajo.
  - Se liberan los localizadores de LOB.
- Para el caso especial de COMMIT:
  - Los paquetes pueden volver a crearse ya sea explícitamente, enlazándolos, o implícitamente, debido a que fueron invalidados y luego vueltos a crear dinámicamente la primera vez que se ha hecho referencia a los mismos. Todos los cursores mantenidos se cierran cuando se vuelve a enlazar el paquete. Ello puede provocar errores en una posterior ejecución.

WITH RETURN
Esta cláusula indica que el cursor está pensado para ser utilizado como conjunto de resultados de un procedimiento almacenado. WITH RETURN sólo es aplicable si el código fuente de un procedimiento almacenado contiene la

sentencia DECLARE CURSOR. En los demás casos, el precompilador puede aceptar la cláusula, pero no tiene ningún efecto.

Dentro de un procedimiento SQL, los cursores declarados mediante la cláusula WITH RETURN que todavía están abiertos cuando finaliza el procedimiento SQL, definen los conjuntos de resultados del procedimiento SQL. Todos los demás cursores abiertos de un procedimiento SQL se cierran cuando finaliza el procedimiento SQL. Dentro de un procedimiento almacenado externo (un procedimiento no definido utilizando LANGUAGE SQL), el valor por omisión para todos los cursores es WITH RETURN TO CALLER. Por lo tanto, todos los cursores que estén abiertos cuando el procedimiento finalice se considerarán conjuntos de resultados.

TO CALLER
Especifica que el cursor puede devolver un conjunto de resultados al llamador. Por ejemplo, si el llamador es otro procedimiento almacenado, el conjunto de resultados se devuelve a ese procedimiento almacenado. Si el llamador es una aplicación cliente, el conjunto de resultados se devuelve a la aplicación cliente.

TO CLIENT
Especifica que el cursor puede devolver un conjunto de resultados a la aplicación cliente. Este cursor es invisible para cualquier procedimiento anidado intermedio. Si una función o un método ha llamado al procedimiento directa o indirectamente, no podrán devolverse conjuntos de resultados al cliente y el cursor se cerrará tras completarse el procedimiento.

sentencia-select
Identifica la sentencia SELECT del cursor. La *sentencia-select* no debe incluir marcadores de parámetro, pero puede incluir referencias a las variables del lenguaje principal. Las declaraciones de variables del lenguaje principal deben preceder a la sentencia DECLARE CURSOR en el programa fuente.

nombre-sentencia
La sentencia SELECT del cursor es la sentencia SELECT preparada que se indica por el *nombre-sentencia* cuando está abierto el cursor. El *nombre-sentencia* no debe ser igual a ningún otro *nombre-sentencia* que esté especificado en otra sentencia DECLARE CURSOR del programa fuente.

Para obtener información acerca de las sentencias SELECT preparadas, consulte "PREPARE".

Notas:

- Un programa invocado desde otro programa o desde otro archivo fuente dentro del mismo programa no puede utilizar el cursor que fue abierto por el programa llamador.
- Los procedimientos almacenados no anidados, no definidos con LANGUAGE SQL, utilizan por omisión la opción WITH RETURN TO CALLER si DECLARE CURSOR está especificado sin una cláusula WITH RETURN, y el cursor se deja abierto en el procedimiento. Esto proporciona compatibilidad con procedimientos almacenados pertenecientes a versiones anteriores, en las cuales los procedimientos almacenados pueden devolver conjuntos de resultados a las aplicaciones cliente correspondientes. Para evitar este comportamiento, cierre todos los cursores abiertos en el procedimiento.
- Si la sentencia SELECT de un cursor contiene CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP, todas las referencias a estos registros especiales

producirán el mismo valor ″datatime″ (de fecha y hora) respectivo en cada FETCH. Este valor se determina al abrir el cursor.

- Para lograr un proceso de datos más eficaz, el gestor de bases de datos puede agrupar los datos en bloques para cursores de sólo lectura cuando se recuperan datos de un servidor remoto. Mediante la cláusula FOR UPDATE, el gestor de bases de datos puede decidir si un cursor es actualizable o no. La posibilidad de actualización también se utiliza para determinar la selección de la vía de acceso. Si un cursor no se va a utilizar en una sentencia UPDATE con posición o DELETE, debería declararse como FOR READ ONLY.
- Un cursor en estado abierto designa una tabla de resultado y una posición relativa para las filas de dicha tabla. La tabla es la tabla de resultados especificada por la sentencia SELECT del cursor.
- Un cursor es *suprimible* si se cumplen todas las condiciones siguientes:
  - cada cláusula FROM de la selección completa externa identifica una sola tabla base o vista suprimible (no puede identificar una expresión de tabla común, una expresión de tabla anidada ni un apodo) sin utilizar la cláusula OUTER
  - la selección completa externa no incluye una cláusula VALUES
  - la selección completa externa no incluye una cláusula GROUP BY ni una cláusula HAVING
  - la selección completa externa no incluye funciones de columna en la lista de selección
  - la selección completa externa no incluye operaciones SET (UNION, EXCEPT o INTERSECT) a excepción de UNION ALL
  - la lista de selección de la selección completa externa no incluye DISTINCT
  - la selección completa externa no incluye una cláusula ORDER BY (incluso si la cláusula ORDER BY está anidada en una vista) y no se ha especificado la cláusula FOR UPDATE
  - la sentencia-select no incluye una cláusula FOR READ ONLY
  - la cláusula FROM de la selección completa externa no incluye una *referencia a tabla de cambio de datos*
  - se dan una o más de las condiciones siguientes:
    - se ha especificado la cláusula FOR UPDATE
    - el cursor está definido de forma estática, a no ser que la opción de vinculación STATICREADONLY sea YES
    - la opción de enlace LANGLEVEL es MIA o SQL92E

  Una columna de la lista de selección de la selección completa externa asociada con un cursor es *actualizable* si se cumplen todas las condiciones siguientes:
  - el cursor es suprimible
  - la resolución de la columna devuelve una columna de la tabla base
  - la opción de enlace LANGLEVEL es MIA o SQL92E, o la sentencia-select incluye la cláusula FOR UPDATE (la columna debe estar especificada explícita o implícitamente en la cláusula FOR UPDATE)

  Un cursor es de *sólo-lectura* si no es suprimible.

  Un cursor es *ambiguo* si se cumplen todas las condiciones siguientes:
  - la sentencia-select se prepara de forma dinámica
  - la sentencia-select no incluye la cláusula FOR READ ONLY ni la cláusula FOR UPDATE
  - la opción de enlace LANGLEVEL es SAA1
  - por lo demás, el cursor cumple los requisitos de un cursor suprimible

Un cursor ambiguo se considera de sólo lectura si la opción de enlace BLOCKING es ALL, de lo contrario se considera actualizable.

- Los cursores de procedimientos almacenados que son invocados por programas de aplicación que se han escrito utilizando CLI pueden utilizarse para definir conjuntos de resultados que se devuelven directamente a la aplicación cliente. Los cursores de procedimientos SQL también pueden devolver resultados a un procedimiento SQL llamador sólo si están definidos utilizando la cláusula WITH RETURN.
- Los cursores que se han declarado en rutinas que se invocan directa o indirectamente desde un cursor declarado como WITH HOLD, no heredan la opción WITH HOLD. De esta forma, a menos que el cursor de la rutina se haya definido explícitamente como WITH HOLD, una ejecución de COMMIT en la aplicación hará que se cierre.

  Consideremos la aplicación y los dos UDF siguientes:

```
Aplicación:

   DECLARE APPCUR CURSOR WITH HOLD FOR SELECT UDF1() ...
   OPEN APPCUR
   FETCH APPCUR ...
   COMMIT

UDF1:

   DECLARE UDF1CUR CURSOR FOR SELECT UDF2() ...
   OPEN UDF1CUR
   FETCH UDF1CUR ...

UDF2:

   DECLARE UDF2CUR CURSOR WITH HOLD FOR SELECT UDF2() ...
   OPEN UDF2CUR
   FETCH UDF2CUR ...
```

  Después de que la aplicación haya buscado el cursor APPCUR, se abrirán los tres cursores. Cuando la aplicación emite la sentencia COMMIT, APPCUR sigue abierto, ya que se ha declarado como WITH HOLD. Sin embargo, en UDF1, el cursor UDF1CUR está cerrado, ya que no se ha definido con la opción WITH HOLD. Cuando se cierra el cursor UDF1CUR, se completan todas las invocaciones de rutina de la sentencia-select correspondiente (recibiendo una llamada final, si así se ha definido). UDF2 se completa, lo que da lugar a que UDF2CUR se cierre.

Ejemplos:

Ejemplo 1: La sentencia DECLARE CURSOR asocia el nombre del cursor C1 con los resultados de la sentencia SELECT.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPARTMENT
  WHERE ADMRDEPT = 'A00';
```

Ejemplo 2: Supongamos que la tabla EMPLOYEE se ha modificado para añadir una columna generada, WEEKLYPAY, que calcula la paga semanal basada en el salario anual. Declare un cursor para recuperar el valor de la columna generada por el sistema de una fila que se va a insertar.

```
EXEC SQL DECLARE C2 CURSOR FOR
  SELECT E.WEEKLYPAY
  FROM NEW TABLE
```

```
(INSERT INTO EMPLOYEE
 (EMPNO, FIRSTNME, MIDINIT, LASTNAME, EDLEVEL, SALARY)
 VALUES('000420', 'Peter', 'U', 'Bender', 16, 31842) AS E;
```

Información relacionada:

- "Sentencia select" en la publicación *Consulta de SQL, Volumen 1*
- "CALL" en la página 107
- "OPEN" en la página 629
- "PREPARE" en la página 634

Ejemplos relacionados:

- "cursor.sqb -- How to update table data with cursor statically (MF COBOL)"
- "tabsql.sqb -- Demonstrates common table expressions using SQL (MF COBOL)"
- "fnuse.sqc -- How to use built-in SQL functions (C)"
- "tbinfo.sqc -- How to get information at the table level (C)"
- "tut_mod.sqc -- How to modify table data (C)"
- "tut_read.sqc -- How to read tables (C)"
- "fnuse.sqC -- How to use built-in SQL functions (C++)"
- "tbinfo.sqC -- How to get information at the table level (C++)"
- "tut_mod.sqC -- How to modify table data (C++)"
- "tut_read.sqC -- How to read tables (C++)"

DECLARE GLOBAL TEMPORARY TABLE

La sentencia DECLARE GLOBAL TEMPORARY TABLE define una tabla temporal para la sesión actual. La descripción de la tabla temporal declarada no aparece en el catálogo del sistema. No es una tabla permanente y no se puede compartir con otras sesiones. Cada sesión que define una tabla temporal global declarada del mismo nombre tiene su propia descripción exclusiva de la tabla temporal. Cuando la sesión finaliza, se suprimen las filas de la tabla y se elimina la descripción de la tabla temporal.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

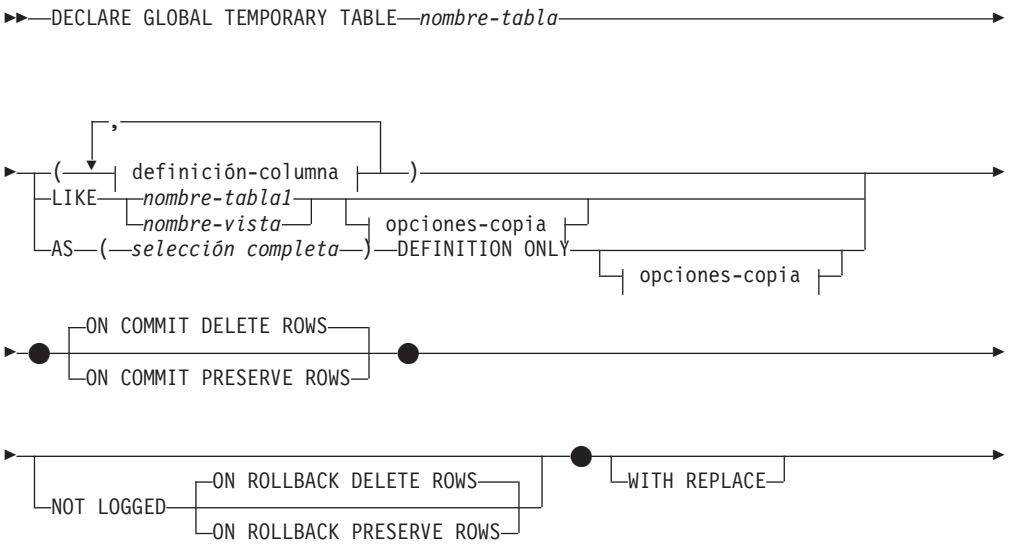
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

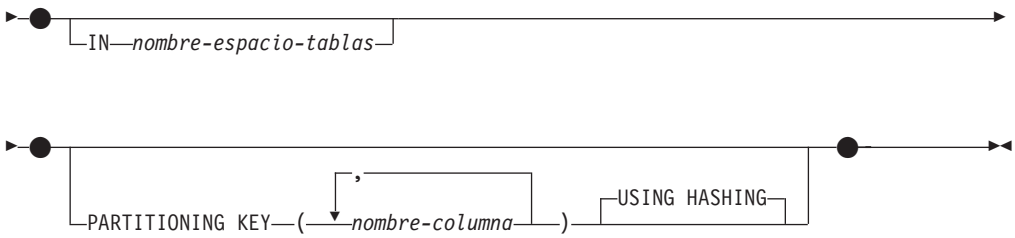
- Autorización SYSADM o DBADM
- Privilegio USE para el espacio de tablas USER TEMPORARY.

Cuando se define una tabla utilizando LIKE o una selección completa, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes para cada tabla o vista identificada:

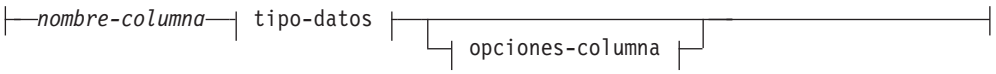
- Privilegio SELECT para la tabla o vista
- Privilegio CONTROL para la tabla o vista
- Autorización SYSADM o DBADM

Sintaxis:

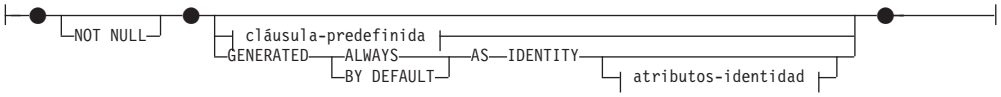




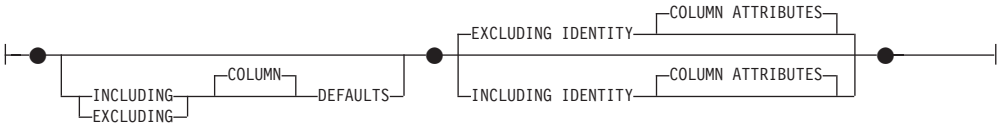
definición-columna:



opciones-columna:



opciones-copia:



Descripción:

nombre-tabla
Designa la tabla temporal. Si se especifica explícitamente, el calificador debe ser SESSION, de lo contrario se produce un error (SQLSTATE 428EK). Si no se especifica el calificador, se asigna SESSION de forma implícita.

Cada sesión que define una tabla temporal global declarada con el mismo *nombre-tabla* tiene su propia descripción exclusiva de esa tabla temporal. La cláusula WITH REPLACE se debe especificar si *nombre-tabla* identifica una tabla temporal declarada que ya existe en la sesión (SQLSTATE 42710).

Es posible que el catálogo ya contenga una tabla, vista, alias o apodo que tenga el mismo nombre y el nombre de esquema SESSION. En este caso:

- Es posible todavía definir una tabla temporal global declarada *nombre-tabla* sin provocar un error ni un aviso
- Las referencias a SESSION.*nombre-tabla* se resolverán en una tabla temporal global declarada, en lugar de en la tabla SESSION.*nombre-tabla* ya definida en el catálogo.

definición-columna
Define los atributos de una columna de la tabla temporal.

nombre-columna
Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no deben superar el valor 4 005.
- Un tamaño de página de 8K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 8 101.
- Un tamaño de página de 16K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 16 293.
- Un tamaño de página de 32K con un máximo de 1 012 columnas, donde el número total de bytes de las columnas no debe superar el valor 32 677.

Para obtener más detalles, consulte el apartado "Tamaño de fila" en "CREATE TABLE".

tipo-datos
Para conocer los tipos que están permitidos, consulte *tipo-datos* en "CREATE TABLE". Los tipos BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, DATALINK, los tipos de referencia y los tipos estructurados no se pueden utilizar en tablas temporales globales declaradas (SQLSTATE 42962). Esta restricción es aplicable también a los tipos diferenciados con origen en esos tipos no permitidos.

Se puede especificar FOR BIT DATA como parte de los tipos de datos de serie de caracteres.

opciones-columna
Define otras opciones relacionadas con las columnas de la tabla.

NOT NULL
Evita que la columna contenga valores nulos. Para obtener información acerca de la especificación de valores nulos, consulte NOT NULL en "CREATE TABLE".

cláusula-predefinida
Para obtener información acerca de la especificación de valores por omisión, consulte *cláusula-por-omisión* en "CREATE TABLE".

IDENTITY y atributos-identidad
Para obtener información acerca de la especificación de columnas de identidad, consulte IDENTITY y *atributos-identidad* en "CREATE TABLE".

LIKE *nombre-tabla1* o *nombre-vista*
Especifica que las columnas de la tabla tienen exactamente el mismo nombre y descripción que las columnas de la tabla (*nombre-tabla1*), vista (*nombre-vista*) o apodo (*apodo*) identificado. El nombre especificado después de LIKE debe identificar una tabla, vista o apodo existentes en el catálogo, o una tabla temporal declarada. No se puede especificar una tabla con tipo ni una vista con tipo (SQLSTATE 428EC).

El uso de LIKE es una definición implícita de *n* columnas, donde *n* es el número de columnas de la tabla o vista identificada.

- Si se designa una tabla, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna de *nombre-tabla1*. Si no se especifica EXCLUDING COLUMN DEFAULTS, también se incluye el valor por omisión de la columna.
- Si se designa una vista, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna resultante de la selección completa definida en *nombre-vista*.

Según cuáles sean las cláusulas de atributos de copia, se pueden incluir o excluir el valor por omisión de la columna y los atributos IDENTITY de la columna.

La definición implícita no incluye otros atributos de la tabla o vista identificada. Por lo tanto, la nueva tabla no tiene ninguna restricción de unicidad, restricción de clave foránea, activadores ni índices. La tabla se crea en el espacio de tablas, de forma implícita o explícita, de acuerdo con lo especificado por la cláusula IN.

Los nombres utilizados para *nombre-tabla1* y *nombre-vista* no pueden ser iguales que el nombre de la tabla temporal global que se está creando (SQLSTATE 428EC).

AS (*selección completa***) DEFINITION ONLY**
Especifica si la definición de tabla se basa en las definiciones de columnas procedentes del resultado de una consulta. El uso de AS (*selección completa*) es una definición implícita de *n* columnas de la tabla temporal global declarada, donde *n* es el número de columnas resultantes de la *selección completa*. Las columnas de la nueva tabla se definen basándose en las columnas que resultan de la *selección completa*. Cada elemento de la lista de selección debe tener un nombre exclusivo (SQLSTATE 42711). La cláusula AS se puede utilizar la cláusula-select para proporcionar nombres exclusivos.

La definición implícita incluye el nombre de columna, el tipo de datos y la posibilidad de contener nulos de cada columna resultante de la *selección completa*.

opciones-copia
Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados original (tabla, vista o selección completa).

INCLUDING COLUMN DEFAULTS
Especifica que deben copiarse los valores por omisión de cada columna actualizable contenida en la definición de la tabla de resultados fuente. Las columnas que no son actualizables no tienen un valor por omisión definido en la correspondiente columna de la tabla creada.

Si se especifica LIKE *nombre-tabla1* y *nombre-tabla1* identifica una tabla base o una tabla temporal declarada, INCLUDING COLUMN DEFAULTS es el valor por omisión.

EXCLUDING COLUMN DEFAULTS
Especifica que no deben copiarse los valores por omisión de columnas contenidos en la definición de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* designa una tabla base o una tabla temporal declarada.

INCLUDING IDENTITY COLUMN ATTRIBUTES
Especifica que deben copiarse, si existen, los atributos de columna de identidad (valores START WITH, INCREMENT BY y CACHE) contenidos en la definición de la tabla de resultados fuente. Se pueden copiar estos atributos si el elemento de la correspondiente columna de la tabla, vista o selección completa es el nombre de una columna de tabla o de vista que, directa o indirectamente, se correlaciona con el nombre de una columna de tabla base que tiene el atributo de identidad. En todos los demás casos, las columnas de la nueva tabla temporal no tendrán el atributo de identidad. Por ejemplo:

- La lista de selección de la selección completa contiene varias instancias del nombre de una columna de identidad (es decir, se selecciona la misma columna más de una vez)
- la lista de selección de la selección completa contiene varias columnas de identidad (es decir, supone la ejecución de una operación de unión)
- la columna de identidad está contenida en una expresión de la lista de selección
- la selección completa contiene una operación de conjuntos (union, except o intersect).

EXCLUDING IDENTITY COLUMN ATTRIBUTES
Especifica que no deben copiarse los atributos de columna de identidad contenidos en la definición de la tabla de resultados fuente.

ON COMMIT
Especifica la acción que se realiza sobre la tabla temporal global cuando se ejecuta una operación COMMIT.

DELETE ROWS
Se suprimen todas las filas de la tabla si no hay ningún cursor abierto en la tabla que esté definido con WITH HOLD. Este es el valor por omisión.

PRESERVE ROWS
Las filas de la tabla se conservan.

NOT LOGGED
Los cambios realizados en la tabla (incluida la creación de la tabla) no se anotan en el archivo de anotaciones cronológicas. Si, cuando se realiza una operación ROLLBACK (o ROLLBACK TO SAVEPOINT), la tabla se crea en la unidad de trabajo (o punto de salvaguarda), ésta se elimina. Si la tabla se ha eliminado de la unidad de trabajo (o punto de salvaguarda), la tabla se restaura, pero sin filas.

ON ROLLBACK
Especifica la acción que va a tener lugar en la tabla temporal global sin anotaciones cronológicas cuando se realice una operación ROLLBACK (o ROLLBACK TO SAVEPOINT).

DELETE ROWS
Si se han cambiado los datos de la tabla, se suprimirán todas las filas. Este es el valor por omisión.

PRESERVE ROWS
Las filas de la tabla se conservan.

WITH REPLACE
Indica que, si ya existe una tabla temporal global declarada con el nombre especificado, la tabla existente es sustituida por la tabla temporal definida en esta sentencia (y se suprimen todas las filas de la tabla existente).

Si no se especifica WITH REPLACE, el nombre especificado no debe identificar una tabla temporal global declarada que ya exista en la sesión actual (SQLSTATE 42710).

IN *nombre-espacio-tablas*
Identifica el espacio de tablas donde se crearán instancias de la tabla temporal global. El espacio de tablas debe existir y estar definido como USER TEMPORARY (SQLSTATE 42838), para el cual el ID de autorización de la sentencia tiene el privilegio USE (SQLSTATE 42501). Si no se especifica esta cláusula, se elige el espacio de tablas USER TEMPORARY con el tamaño de página menor posible para el cual el ID de autorización de la sentencia tenga

el privilegio USE. Cuando existe más de un espacio de tablas que puede elegirse, se establece una prioridad basada en quién recibió el privilegio USE:

1. el ID de autorización
2. un grupo al cual pertenece el ID de autorización
3. PUBLIC

Si todavía existe más de un espacio de tablas que puede elegirse, el gestor de bases de datos toma la decisión final. Cuando no hay ninguna tabla USER TEMPORARY elegible, se produce un error (SQLSTATE 42727).

La determinación del espacio de tablas puede cambiar cuando:

- se eliminan o crean espacios de tablas
- se otorgan o revocan privilegios USE.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Para obtener más detalles, consulte el apartado "Tamaño de fila" en "CREATE TABLE".

PARTITIONING KEY (*nombre-columna,...***)**
Especifica la clave de particionamiento utilizada cuando se particionan los datos de la tabla. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

Si no se especifica esta cláusula y esta tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de particionamiento se define como la primera columna de la tabla temporal declarada.

Para las tablas temporales declaradas, en espacios de tabla definidos en grupos de particiones de base de datos de una sola partición, puede utilizarse cualquier colección de columnas para definir la clave de particionamiento. Si no especifica este parámetro, no se crea ninguna clave de particionamiento.

USING HASHING
Especifica la utilización de la función de generación aleatoria como el método de partición para la distribución de datos. Es el único método de partición soportado.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para OS/390 y z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - CCSID ASCII
      - CCSID UNICODE
- Debe existir un espacio de tablas temporal de usuario antes de poder declarar una tabla temporal definida por el usuario (SQLSTATE 42727).
- **Referencia a una tabla temporal global declarada:** La descripción de una tabla temporal global declarada no aparece en el catálogo de DB2 (SYSCAT.TABLES); por consiguiente, no es permanente y no se puede compartir con otras conexiones a la base de datos. Esto significa que cada sesión que define una tabla temporal global declarada (*nombre-tabla*) tiene su propia descripción exclusiva de esa tabla.

  Para hacer referencia a la tabla temporal global declarada en una sentencia de SQL (distinta de la sentencia DECLARE GLOBAL TEMPORARY TABLE), la tabla debe estar calificada, implícita o explícitamente, por el nombre de esquema SESSION. Si *nombre-tabla* no está calificado por SESSION, las tablas temporales globales declaradas no se tienen en cuenta al resolver la referencia.

Una referencia a SESSION.*nombre-tabla* en una conexión que no ha declarado una tabla temporal global mediante ese nombre intentará realizar la resolución a partir de objetos permanentes del catálogo. Si dicho objeto no existe, se produce un error (SQLSTATE 42704).

- Cuando se enlaza un paquete que tiene sentencias de SQL estático que hacen referencia a tablas calificadas, implícita o explícitamente, por SESSION, esas sentencias no se enlazarán de forma estática. Cuando se invocan estas sentencias, se enlazan de forma incremental, cualquiera que sea la opción de VALIDATE elegida al enlazar el paquete. Durante la ejecución, la resolución de cada referencia a tabla da como resultado una tabla temporal declarada o una tabla permanente. Si no existe ninguna de las dos, se produce un error (SQLSTATE 42704).
- **Privilegios:** Cuando se define una tabla temporal global declarada, el definidor de la tabla recibe todos los privilegios para la tabla, incluida la capacidad para eliminar la tabla. Además, estos privilegios se otorgan a PUBLIC. (Con la opción GRANT no se otorga ninguno de los privilegios, y en la tabla de catálogo no aparece ninguno de los privilegios.) Esto permite que cualquier sentencia de SQL de la sesión haga referencia a una tabla temporal global declarada que ya se ha definido en esa sesión.
- **Creación de instancias y terminación:** En lo que respecta a la explicación que sigue a continuación, P representa una sesión y T es una tabla temporal global declarada de la sesión P:
  - La sentencia DECLARE GLOBAL TEMPORARY TABLE que se ejecuta en P crea una instancia vacía de T.
  - Cualquier sentencia de SQL de P puede hacer referencia a T, y cualquier referencia a T en P es una referencia a esa misma instancia de T.
  - Si se especifica una sentencia DECLARE GLOBAL TEMPORARY TABLE dentro de la sentencia compuesta del procedimiento SQL (definida por BEGIN y END), el ámbito de la tabla temporal global declarada es la conexión, no sólo la sentencia compuesta, y la tabla es conocida fuera de la sentencia compuesta. La tabla no se elimina implícitamente al final de la sentencia compuesta. Una tabla temporal global declarada no se puede definir varias veces con el mismo nombre en otras sentencias compuestas de la sesión, a menos que se haya eliminado explícitamente la tabla.
  - Si se ha especificado, implícita o explícitamente, la cláusula ON COMMIT DELETE ROWS, cuando una operación de confirmación finaliza una unidad de trabajo en P, y no existe ningún cursor abierto en P que esté definido con WITH HOLD y dependa de T, la confirmación incluye la operación DELETE FROM SESSION.T.
  - Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye una modificación en SESSION.T, si se ha especificado NOT LOGGED, la retrotracción incluye la operación DELETE de SESSION.T, a menos que los cambios en T se hayan deshecho.

    Cuando una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye la declaración de SESSION.T, la retrotracción incluye la operación DROP SESSION.T.

    Si una operación de retrotracción finaliza una unidad de trabajo o un punto de salvaguarda en P, y esa unidad de trabajo o punto de salvaguarda incluye la eliminación de la tabla temporal declarada SESSION.T, la retrotracción deshace la eliminación de la tabla. Si se ha especificado NOT LOGGED, la tabla también se habrá vaciado.

  - Cuando el proceso de aplicación donde se declaró T finaliza o se desconecta de la base de datos, se elimina T y se destruyen las instancias de sus filas.
  - Cuando finaliza la conexión con el servidor donde se declaró T, se elimina T y se destruyen las instancias de sus filas.
- **Restricciones sobre la utilización de tablas temporales globales declaradas:** Las tablas temporales globales declaradas no pueden:
  - No se pueden especificar en las sentencias ALTER, COMMENT, GRANT, LOCK, RENAME ni REVOKE (SQLSTATE 42995).
  - No pueden estar referenciadas en las sentencias CREATE ALIAS, CREATE FUNCTION (escalar SQL, de tabla o de fila), CREATE TRIGGER o CREATE VIEW (SQLSTATE 42995).
  - No se pueden especificar en restricciones de referencia (SQLSTATE 42995).

Información relacionada:

- "CREATE TABLE" en la página 341

Ejemplos relacionados:

- "tbtemp.sqc -- How to use a declared temporary table (C)"
- "TbTemp.java -- How to use Declared Temporary Table (JDBC)"

DELETE

La sentencia DELETE suprime filas de una tabla, apodo o vista, o de las tablas, apodos o vistas subyacentes de la selección completa especificada. La supresión de una fila de un apodo suprime la fila del objeto de fuente de datos al que hace referencia el apodo. La supresión de una fila de una vista suprime la fila de la tabla en la que se basa la vista si no se ha definido ningún activador INSTEAD OF para la operación de supresión en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador.

Existen dos formas de esta sentencia:

- La forma DELETE *con búsqueda* se utiliza para suprimir una o varias filas (determinadas opcionalmente mediante una condición de búsqueda).
- La forma DELETE *con posición* se utiliza para suprimir una fila exactamente (determinada por la posición actual del cursor).

Invocación:

Una sentencia DELETE puede incorporarse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Para ejecutar cualquiera de las dos formas de esta sentencia, el ID de autorización de la sentencia debe poseer como mínimo los siguientes privilegios:

- Privilegio DELETE para la tabla, vista o apodo para el que van a suprimirse filas
- Privilegio CONTROL para la tabla, vista o apodo para el que van a suprimirse filas
- Autorización SYSADM o DBADM.

Para ejecutar una sentencia DELETE buscada, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes para cada tabla, vista o apodo al que haga referencia una subconsulta:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Si el paquete utilizado para procesar la sentencia se ha compilado previamente con normas SQL92 (opción LANGLEVEL con un valor de SQL92E o MIA) y la forma buscada de una sentencia DELETE incluye una referencia a una columna de la tabla o vista en la *condición-búsqueda*, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

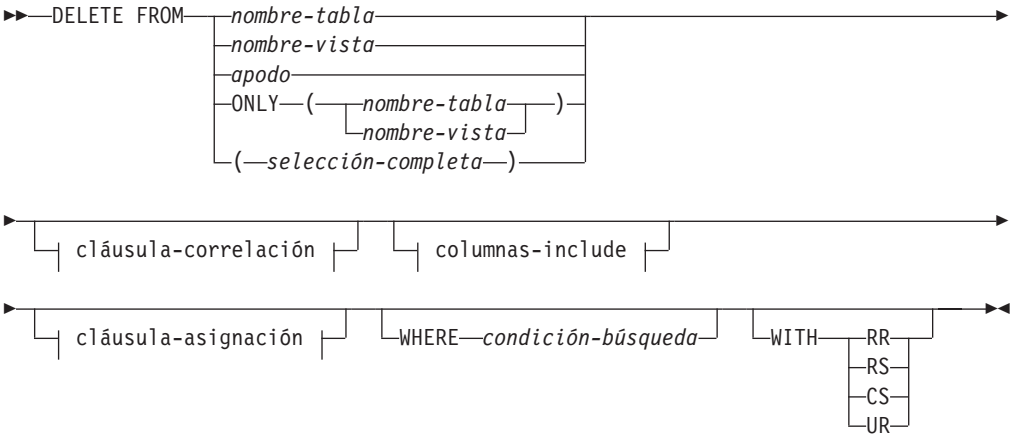
Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

Los privilegios de grupo no se comprueban para las sentencias DELETE estáticas.

Si el destino de la operación de supresión es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

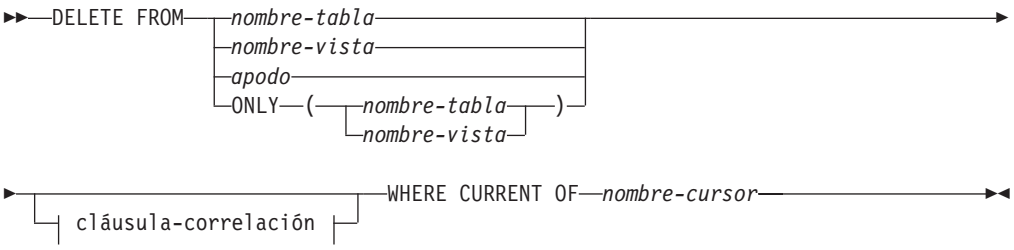
Sintaxis:

supresión-búsqueda:

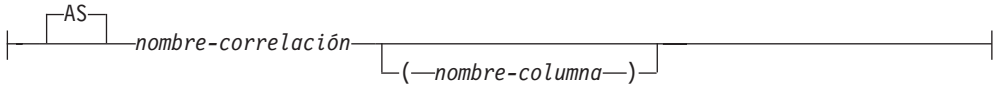


columnas-include:

supresión-posicionada:



cláusula-correlación:



Descripción:

FROM *nombre-tabla*, *nombre-vista*, *apodo* **o** *(selección-completa)*
Identifica el objeto de la operación de supresión. El nombre debe identificar una tabla o vista que exista en el catálogo, pero no debe identificar una tabla

de catálogo, una vista de catálogo, una tabla de consultas materializadas mantenida por el sistema o una vista de sólo lectura.

Si *nombre-tabla* es una tabla con tipo, la sentencia puede suprimir filas de la tabla o cualquiera de sus subtablas correspondientes.

Si *nombre-vista* es una vista con tipo, la sentencia puede que elimine las filas de la vista subyacente o de las vistas subyacentes de las subvistas correspondientes de la vista. Si *nombre-vista* es una vista normal con una tabla subyacente que es una tabla con tipo, puede que la sentencia suprima filas de la tabla con tipo o cualquiera de sus propias subtablas.

Si el objeto de la operación de supresión es una selección completa, esta debe ser suprimible, según lo definido en el elemento de Notas "Vistas suprimibles" de la descripción de la sentencia CREATE VIEW.

Sólo puede hacerse referencia a las columnas de la tabla especificada en la cláusula WHERE. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la tabla o vista en la cláusula FROM sin utilizar ONLY.

FROM ONLY (*nombre-tabla***)**
Aplicable a las tablas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la tabla especificada y no puede suprimir las filas de las subtablas correspondientes. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la tabla en la cláusula FROM utilizando ONLY. Si *nombre-tabla* no es una tabla con tipo, la palabra clave ONLY no tiene ningún efecto en la sentencia.

FROM ONLY (*nombre-vista***)**
Aplicable a las vistas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la vista especificada y no puede suprimir las filas de las subvistas correspondientes. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la vista en la cláusula FROM utilizando ONLY. Si *nombre-vista* no es una vista con tipo, la palabra clave ONLY no tiene ningún efecto en la sentencia.

cláusula-correlación
Se puede utilizar dentro de la *condición-búsqueda* para designar una tabla, vista, apodo o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte "referencia-tabla" en la descripción de "Subselección".

columnas-include
Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia DELETE cuando está anidada en la cláusula FROM de una selección completa. Las *columnas-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

INCLUDE
Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia DELETE.

nombre-columna
Especifica una columna de la tabla de resultados intermedia de la sentencia DELETE. El nombre no puede coincidir con el nombre de otra columna include ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

tipo-datos
Especifica el tipo de datos de la columna include. El tipo de datos debe ser uno que reciba soporte de la sentencia CREATE TABLE.

cláusula-asignación
Consulte la descripción de *cláusula-asignación* bajo la sentencia UPDATE. Se aplican las mismas reglas. Las *columnas-include* son las únicas columnas que se pueden definir utilizando la *cláusula-asignación* (SQLSTATE 42703).

WHERE
Especifica una condición que selecciona las filas que se deben suprimir. Puede omitirse la cláusula, se puede especificar una condición de búsqueda o nombrar un cursor. Si se omite la cláusula, se suprimen todas las filas de la tabla o vista.

condición-búsqueda
Cada *nombre-columna* de la condición de búsqueda, que no sea una subconsulta, debe identificar una columna de la tabla o vista.

La *condición-búsqueda* se aplica a cada fila de la tabla, vista o apodo y las filas que se suprimen son aquellas para las que el resultado de la *condición-búsqueda* es verdadero.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que ésta se ejecuta cada vez que se aplica la *condición de búsqueda* a una fila y que los resultados se utilizan para aplicar la *condición de búsqueda*. De hecho, una subconsulta sin referencias correlacionadas sólo se ejecuta una vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila. Si una subconsulta hace referencia a una tabla de objetos de una sentencia DELETE o a una tabla dependiente con una norma de supresión de CASCADE o SET NULL, la subconsulta se evalúa por completo antes de suprimir cualquier fila.

CURRENT OF *nombre-cursor*
Identifica un cursor que se ha definido en una sentencia DECLARE CURSOR del programa. La sentencia DECLARE CURSOR debe preceder a la sentencia DELETE.

La tabla, vista o apodo que se ha indicado también debe indicarse en la cláusula FROM de la sentencia SELECT del cursor y la tabla de resultados del cursor no debe ser de sólo lectura. (Para obtener información acerca de las tablas de resultados de sólo lectura, consulte "DECLARE CURSOR".)

Cuando se ejecuta la sentencia DELETE, el cursor debe posicionarse en una fila: dicha fila es la que se suprime. Después de la supresión, el cursor se posiciona antes de la siguiente fila de la tabla de resultados. Si no hay una fila siguiente, el cursor se posiciona después de la última fila.

WITH
Especifica el nivel de aislamiento utilizado al localizar las filas que se deben suprimir.

RR
Lectura repetible

RS
Estabilidad de lectura

CS
Estabilidad del cursor

UR
Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

Normas:

- ***Activadores:*** Las sentencias DELETE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en las filas suprimidas. Si una sentencia DELETE de una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobará la integridad referencial de las actualizaciones que se han realizado en el activador y no la de las tablas subyacentes de la vista que ha dado lugar a que se ejecutara el activador.
- ***Integridad referencial:*** Si la tabla identificada o la tabla base de la vista identificada es padre, las filas seleccionadas para la supresión no deberán tener ningún dependiente en una relación con una norma de supresión de RESTRICT, y la sentencia DELETE no deberá aplicarse en cascada a las filas descendentes que tengan dependientes en una relación con una norma de supresión de RESTRICT.

  Si no se impide la operación de supresión por una norma de supresión RESTRICT, se suprimen las filas seleccionadas. Las fila que son dependientes de la filas seleccionadas también se ven afectadas:

  – Las columnas con posibilidad de contener nulos de las claves foráneas de cualquier fila que sea dependiente en una relación con una norma de supresión de SET NULL se establecen en el valor nulo.
  – Las filas que sean sus dependientes en una relación con una norma de supresión de CASCADE también se suprimen y se aplican, a su vez, las normas anteriores a estas filas.

  Se comprueba la norma de supresión de NO ACTION para imponer que las claves foráneas no nulas hagan referencia a una fila padre existente después de que se hayan impuesto otras restricciones de referencia.

Notas:

- Si se produce un error durante la ejecución de una sentencia DELETE de múltiples filas, no se realiza ningún cambio en la base de datos.
- Salvo que ya existan los bloqueos adecuados, se adquieren uno o más bloqueos exclusivos durante la ejecución de una sentencia DELETE satisfactoria. La emisión de una sentencia COMMIT o ROLLBACK liberará los bloqueos. Hasta que se liberen los bloqueos por una operación de confirmación o de retrotracción, el efecto de la operación de supresión sólo lo percibirán:

  – El proceso de aplicación que ha realizado la supresión
  – Otro proceso de aplicación que utilice el nivel de aislamiento UR.

  Los bloqueos pueden evitar que otros procesos de aplicación realicen operaciones en la tabla.
- Si un proceso de aplicación suprime una fila en la que está posicionado alguno de sus cursores, dichos cursores se posicionan antes de la siguiente fila de la tabla de resultados. Supongamos que C sea un cursor que está situado antes de la fila R (como resultado de una operación OPEN, una operación DELETE a través de C, una operación DELETE a través de otro cursor o una operación DELETE con búsqueda). Si existen operaciones INSERT, UPDATE y DELETE que afectan a la tabla base de la que deriva R, la siguiente operación FETCH que

haga referencia a C no posiciona necesariamente C en R. Por ejemplo, la operación puede posicionar C en R', donde R' es una nueva fila que ahora es la fila siguiente de la tabla de resultados.

- SQLERRD(3) en la SQLCA muestra el número de filas que van a incluirse en la operación de supresión. En el contexto de una sentencia de procedimiento de SQL, el valor puede recuperarse utilizando la variable ROW_COUNT de la sentencia GET DIAGNOSTICS. SQLERRD(5) en la SQLCA muestra el número de filas afectadas por las restricciones de referencia y por las sentencias activadas. Incluye filas que se han suprimido como resultado de una norma de supresión CASCADE y filas en las que se han establecido claves foráneas en NULL como resultado de una norma de supresión SET NULL. En relación a la sentencias activadas, incluye el número de filas que se han insertado, actualizado o suprimido.
- Si se produce un error que impide la supresión de todas las filas que coincidan con la condición de búsqueda y todas las operaciones necesarias para las restricciones de referencia existentes, no se realiza ningún cambio en la tabla y se devuelve un error.
- Para los apodos, la opción del servidor externo `iud_app_svpt_enforce` plantea una limitación adicional. Consulte la documentación acerca de los objetos federados para obtener más información.
- Para algunas fuentes de datos, SQLCODE -20190 podría devolverse en una supresión realizada para un apodo debido a una posible falta de coherencia en los datos. Consulte la documentación acerca de los objetos federados para obtener más información.
- En cualquier fila suprimida que incluyera archivos actualmente enlazados mediante columnas DATALINK, los archivos se desenlazan y se restaurarán o se suprimirán según la definición de la columna DATALINK.

  Podría producirse un error al intentar suprimir un valor de DATALINK si el servidor de archivos al que se hace referencia en el valor ya no está registrado en el servidor de bases de datos (SQLSTATE 55022).

  También puede producirse un error cuando se suprima una fila que tenga un enlace con un servidor que no esté disponible en el momento de la supresión (SQLSTATE 57050).

Ejemplos:

Ejemplo 1: Suprima el departamento (DEPTNO) 'D11' de la tabla DEPARTMENT.

```
DELETE FROM DEPARTMENT
 WHERE DEPTNO = 'D11'
```

Ejemplo 2: Suprima todos los departamentos de la tabla DEPARTMENT (es decir, vacíe la tabla).

```
DELETE FROM DEPARTMENT
```

Ejemplo 3: Suprima de la tabla EMPLOYEE los representantes de ventas o los representantes de zona que no hayan realizado ninguna venta en 1995.

```
DELETE FROM EMPLOYEE
  WHERE LASTNAME NOT IN
    (SELECT SALES_PERSON
      FROM SALES
      WHERE YEAR(SALES_DATE)=1995)
      AND JOB IN ('SALESREP','FIELDREP')
```

Ejemplo 4: Suprima todas las filas de empleado duplicadas de la tabla EMPLOYEE. Una fila de empleado se considera duplicada si coinciden los apellidos. Conserve la fila de empleado con el nombre menor en orden léxico.

```
DELETE FROM
  (SELECT ROWNUMBER() OVER (PARTITON BY LASTNAME ORDER BY>/ph> FIRSTNME)
    FROM EMPLOYEE) AS E(RN)
    WHERE RN = 1
```

Información relacionada:

- "Condiciones de búsqueda" en la publicación *Consulta de SQL, Volumen 1*
- "Subselección" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE VIEW" en la página 466
- "DECLARE CURSOR" en la página 483
- "UPDATE" en la página 761
- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbuse.c -- How to use a database"
- "tbmod.c -- How to modify table data"
- "dbuse.sqc -- How to use a database (C)"
- "tbconstr.sqc -- How to create, use, and drop constraints (C)"
- "tbmod.sqc -- How to modify table data (C)"
- "dbuse.sqC -- How to use a database (C++)"
- "tbconstr.sqC -- How to create, use, and drop constraints (C++)"
- "tbmod.sqC -- How to modify table data (C++)"
- "delet.sqb -- How to delete table data (MF COBOL)"
- "updat.sqb -- How to update, delete and insert table data (MF COBOL)"
- "DbUse.java -- How to use a database (JDBC)"
- "TbConstr.java -- How to create, use and drop constraints (JDBC)"
- "TbMod.java -- How to modify table data (JDBC)"
- "DbUse.sqlj -- How to use a database (SQLj)"
- "TbConstr.sqlj -- How to create, use and drop constraints (SQLj)"
- "TbMod.sqlj -- How to modify table data (SQLj)"

DESCRIBE

La sentencia DESCRIBE obtiene información acerca de una sentencia preparada.

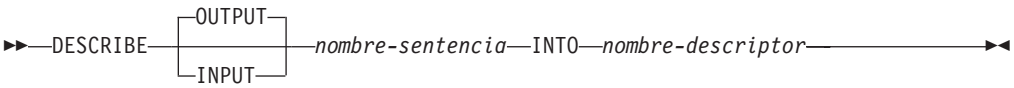
Invocación:

Esta sentencia sólo se puede incluir en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:

OUTPUT
Palabra clave opcional que se utiliza para indicar que el programa de utilidad de descripción debe obtener información acerca de las columnas de lista de selección en la sentencia preparada o bien los marcadores de parámetro de salida que se asocian a los parámetros OUT e INOUT, para la llamada a un procedimiento almacenado.

INPUT
Especifica que el programa de utilidad de descripción debe obtener información acerca de los marcadores de parámetro de salida en una sentencia preparada. Para una sentencia CALL, esto incluye los marcadores de parámetro que se asocian a los parámetros IN e INOUT para el procedimiento almacenado. Los marcadores de parámetro de entrada se consideran siempre anulables, independientemente de su uso.

nombre-sentencia
Identifica la sentencia sobre la que se necesita información. Cuando se ejecuta la sentencia DESCRIBE, el nombre debe identificar una sentencia preparada.

INTO *nombre-descriptor*
Identifica un área de descriptor de SQL (SQLDA). Antes de ejecutar la sentencia DESCRIBE, deben establecerse las siguientes variables en la SQLDA:

SQLN Indica el número de variables representadas por SQLVAR. (SQLN proporciona la dimensión de la matriz SQLVAR.) SQLN debe establecerse en un valor mayor o igual que cero antes de ejecutar la sentencia DESCRIBE.

Cuando se ejecuta la sentencia DESCRIBE, el gestor de bases de datos asigna valores a las variables de la SQLDA de la siguiente manera:

SQLDAID
Los 6 primeros bytes se establecen en 'SQLDA ' (es decir, 5 letras seguidas del carácter de espacio).

El séptimo byte, llamado SQLDOUBLED, se establece en '2' si la SQLDA contiene dos entradas SQLVAR para cada elemento de la lista-selección (o la *columna* de la tabla de resultados). Esta técnica se utiliza para acomodar

columnas resultantes de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia. De lo contrario, SQLDOUBLED se establece en el carácter de espacio.

Se establece el doble distintivo en espacio si no hay suficiente espacio en la SQLDA para contener toda la respuesta DESCRIBE.

El octavo byte se establece en el carácter de espacio.

SQLDABC
Longitud de la SQLDA.

SQLD Si la sentencia preparada es SELECT, el número de columnas de su tabla de resultados; de lo contrario, 0.

SQLVAR
Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las ocurrencias de SQLVAR.

Si el valor es *n*, donde *n* es mayor que 0 pero menor o igual al valor de SQLN, los valores se asignan a las *n* primeras ocurrencias de SQLVAR para que la primera ocurrencia de SQLVAR contenga una descripción de la primera columna de la tabla de resultados, la segunda ocurrencia de SQLVAR contenga una descripción de la segunda columna de la tabla de resultados, etcétera. La descripción de una columna consta de los valores asignados a SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN y SQLDATATYPE_NAME.

SQLVAR base

SQLTYPE
Un código que muestra el tipo de datos de la columna y si puede contener valores nulos o no.

SQLLEN
Un valor de longitud que depende del tipo de datos de las columnas del resultado. SQLLEN es 0 para tipos de datos LOB.

SQLNAME
El valor de la variable sqlname se obtiene como se indica a continuación:

- Si SQLVAR corresponde a una columna que se ha obtenido para una referencia a una columna simple en la lista de selección de la sentencia-select, sqlname es el nombre de la columna.
- Si SQLVAR corresponde a un marcador de parámetro que no forma parte de una expresión en la lista de parámetros de un procedimiento almacenado, sqlname contiene el nombre del parámetro, si se ha especificado uno en CREATE PROCEDURE.
- De otro modo, sqlname contiene un valor literal numérico ASCII que representa la posición de SQLVAR dentro de la SQLDA.

SQLVAR secundaria

Estas variables sólo se utilizan si el número de entradas SQLVAR se doblan para acomodar columnas de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN
El atributo de longitud de una columna BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME
Para cualquier columna de tipo definido por el usuario (diferenciado o estructurado), el gestor de bases de datos establece esta opción en el nombre del tipo definido por el usuario, calificado al completo. Para una columna de tipo de referencia, el gestor de bases de datos establece esta opción en el nombre definido por el usuario, calificado al completo, del tipo destino de la referencia. De lo contrario, el nombre de esquema es SYSIBM y el nombre de tipo es el nombre que aparece en la columna TYPENAME de la vista de catálogo SYSCAT.DATATYPES.

Notas:

- Antes de ejecutar la sentencia DESCRIBE, el valor de SQLN debe establecerse de manera que indique cuántas ocurrencias de SQLVAR se proporcionan en la SQLDA y debe asignarse suficiente almacenamiento para contener ocurrencias de SQLN. Por ejemplo, para obtener la descripción de las columnas de la tabla de resultados de una sentencia SELECT preparada, el número de apariciones de SQLVAR no debe ser menor que el número de columnas.
- Si se espera un LOB de gran tamaño, tenga en cuenta que el manejo de este LOB afectará a la memoria de la aplicación. Si se da esta condición, considere la posibilidad de utilizar localizadores o variables de referencia a archivos. Modifique la SQLDA después de haber ejecutado la sentencia DESCRIBE pero antes de asignar almacenamiento para que un SQLTYPE de SQL_TYP_xLOB se cambie por SQL_TYP_xLOB_LOCATOR o un SQL_TYP_xLOB_FILE por los correspondientes cambios que se han realizado en otros campos como, por ejemplo, SQLLEN. Después asigne el almacenamiento basado en SQLTYPE y continúe.
- Las conversiones de páginas de códigos entre el código Unix ampliado (EUC) y las páginas de códigos DBCS pueden dar como resultado la expansión y contracción de las longitudes de caracteres.
- Si se selecciona un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se especificó ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 428EM), o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744)), DESCRIBE emitirá un error.
- **Asignación de la SQLDA:** Entre las maneras posibles de asignar la SQLDA están las tres descritas abajo.

  Primera técnica: Asigne una SQLDA con suficientes ocurrencias de SQLVAR para acomodar cualquier lista de selección que la aplicación vaya a tener que procesar. Si la tabla contiene cualquier columna de tipo LOB, tipo diferenciado, tipo estructurado o de tipo de referencia, el número de las SQLVAR debe ser el doble que el número máximo de columnas; de lo contrario, el número debe ser igual al número máximo de columnas. Después de realizar la asignación, la aplicación puede utilizar esta SQLDA repetidas veces.

  Esta técnica utiliza una gran cantidad de almacenamiento que nunca se desasigna, incluso cuando la mayor parte de su almacenamiento no se utilice para una lista de selección en particular.

  Segunda técnica: Repita los dos siguientes pasos para cada lista de selección procesada:

  1. Ejecute una sentencia DESCRIBE con una SQLDA que no tenga ninguna ocurrencia de SQLVAR; es decir, una SQLDA cuyo SQLN sea cero. El valor devuelto para SQLD es el número de columnas de la tabla de resultados. Este es el número necesario de ocurrencias de SQLVAR o la mitad del

número necesario. Puesto que no había ninguna entrada SQLVAR, se emitirá un aviso con SQLSTATE 01005. Si el SQLCODE que se asocia a ese aviso es +237, +238 o +239, el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD. (En la devolución de estos SQLCODE positivos se da por supuesto que el valor de la opción de enlace SQLWARN era YES (devolución de SQLCODE positivos). Si SQLWARN se había establecido en NO, sigue devolviéndose +238 para indicar que el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD.)

2. Asigne una SQLDA con suficientes ocurrencias de SQLVAR. Después ejecute la sentencia DESCRIBE de nuevo, utilizado esta SQLDA nueva.

Esta técnica permite una mejor gestión del almacenamiento que la primera técnica, pero dobla el número de sentencias DESCRIBE.

Tercera técnica: Asigne una SQLDA que sea lo suficientemente grande para manejar la mayoría de, y quizá todas, las listas de selección pero que también sea razonablemente pequeña. Ejecute DESCRIBE y compruebe el valor SQLD. Utilice el valor SQLD para el número de ocurrencias de SQLVAR para asignar una SQLDA mayor, si es necesario.

Esta técnica está comprendida entre las dos primeras técnicas. Su eficacia depende de una buena elección del tamaño de la SQLDA original.

Ejemplo:

En un programa C, ejecute una sentencia DESCRIBE con una SQLDA que no tenga ninguna ocurrencia de SQLVAR. Si SQLD es mayor que cero, utilice el valor para asignar una SQLDA con el número necesario de ocurrencias de SQLVAR y después ejecute una sentencia DESCRIBE que utilice dicha SQLDA.

```
EXEC SQL  BEGIN DECLARE SECTION;    char    stmt1_str[200];
  EXEC SQL  END DECLARE SECTION;
  EXEC SQL  INCLUDE SQLDA;
  EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

  ... /* código para solicitar una consulta al usuario, después generar */
      /* una sentencia-select en la stmt1_str            */
  EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

  ... /* código para establecer SQLN en cero y asignar la SQLDA */
  EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

  ... /* código para comprobar que SQLD se mayor que cero, para establecer */
      /* SQLN en SQLD, después para volver a asignar SQLDA          */
  EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

  ... /* código para preparar la utilización de SQLDA             */
      /* y asignar almacenamientos intermedios para recibir datos   */
  EXEC SQL  OPEN DYN_CURSOR;

  ... /* bucle para leer filas de la tabla de resultados
*/
  EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
   .
   .
   .
```

Información relacionada:

- "PREPARE" en la página 634
- "SQLDA (área de descriptores de SQL)" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "tbread.sqc -- How to read tables (C)"
- "tbread.sqC -- How to read tables (C++)"

DISCONNECT

La sentencia DISCONNECT finaliza una o más conexiones cuando no existe ninguna unidad de trabajo activa (es decir, tras una operación de confirmación o de retrotracción). Si el destino de la sentencia DISCONNECT es una única conexión, la conexión finaliza sólo si la base de datos ha participado en una unidad de trabajo existente, con independencia de si existe o no una unidad de trabajo activa. Por ejemplo, si otras bases de datos han realizado tareas, pero el destino en cuestión no, la desconexión puede tener lugar sin finalizarse la conexión.

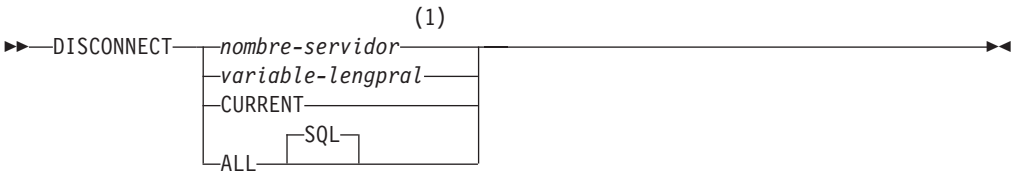
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna.

Sintaxis:



Notas:

1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal.

Descripción:

nombre-servidor o *variable-lengpral*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante la *variable-lengpral* donde está contenido el *nombre-servidor*.

Si se especifica una *variable-lengpral*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lengpral* debe estar justificado por la izquierda y no estar delimitado por comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El alias-basedatos especificado o el alias-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el alias-basedatos no identifica ninguna conexión existente, se genera un error (SQLSTATE 08003).

CURRENT
Identifica la conexión actual del proceso de aplicación. El proceso de aplicación debe estar en el estado conectado. Si no se genera un error (SQLSTATE 08003).

ALL
Indica que se deben destruir todas las conexiones existentes del proceso de aplicación. No se produce ningún error ni aviso si no existen conexiones cuando se ejecuta la sentencia. La palabra clave opcional SQL se incluye para ser coherente con la sintaxis de la sentencia RELEASE.

Normas:

- Generalmente, la sentencia DISCONNECT no se puede ejecutar mientras se está en una unidad de trabajo. Si se intenta, se genera un error (SQLSTATE 25000). La excepción a esta norma es si se especifica una sola conexión que se haya de desconectar y la base de datos no ha participado en una unidad de trabajo existente. En este caso, no importa si hay una unidad de trabajo activa cuando se emite la sentencia DISCONNECT.
- La sentencia DISCONNECT no se puede ejecutar nunca en el entorno del Supervisor del Proceso de transacción (TP) (SQLSTATE 25000). Si se utiliza cuando la opción del precompilador es SYNCPOINT se establece en TWOPHASE.

Notas:

- Si la sentencia DISCONNECT se realiza satisfactoriamente, se destruye cada conexión.

  Si la sentencia DISCONNECT no es satisfactoria, el estado de la conexión del proceso de aplicación y los estados de sus conexiones no se cambian.
- Si se utiliza DISCONNECT para destruir la conexión actual, la siguiente sentencia de SQL ejecutada debe ser CONNECT o SET CONNECTION.
- La semántica de CONNECT Tipo 1 no excluye la utilización de DISCONNECT. Sin embargo, aunque se puedan utilizar DISCONNECT CURRENT y DISCONNECT ALL, no dan como resultado una operación de confirmación como lo haría la sentencia CONNECT RESET.

  Si se especifica *nombre-servidor* o *variable-lengpral* en la sentencia DISCONNECT, debe identificar la conexión actual porque CONNECT Tipo 1 sólo da soporte a una conexión cada vez. Generalmente, DISCONNECT caerá dentro de una unidad de trabajo con la excepción señalada en “Normas”.
- Es necesario que los recursos creen y mantengan conexiones remotas. Por lo tanto, una conexión remota que no se vaya a volver a utilizar debe destruirse lo más pronto posible.
- Las conexiones también se pueden destruir durante una operación de confirmación porque la opción de conexión esté en vigor. La opción de conexión podría ser AUTOMATIC, CONDITIONAL o EXPLICIT, que puede establecerse como una opción del precompilador o a través de la API SET CLIENT en tiempo de ejecución. Para obtener información acerca de la especificación de la opción DISCONNECT, consulte el apartado “Bases de datos relacionales distribuidas”.

Ejemplos:

Ejemplo 1: La aplicación ya no necesita la conexión SQL con IBMSTHDB. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir la conexión.

```
EXEC SQL DISCONNECT IBMSTHDB;
```

Ejemplo 2: La aplicación ya no necesita la conexión actual. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir la conexión.

```
EXEC SQL DISCONNECT CURRENT;
```

Ejemplo 3: La aplicación ya no necesita las conexiones existentes. Debe ejecutarse la sentencia siguiente después de una operación de confirmación o de retrotracción para destruir todas las conexiones.

```
EXEC SQL DISCONNECT ALL;
```

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbconn.sqc -- How to connect to and disconnect from a database (C)"
- "dbmcon.sqc -- How to use multiple databases (C)"
- "dbconn.sqC -- How to connect to and disconnect from a database (C++)"
- "dbmcon.sqC -- How to use multiple databases (C++)"
- "Util.java -- Utilities for JDBC sample programs (JDBC)"
- "Util.sqlj -- Utilities for SQLJ sample programs (SQLj)"

DROP

La sentencia DROP suprime un objeto. Cualquier objeto que sea dependiente directa o indirectamente de dicho objeto se suprime o pasa a estar no operativo. Siempre que se suprime un objeto, se suprime su descripción del catálogo y se invalidan los paquetes que hacen referencia al objeto.

Invocación:

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

Autorización:

El ID de autorización de la sentencia DROP debe incluir uno de los privilegios siguientes cuando se eliminan objetos que admiten nombres de dos partes; de lo contrario se produce un error (SQLSTATE 42501):

- Autorización SYSADM o DBADM
- Privilegio DROPIN para el esquema del objeto
- Definidor del objeto, tal como está registrado en la columna DEFINER de la vista de catálogo del objeto
- Privilegio CONTROL para el objeto (aplicable sólo a índices, especificaciones de índices, apodos, paquetes, tablas y vistas).
- Definidor del tipo definido por el usuario, tal como está registrado en la columna DEFINER de la vista de catálogo SYSCAT.DATATYPES (sólo aplicable al eliminar un método asociado a un tipo definido por el usuario)

El ID de autorización de la sentencia DROP para eliminar una jerarquía de tablas o de vistas debe tener uno de los privilegios anteriores para cada tabla o vista de la jerarquía.

Cuando se elimina un esquema, el ID de autorización de la sentencia DROP debe tener autorización SYSADM o DBADM, o ser el propietario de esquema tal como está registrado en la columna OWNER de SYSCAT.SCHEMATA.

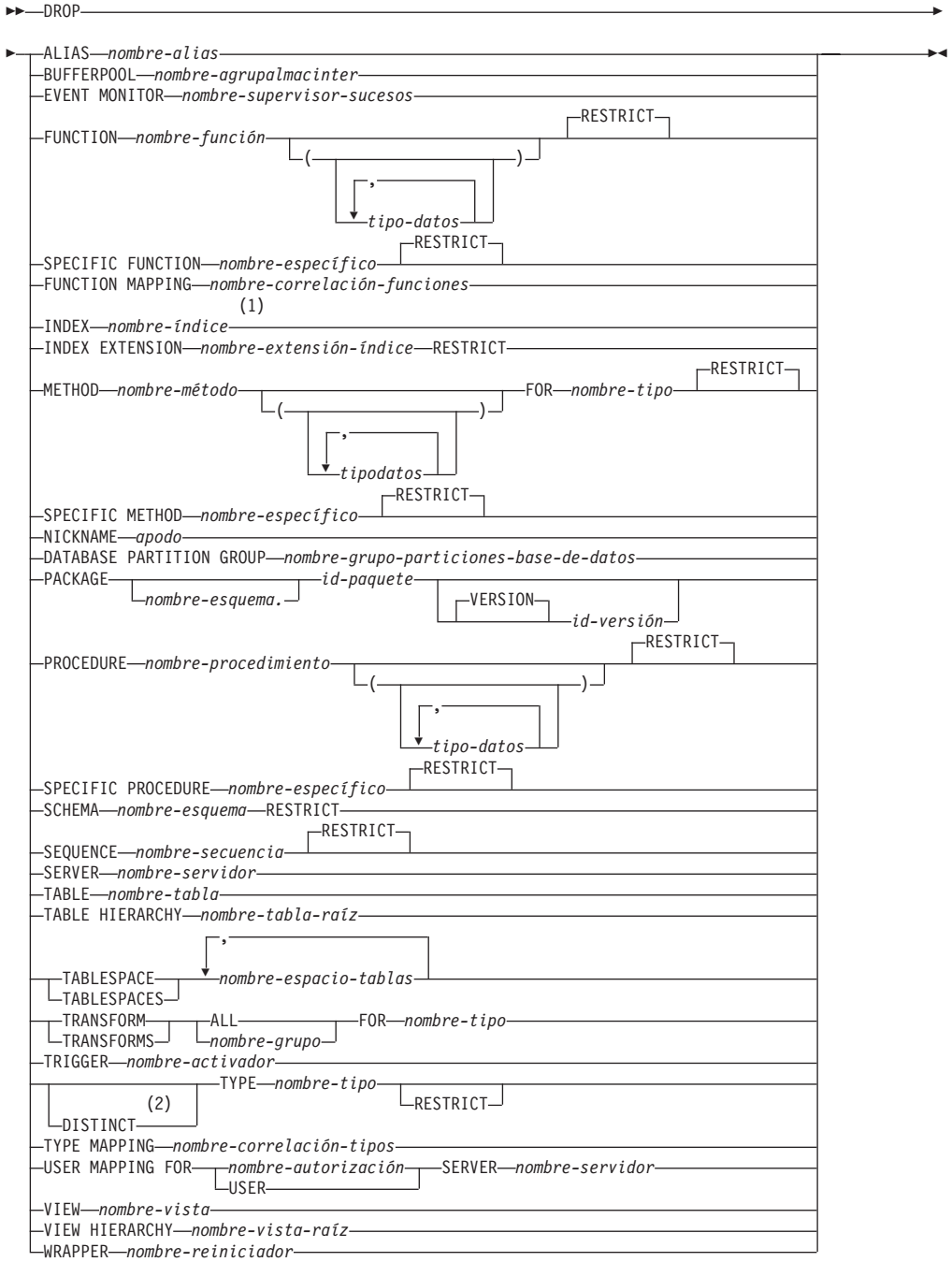
Cuando se elimina una agrupación de almacenamientos intermedios, un grupo de particiones de base de datos o un espacio de tabla, el ID de autorización de la sentencia DROP debe tener autorización SYSADM o SYSCTRL.

Cuando se elimina un supervisor de sucesos, una definición de servidor, una correlación de tipos de datos, una función de correlación o reiniciador, el ID de autorización de la sentencia DROP debe tener una autorización SYSADM o DBADM.

Cuando se elimina una correlación de usuarios, intermedios, el ID de autorización de la sentencia DROP debe tener autorización SYSADM o SYSCTRL si este ID de autorización es diferente del nombre de autorización de la base de datos federada dentro de la correlación. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o autorizaciones.

Cuando se elimina una transformación, el ID de autorización de la sentencia DROP debe tener autorización SYSADM o DBADM, o ser el definidor de *nombre-tipo*.

Sintaxis:



Notas:

1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.

2 También puede utilizarse DATA cuando se elimine cualquier tipo definido por el usuario.

Descripción:

ALIAS *nombre-alias*
Identifica el alias que se debe eliminar. El *nombre-alias* debe designar un alias descrito en el catálogo (SQLSTATE 42704). Se suprime el alias especificado.

Todas las tablas, vistas y activadores que hacen referencia al alias dejan de ser operativos. (Esto incluye a la tabla a la que se hace referencia en la cláusula ON de la sentencia CREATE TRIGGER y a todas las tablas a las que se hace referencia dentro de las sentencias de SQL activadas.)

BUFFERPOOL *nombre-agrupalmacinter*
Identifica la agrupación de almacenamientos intermedios que se debe eliminar. El *nombre-agrupalmacinter* debe identificar una agrupación de almacenamientos intermedios que se haya descrito en el catálogo (SQLSTATE 42704). Puede que no haya ningún espacio de tablas asignado a la agrupación de almacenamientos intermedios (SQLSTATE 42893). La agrupación de almacenamientos intermedios IBMDEFAULTBP no se puede eliminar (SQLSTATE 42832). La memoria de la agrupación de almacenamientos intermedio se libera inmediatamente, para que DB2 pueda utilizarla. Puede que el almacenamiento en disco no se libere hasta que se produzca la siguiente conexión con la base de datos.

EVENT MONITOR *nombre-supervisor-sucesos*
Identifica el supervisor de sucesos que se debe eliminar. El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que se haya descrito en el catálogo (SQLSTATE 42704).

Si el supervisor de sucesos identificado tiene el valor ON, se devolverá un error (SQLSTATE 55034); de lo contrario, se suprimirá el supervisor de sucesos.

Si hay archivos de sucesos en la vía de acceso de destino del supervisor de sucesos cuando se elimina el supervisor de sucesos, los archivos de sucesos no se suprimen. Sin embargo, si se crea un nuevo supervisor de sucesos que especifique la misma vía de acceso de destino, se suprimirán los archivos de sucesos.

Cuando se eliminan supervisores de sucesos WRITE TO TABLE, se elimina la información de tabla de la vista de catálogo SYSCAT.EVENTTABLES, pero no se eliminan las tablas en sí.

FUNCTION
Identifica una instancia de una función definida por el usuario (una función completa o una plantilla de función) que se debe eliminar. La instancia de función especificada debe ser una función definida por el usuario descrita en el catálogo. Las funciones generadas implícitamente por la sentencia CREATE DISTINCT TYPE no se pueden eliminar.

Hay varias maneras distintas disponibles para identificar la instancia de función:

FUNCTION *nombre-función*
Identifica la función específica, y sólo es válido si hay exactamente una instancia de función con el *nombre-función*. La función así identificada puede tener cualquier número de parámetros definidos para la misma. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Si no existe ninguna función con este nombre en el esquema nombrado o implícito, se produce un error (SQLSTATE 42704). Si

existe más de una instancia específica de la función en el esquema mencionado o implicado, se producirá un error (SQLSTATE 42725).

FUNCTION *nombre-función* **(***tipo-datos***,...)**
Proporciona la signatura de la función, que identifica de manera exclusiva la función que se debe eliminar. El algoritmo de selección de funciones no se utiliza.

nombre-función
Proporciona el nombre de función de la función que se debe eliminar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos, ...)
Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utiliza para identificar la instancia de función específica que se debe eliminar.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

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

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

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

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que 0 <n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

RESTRICT
La palabra clave RESTRICT aplica la norma que establece que la función no se eliminará si existe alguna de las dependencias siguientes:

- Otra rutina tiene la función como origen.
- Una vista utiliza la función.
- Un activador utiliza la función.
- Una tabla de consultas materializadas utiliza la función de su definición.

RESTRICT es el comportamiento por omisión.

Si no se especifica ninguna función con la signatura especificada en el esquema nombrado o implícito, se genera un error (SQLSTATE 42883).

SPECIFIC FUNCTION *nombre-específico*
Identifica la función definida por el usuario en particular que se debe eliminar, utilizando el nombre específico especificado o que toma por omisión en el momento de creación de la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema nombrado o implícito; de lo contrario, se produce un error (SQLSTATE 42704).

RESTRICT
La palabra clave RESTRICT aplica la norma que establece que la función no se eliminará si existe alguna de las dependencias siguientes:

- Otra rutina tiene la función como origen.
- Una vista utiliza la función.
- Un activador utiliza la función.

RESTRICT es el comportamiento por omisión.

No es posible eliminar una función que se encuentre en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

Otros objetos pueden depender de una función. Deben eliminarse todas estas dependencias antes de que pueda eliminarse la función, a excepción de los paquetes que están marcados como no operativos. El intento de eliminar una función con dichas dependencias dará como resultado un error (SQLSTATE 42893). En la página 527 encontrará una lista de estas dependencias.

Si la función puede eliminarse, se elimina.

Cualquier paquete dependiente de la función específica que se está eliminando se marca como no operativo. Dicho paquete no se vuelve a enlazar implícitamente. Deberá volver a enlazarse mediante la utilización del mandato BIND o REBIND o deberá volver a prepararse mediante la utilización del mandato PREP.

FUNCTION MAPPING *nombre-correlación-funciones*
Identifica la correlación de funciones que se debe descartar. El *nombre-correlación-funciones* debe identificar una correlación de funciones definida por el usuario que se haya descrito en el catálogo (SQLSTATE 42704). La correlación de funciones se suprime de la base de datos.

Las correlaciones de funciones por omisión no se pueden descartar, pero se pueden inhabilitar utilizando la sentencia CREATE FUNCTION MAPPING. Si se descarta una correlación de funciones definida por el usuario que se había creado para alterar temporalmente la correlación de funciones por omisión, se restituye la correlación de funciones por omisión.

Los paquetes que tengan una dependencia de la correlación de funciones eliminada se invalidarán.

INDEX *nombre-índice*
Identifica el índice o especificación de índice que se debe eliminar. El *nombre-índice* debe identificar un índice o especificación de índice que se describa en el catálogo (SQLSTATE 42704). No puede ser un índice que el sistema necesita para una clave primaria o restricción de unicidad o para una

tabla de consultas materializadas duplicada (SQLSTATE 42917). El índice especificado o la especificación de índice se suprime.

Los paquetes que tengan una dependencia de un índice o de una especificación eliminada se invalidarán.

INDEX EXTENSION *nombre-extensión-índice* **RESTRICT**
Identifica la extensión de índice que se debe eliminar. El *nombre-extensión-índice* debe identificar una extensión de índice que esté descrita en el catálogo (SQLSTATE 42704). La palabra clave RESTRICT impide definir cualquier índice que dependa de esta definición de extensión de índice (SQLSTATE 42893).

METHOD
Identifica un cuerpo de método que se debe eliminar. El cuerpo de método especificado debe ser un método descrito en el catálogo (SQLSTATE 42704). Los cuerpos de método que son generados implícitamente por la sentencia CREATE TYPE no se pueden eliminar.

DROP METHOD suprime el cuerpo de un método, pero la especificación del método (signatura) se conserva como parte de la definición del tipo sujeto. Después de eliminar el cuerpo de un método, se puede eliminar la especificación del método en la definición del tipo sujeto, mediante ALTER TYPE DROP METHOD.

Existen varias formas de identificar el cuerpo de método que debe eliminarse:

METHOD *nombre-método*
Identifica el método en concreto que se debe eliminar y sólo es válido si hay exactamente una instancia de método con el nombre *nombre-método* y el tipo de sujeto *nombre-tipo*. El método así identificado puede tener un número cualquiera de parámetros. Si no existe ningún método con este nombre para el tipo *nombre-tipo*, se produce un error (SQLSTATE 42704). Si existe más de una instancia específica del método para el tipo de datos mencionado, se produce un error (SQLSTATE 42725).

METHOD *nombre-método* **(***tipo-datos***,...)**
Proporciona la signatura del método, que identifica de manera exclusiva el método que se debe eliminar. El algoritmo de selección de métodos no se utiliza.

nombre-método
Es el nombre del método que se debe eliminar correspondiente al tipo especificado. El nombre debe ser un identificador no calificado.

(*tipo-datos***,...)**
Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE FUNCTION o ALTER TYPE, en las posiciones correspondientes la especificación de método. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la instancia de método específica que se debe eliminar.

Si el tipo-datos no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL.

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede codificarse un conjunto vacío de paréntesis para indicar que estos atributos deben ignorarse al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

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

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que 0 <n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún método que tenga la signatura especificada para el tipo de datos indicado, se produce un error (SQLSTATE 42883).

FOR *nombre-tipo*
Designa el tipo para el cual se debe eliminar el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de tipo no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados.

RESTRICT
La palabra clave RESTRICT aplica la norma que establece que el método no se eliminará si existe alguna de las dependencias siguientes:

- Otra rutina tiene el método como origen.
- Una vista utiliza el método.
- Un activador utiliza el método.

RESTRICT es el comportamiento por omisión.

SPECIFIC METHOD *nombre-específico*
Identifica el método específico que se debe eliminar, mediante un nombre especificado o un nombre que se toma por omisión al ejecutar CREATE TYPE o ALTER TYPE. En el SQL dinámico, si el nombre específico no está calificado, se utiliza el registro especial CURRENT SCHEMA como calificador para un nombre específico no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para un nombre específico no calificado. El nombre específico debe identificar un método; de lo contrario se produce un error (SQLSTATE 42704).

RESTRICT
La palabra clave RESTRICT aplica la norma que establece que el método no se eliminará si existe alguna de las dependencias siguientes:

- Otra rutina tiene el método como origen.
- Una vista utiliza el método.
- Un activador utiliza la función.

RESTRICT es el método por omisión.

Otros objetos pueden depender de un método. Todas estas dependencias deben eliminarse para poder eliminar el método, con la excepción de los paquetes, que se marcarán como no operativos si la eliminación es efectiva. El intento de eliminar un método con dichas dependencias dará como resultado un error (SQLSTATE 42893).

Si el método se puede eliminar, se eliminará.

Los paquetes dependientes del método específico que se está eliminando se marcarán como no operativos. Dichos paquetes no se vuelven a enlazar implícitamente. Deben volverse a enlazar mediante el mandato BIND o REBIND, o deben volverse a preparar mediante el mandato PREP.

Si el método específico que está eliminándose altera temporalmente a otro método, se invalidarán todos los paquetes que dependen del método alterado temporalmente —y que dependen de los métodos que alteran temporalmente a este método en los supertipos del método específico que está eliminándose—.

NICKNAME *apodo*
Identifica el apodo que va a eliminarse. El apodo debe aparecer en la lista del catálogo (SQLSTATE 42704). El apodo se suprime de la base de datos.

Toda la información acerca de las columnas e índices asociados con el apodo se elimina del catálogo. Se elimina cualquier tabla de consultas materializadas que dependa del apodo. Se elimina cualquier especificación de índice que es dependiente del apodo. Las vistas que dependen del apodo se marcan como no operativas. Se invalidan los paquetes que dependen de las especificaciones de índice o de las vistas no operativas que se han eliminado. No afecta a la tabla de fuente de datos al que el apodo hace referencia.

Si una función o método de SQL depende de un apodo, ese apodo no se puede descartar (SQLSTATE 42893).

DATABASE PARTITION GROUP *nombre-grupo-particiones-base-de-datos*
Identifica el grupo de particiones de base de datos que va a eliminarse. El parámetro *nombre-grupo-particiones-base-de-datos* debe identificar un grupo de particiones de base de datos que se haya descrito en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

La eliminación de un grupo de particiones de base de datos elimina todos los espacios de tabla que se han definido en el grupo de particiones de base de datos. Todos los objetos de base de datos que existan con dependencias en las tablas del espacio de tablas (por ejemplo, paquetes, restricciones de referencia, etcétera) se descartan o invalidan (según sea adecuado), y las vistas y los activadores dependientes no estarán operativos.

Los grupos de particiones de base de datos definidos por el sistema no pueden eliminarse (SQLSTATE 42832).

Si se emite una sentencia DROP DATABASE PARTITION GROUP para un grupo de particiones de base de datos que actualmente está sometiéndose a una redistribución de datos, la operación de eliminación del grupo de particiones de base de datos no se realiza satisfactoriamente y se devuelve un error (SQLSTATE 55038). Sin embargo, puede eliminarse un grupo de particiones de base de datos redistribuido parcialmente. Un grupo de particiones de base de datos puede redistribuirse parcialmente si no se completa la ejecución de un mandato REDISTRIBUTE DATABASE PARTITION GROUP. Esto puede suceder si se interrumpe a consecuencia de un error o por haberse emitido un mandato FORCE APPLICATION ALL. (Para un grupo de particiones de base de datos redistribuido parcialmente, el REBALANCE_PMAP_ID del catálogo SYSCAT.DBPARTITIONGROUPS no es –1.)

PACKAGE *nombre-esquema.id-paquete*
Identifica el paquete que se debe eliminar. Si no se especifica un nombre de esquema, el esquema por omisión califica implícitamente el identificador del paquete. El nombre de esquema y el identificador de paquete, junto con el identificador de versión especificado implícita o explícitamente, deben

identificar un paquete que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el paquete especificado. Si el paquete que está eliminándose es el único paquete que *nombre-esquema.id-paquete* identifican (es decir, si no existen otras versiones), también se suprimen todos los privilegios para el paquete.

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

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

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

PROCEDURE
Identifica una instancia de un procedimiento almacenado que no se puede eliminar. La instancia del procedimiento especificado debe ser un procedimiento almacenado descrito en el catálogo.

Hay varias maneras disponibles de identificar la instancia de procedimiento:

PROCEDURE *nombre-procedimiento*
Identifica el procedimiento en particular que se debe eliminar y sólo es válido si hay exactamente una instancia de procedimiento con el *nombre-procedimiento* en el esquema. El procedimiento identificado de esta manera puede tener cualquier número de parámetros definido. Si no existe ningún procedimiento con este nombre en el esquema nombrado o implícito, se genera un error (SQLSTATE 42704). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Si hay más de una instancia específica del procedimiento en el esquema mencionado o implícito, se devuelve un error (SQLSTATE 42725).

RESTRICT
La palabra clave RESTRICT impide que se descarte el procedimiento si una definición de activador, una función de SQL o un método de SQL contiene una sentencia CALL con el nombre del procedimiento. RESTRICT es el comportamiento por omisión.

PROCEDURE *nombre-procedimiento* **(***tipo-datos***,...)**
Proporciona la signatura del método, que identifica de manera exclusiva el procedimiento que se debe eliminar. El algoritmo de selección de procedimientos no se utiliza.

nombre-procedimiento
Proporciona el nombre del procedimiento que se debe eliminar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En

las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados.

(tipo-datos, ...)
Deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la instancia específica del procedimiento que se debe eliminar.

Si *tipo-datos* no está calificado, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede codificarse un conjunto vacío de paréntesis para indicar que estos atributos deben ignorarse al buscar coincidencias de un tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION.

No es necesario que un tipo de FLOAT(n) coincida con el valor definido para n puesto que 0 <n<25 significa REAL y 24<n<54 significa DOUBLE. La coincidencia se produce basándose en si el tipo es REAL o DOUBLE.

Si no existe ningún procedimiento con la signatura especificada en el esquema indicado o implícito, se devuelve un error (SQLSTATE 42883).

SPECIFIC PROCEDURE *nombre-específico*
Identifica el procedimiento almacenado en particular que se debe eliminar, utilizando el nombre específico que se ha especificado o que ha tomado por omisión en el momento de la creación del procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. El *nombre-específico* debe identificar una instancia del procedimiento específico en el esquema nombrado o implícito; de lo contrario, se genera un error (SQLSTATE 42704).

RESTRICT
La palabra clave RESTRICT impide que se descarte el procedimiento si una definición de activador, una función de SQL o un método de SQL contiene una sentencia CALL con el nombre del procedimiento. RESTRICT es el comportamiento por omisión.

No es posible eliminar un procedimiento que se encuentre en el esquema SYSIBM, SYSFUN o SYSPROC (SQLSTATE 42832).

SCHEMA *nombre-esquema* **RESTRICT**
Identifica el esquema en particular que se debe eliminar. El *nombre-esquema* debe identificar un esquema que está descrito en el catálogo (SQLSTATE

42704). La palabra clave RESTRICT impone la norma de que no puede haber ningún objeto definido en el esquema especificado para que se suprima de la base de datos (SQLSTATE 42893).

SEQUENCE *nombre-secuencia*
Identifica la secuencia en particular que se debe eliminar. El *nombre-secuencia*, junto con el nombre de esquema implícito o explícito, debe identificar una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre en el esquema especificado explícita o implícitamente, se produce un error (SQLSTATE 42704).

La opción RESTRICT, que es el valor por omisión, evita que la secuencia se elimine si existe alguna de las dependencias siguientes:

- Existe un activador que hace que una expresión NEXT VALUE o PREVIOUS VALUE en el activador especifique la secuencia (SQLSTATE 42893).
- Existe una función de SQL o un método de SQL que hace que una expresión de NEXT VALUE del cuerpo de la rutina especifique la secuencia (SQLSTATE 42893).

SERVER *nombre-servidor*
Identifica la fuente de datos cuya definición se debe eliminar del catálogo. El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). Se elimina la definición de la fuente de datos.

Se eliminan todos los apodos para tablas y vistas que residen en la fuente de datos. Se elimina cualquier especificación de índice dependiente de estos apodos. También es eliminada cualquier correlación de funciones definida por el usuario, correlación de tipos definida por el usuario y correlación de usuarios que es dependiente de la definición de servidor. Se invalidan todos los paquetes dependientes de la definición de servidor, correlaciones de función, apodos y especificaciones de índices eliminados.

TABLE *nombre-tabla*
Identifica la tabla base, la tabla temporal declarada o el apodo que se debe descartar. El *nombre-tabla* debe identificar una tabla que esté descrita en el catálogo, si se trata de una tabla temporal declarada, el *nombre-tabla* debe estar calificado por el nombre de esquema SESSION y existir en la aplicación (SQLSTATE 42704). Las subtablas de una tabla con tipo dependen de sus supertablas. Deben eliminarse todas las subtablas antes de poder eliminar una supertabla (SQLSTATE 42893). La tabla especificada se suprime de la base de datos.

Se eliminan todos los índices, claves primarias, claves foráneas, restricciones de comprobación, tablas de consultas materializadas y tablas de etapas que hacen referencia a la tabla. Todas las vistas y activadores que hacen referencia a la tabla pasan a ser no operativos. (Esto incluye a la tabla a la que se hace referencia en la cláusula ON de la sentencia CREATE TRIGGER y a todas las tablas a las que se hace referencia dentro de las sentencias de SQL activadas.) Todos los paquetes que dependen de cualquier objeto eliminado o marcado como no operativo se invalidarán. Esto incluye los paquetes que dependan de cualquier supertabla por encima de la subtabla en la jerarquía. Las columnas de referencia para las que la tabla eliminada se defina como ámbito de la referencia se quedan sin ámbito.

Los paquetes no dependen de tablas temporales declaradas, y por tanto no se invalidan cuando se elimina una tabla esa clase.

Todos los archivos que estén enlazados mediante columnas DATALINK se desenlazan. La operación de desenlace se lleva a cabo de manera asíncrona, por lo que es posible que los archivos no estén inmediatamente disponibles para otras operaciones.

En un sistema federado, puede descartarse una tabla remota que se ha creado utilizando un DDL transparente. Si se descarta una tabla remota también se descarta el apodo asociado a la misma y se invalidan los paquetes que dependen de ese apodo.

Cuando se elimina una subtabla de una jerarquía de tablas, las columnas asociadas a la subtabla ya no son accesibles aunque sigan teniéndose en cuenta con respecto a los límites del número de columnas y tamaño de la fila. Cuando se elimina una subtabla, todas sus filas se suprimen en las supertablas. Ello puede provocar la activación de activadores o de restricciones de integridad referencial definidos para las supertablas.

Cuando se elimina una tabla temporal declarada, y su creación fue anterior a la unidad de trabajo activa o punto de salvaguarda, se inhabilita la tabla y la aplicación no puede acceder a ella. Sin embargo, la tabla seguirá reservando parte del espacio de su espacio de tablas y evitará que el espacio de tablas USER TEMPORARY se elimine o que el grupo de particiones de base de datos del espacio de tablas USER TEMPORARY se redistribuya hasta que se confirme la unidad de trabajo o hasta que finalice el punto de salvaguarda. La eliminación de una tabla temporal declarada hace que se destruyan los datos de la tabla, con independencia de si DROP se confirma o retrotrae.

Una tabla no podrá eliminarse si tiene el atributo RESTRICT ON DROP.

TABLE HIERARCHY *nombre-tabla-raíz*
Identifica la jerarquía de la tabla con tipo que se debe eliminar. El *nombre-tabla-raíz* debe identificar una tabla con tipo que es tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR). La tabla con tipo identificada mediante el *nombre-tabla-raíz* y todas sus subtablas se suprimen de la base de datos.

Se eliminan todos los índices, tablas de consultas materializadas, tablas de etapas, claves primarias, claves foráneas y restricciones de comprobación que hacen referencia a las tablas eliminadas. Todas las vistas y activadores que hacen referencia a las tablas eliminadas se hacen no operativas. Todos los paquetes que dependen de cualquier objeto eliminado o marcado como no operativo se invalidarán. Todas las columnas de referencia para las que una de las tablas eliminadas se define como ámbito de la referencia se quedan sin ámbito.

Todos los archivos que estén enlazados mediante columnas DATALINK se desenlazan. La operación de desenlace se lleva a cabo de manera asíncrona, por lo que es posible que los archivos no estén inmediatamente disponibles para otras operaciones.

A diferencia de la eliminación de una subtabla individual, la eliminación de la jerarquía de tablas no produce la activación de los activadores de supresión en ninguna tabla de la jerarquía ni registra en el archivo de anotaciones las filas suprimidas.

TABLESPACE o **TABLESPACES** *nombre-espacio-tabla*
Identifica los espacios de tablas que se deben eliminar. El *nombre-espacio-tabla* debe identificar un espacio de tablas que se haya descrito en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

Los espacios de tabla no se eliminarán (SQLSTATE 55024) si existe alguna tabla que almacena, como mínimo, una de sus partes en un espacio de tablas que está eliminándose y si tiene una o más de sus partes en otro espacio de tablas que no está eliminándose (estas tablas deben eliminarse primero) o si alguna tabla que reside en el espacio de tablas tiene el atributo RESTRICT ON DROP. Los espacios de tablas del sistema no se pueden eliminar (SQLSTATE 42832). Un espacio de tablas temporal del sistema (SYSTEM TEMPORARY) no se puede eliminar (SQLSTATE 55026) si es el único espacio de tablas temporal que existe en la base de datos. Un espacio de tablas temporal del usuario (USER TEMPORARY) no se puede eliminar si en él existe una tabla temporal declarada (SQLSTATE 55039). Aunque se haya eliminado una tabla temporal declarada, se considera que el espacio de tablas USER TEMPORARY todavía está en uso hasta que se confirme la unidad de trabajo de la sentencia DROP TABLE .

La eliminación de un espacio de tablas elimina todos los objetos definidos en el espacio de tablas. Todos los objetos de base de datos existentes con dependencias en el espacio de tablas como, por ejemplo, paquetes, restricciones de referencia, etc. se eliminan o invalidan (lo que sea adecuado) y las vistas y activadores dependientes pasan a estar no operativos.

No se suprimen los contenedores creados por un usuario. Se suprime cualquier directorio en la vía de acceso del nombre de contenedor que se haya creado por el gestor de bases de datos en CREATE TABLESPACE. Se suprimen todos los contenedores que están por debajo del directorio de bases de datos. Cuando DROP TABLESPACE se confirma, los contenedores de archivos DMS o los contenedores SMS correspondientes al espacio de tablas especificados se suprimen, si es posible. Si los contenedores no se pueden suprimir (porque los conserva abiertos otro agente, por ejemplo), los archivos se truncan a una longitud cero. Una vez terminados todos los contenedores, o cuando se emite el mandato DEACTIVATE DATABASE, estos archivos de longitud cero se suprimen.

TRANSFORM ALL FOR *nombre-tipo*
Indica que se deben eliminar todos los grupos de transformación definidos para el tipo de datos definido por el usuario, *nombre-tipo*. Las funciones de transformación referenciadas en estos grupos no se eliminan. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704).

Si no hay transformaciones definidas para *nombre-tipo*, se emite un error (SQLSTATE 42740).

DROP TRANSFORM es lo opuesto de CREATE TRANSFORM. Hace que las funciones de transformación asociadas a determinados grupos, para un tipo de datos proporcionado, pasen a estar no definidas. Las funciones que estaban asociadas a estos grupos siguen existiendo y todavía pueden invocarse explícitamente, pero ya no tienen el atributo de transformación y no se invocan implícitamente para intercambiar valores con el entorno del lenguaje principal.

El grupo de transformación no se elimina si existe una función (o método) definida por el usuario, escrita en un lenguaje distinto del SQL, que depende de una de las funciones de transformación del grupo definidas para el tipo definido por el usuario *nombre-tipo* (SQLSTATE 42893). Dicha función depende de la función de transformación asociada al grupo de transformación

referenciado que se ha definido para el tipo *nombre-tipo*. Los paquetes que dependen de una función de transformación asociada al grupo de transformación referenciado se marcan como no operativos.

TRANSFORMS *nombre-grupo* **FOR** *nombre-tipo*
Indica que debe eliminarse el grupo de transformación especificado para el tipo de datos definido por el usuario *nombre-tipo*. Las funciones de transformación referenciadas en este grupo no se eliminan. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados. El *nombre-tipo* debe identificar un tipo definido por el usuario que esté descrito en el catálogo (SQLSTATE 42704), y el *nombre-grupo* debe identificar un grupo de transformación existente para *nombre-tipo*.

TRIGGER *nombre-activador*
Identifica el activador que se debe eliminar. El *nombre-activadores* debe identificar un activador que se haya descrito en el catálogo (SQLSTATE 42704). Se suprime el activador especificado.

La eliminación de activadores hace que algunos paquetes se marquen como no válidos.

Si *nombre-activador* especifica un activador INSTEAD OF en una vista, puede que otro activador dependa de éste a través de una actualización que debe realizarse para la vista.

TYPE *nombre-tipo*
Identifica el tipo definido por el usuario que se debe eliminar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de objetos no calificados. Para un tipo estructurado, también se elimina el tipo de referencia asociado. El *nombre-tipo* debe identificar un tipo definido por el usuario descrito en el catálogo. Si se especifica DISTINCT, entonces el *nombre-tipo* debe identificar un tipo diferenciado que esté descrito en el catálogo.

RESTRICT
El tipo no se elimina (SQLSTATE 42893) si se da alguna de las condiciones siguientes:

- El tipo se utiliza como tipo de una columna de una tabla o vista.
- El tipo tiene un subtipo.
- El tipo es un tipo estructurado que se utiliza como tipo de datos de una tabla con tipo o vista con tipo.
- El tipo es un atributo de otro tipo estructurado.
- Existe una columna de una tabla cuyo tipo puede contener una instancia de *nombre-tipo*. Esto puede ocurrir si *nombre-tipo* es el tipo de la columna o se utiliza en otro lugar de la jerarquía de tipos asociada de la columna. Es decir, para cualquier tipo T, no se puede eliminar T si existe una columna de una tabla cuyo tipo utiliza, directa o indirectamente, *nombre-tipo*.
- El tipo es el tipo destino de una columna de tipo de referencia de la tabla o vista o un atributo de tipo de referencia de otro tipo estructurado.

- El tipo, o una referencia al tipo, es un tipo de parámetro o un tipo de valor de retorno de una función o un método.
- El tipo, o una referencia al tipo, se utiliza en el cuerpo de una función o método SQL, pero no es un tipo de parámetro ni un tipo de valor de retorno.
- El tipo se utiliza en una restricción de comprobación, un activador, una definición de vista o en una extensión de índice.

Si no se especifica RESTRICT, el comportamiento es el mismo que RESTRICT, excepto por las funciones y los métodos que utilizan el tipo.

Funciones que utilizan el tipo: Si puede eliminarse el tipo definido por el usuario, para cada función F (con el nombre específico SF), que tenga parámetros o un valor de retorno del tipo que se elimina, o una referencia al tipo que se elimina, la siguiente sentencia DROP FUNCTION se ejecuta eficazmente:

```
DROP SPECIFIC FUNCTION SF
```

Es posible que esta sentencia también elimine en cascada las funciones dependientes. Si todas estas funciones también están en la lista que se debe eliminar debido a una dependencia del tipo definido por el usuario, la eliminación del tipo definido por el usuario será satisfactoria (de lo contrario, falla con SQLSTATE 42893).

Métodos que utilizan el tipo: Si puede eliminarse el tipo definido por el usuario, para cada método M de tipo T1 (con el nombre específico SM), que tenga parámetros o un valor de retorno del tipo que se elimina, o una referencia al tipo que se elimina, las siguientes sentencias se ejecutan eficazmente:

```
DROP SPECIFIC METHOD SM
ALTER TYPE T1 DROP SPECIFIC METHOD SM
```

La existencia de objetos que dependen de estos métodos puede dar lugar a que la operación DROP TYPE no se ejecute satisfactoriamente.

Se invalidan todos los paquetes que dependen de métodos que se han definido en supertipos del tipo que está eliminándose y que pueden elegirse para la alteración temporal.

TYPE MAPPING *nombre-correlación-tipos*
Identifica la correlación de tipos de datos definida por el usuario que se debe eliminar. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704). La correlación de tipos de datos se suprime de la base de datos.

No se eliminan objetos adicionales.

USER MAPPING FOR *nombre-autorización* | **USER SERVER** *nombre-servidor*
Identifica la correlación de usuarios que se debe eliminar. Esta correlación asocia un nombre de autorización que se utiliza para acceder a la base de datos federada con un nombre de autorización que se utiliza para acceder a la fuente de datos. Se identifica el primero de estos dos nombres de autorización mediante el *nombre-autorización* o se hace referencia mediante el registro especial USER. El *nombre-servidor* identifica la fuente de datos que el segundo nombre de autorización utiliza para tener acceso.

El *nombre-autorización* debe aparecer en la lista del catálogo (SQLSTATE 42704). El *nombre-servidor* debe identificar una fuente de datos que está descrita en el catálogo (SQLSTATE 42704). Se suprime la correlación del usuario.

No se eliminan objetos adicionales.

VIEW *nombre-vista*
Identifica la vista que se debe eliminar. El *nombre-vista* debe identificar una vista que esté descrita en el catálogo (SQLSTATE 42704). Las subvistas de una vista con tipo dependen de sus supervistas. Deben eliminarse todas las subvistas antes de poder eliminar una supervista (SQLSTATE 42893).

Se suprime la vista especificada. La definición de cualquier vista o activador que sea dependiente directa o indirectamente de esta vista se marca como no operativa. Se elimina cualquier tabla de consultas materializadas o tabla de etapas que dependa de cualquier vista que se ha marcado como no operativa. Se invalidará cualquier paquete que dependa de una vista que se elimine o se marque como no operativa. Esto incluye los paquetes que dependan de cualquier supervista por encima de la subvista en la jerarquía. Las columnas de referencia para las que la vista eliminada se defina como ámbito de la referencia se quedan sin ámbito.

VIEW HIERARCHY *nombre-vista-raíz*
Identifica la jerarquía de vistas con tipo que se debe eliminar. El *nombre-vista-raíz* debe identificar una vista con tipo que es vista raíz de la jerarquía de vistas con tipo (SQLSTATE 428DR). Se suprimen de la base de datos la vista con tipo identificada mediante el *nombre-vista-raíz* y todas sus subvistas.

La definición de cualquier vista o activador que sea directa o indirectamente dependiente de cualquiera de las vista eliminadas se marca como no operativa. Cualquier paquete que dependa de cualquier vista o activador que se elimine o se marque como no operativo será invalido. Cualquier columna de referencia para la que una vista eliminada o vista marcada como no operativa se define como ámbito de la referencia se queda sin ámbito.

WRAPPER *nombre-reiniciador*
Identifica el reiniciador que se debe eliminar. El *nombre-reiniciador* debe identificar un reiniciador que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el reiniciador.

Se eliminan todas las definiciones de servidor, correlaciones de función definidas por el usuario y correlaciones de tipo de datos definidas por el usuario que dependen del reiniciador. También se eliminan todas las correlaciones de función definidas por el usuario, apodos, correlaciones de tipo de datos definidas por el usuario y correlaciones de usuario que son dependientes de las definiciones de servidor. Se elimina cualquier especificación de índice dependiente de estos apodos eliminados y se marca como no operativa cualquier vista dependiente de estos apodos. Se invalidan todos los paquetes dependientes de los objetos eliminados y vistas no operativas.

Normas:

Dependencias: En la Tabla 10 en la página 529 se muestran las dependencias que tienen los objetos entre sí. No todas las dependencias se graban explícitamente en el catálogo. Por ejemplo, no existe ningún registro de las restricciones de las que depende un paquete. Se muestran cuatro tipos de dependencias distintas:

R Semántica de restricción. El objeto principal no puede eliminarse mientras exista el objeto que depende del él.

C Semántica en cascada. La eliminación del objeto principal hace que el objeto que depende de él (objeto dependiente) se elimine también. Sin embargo, si el objeto dependiente no puede eliminarse debido a que tiene una dependencia de restricción en otro objeto, la eliminación del objeto principal fallará.

X Semántica de no operativo. La eliminación del objeto principal hace que el objeto que depende de él pase a estar no operativo. Permanece no operativo hasta que un usuario lleva a cabo una acción explícita.

A Semántica de invalidación/revalidación automática. La eliminación del objeto principal hace que el objeto que depende del mismo pase a ser no válido. El gestor de bases de datos intenta revalidar el objeto no válido.

Un paquete que una función o un método utiliza, o que utiliza un procedimiento que se llama directa o indirectamente desde una función o desde un método, sólo volverá a validarse automáticamente si la rutina se ha definido como MODIFIES SQL DATA. Si la rutina no es MODIFIES SQL DATA, se devuelve un error (SQLSTATE 56098).

Algunos objetos y parámetros de la sentencia DROP no se muestran en Tabla 10 en la página 529 porque darían como resultado columnas o filas en blanco:

- Las sentencias EVENT MONITOR, PACKAGE, PROCEDURE, SCHEMA, TYPE MAPPING y USER MAPPING DROP no tienen dependencias de objeto.
- No tienen dependencias de sentencia DROP los alias, agrupaciones de almacenamiento intermedio, claves de particionamiento, privilegios y tipos de objetos de procedimiento.
- Una sentencia DROP SERVER, DROP FUNCTION MAPPING, o DROP TYPE MAPPING en una unidad de trabajo dada (UOW) no puede procesarse bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o vista dentro de esta fuente de datos (SQLSTATE 55006).
  - La sentencia hace referencia a una categoría de una fuente de datos (por ejemplo, todas las fuentes de datos para un tipo específico y versión) y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o vista dentro de estas fuentes de datos (SQLSTATE 55006).

Tabla 10. Dependencias

Tipo de objeto → / Sentencia ↓	CONSTRAINT	FUNCTION	FUNCMAPPING	INDEX	INDEXTENSION	METHOD	NICKNAME	NODEGROUP	PACKAGE[31]	SERVER	TABLE	TABLESPACE	TRIGGER	TYPE	TYPEMAPPING	USERMAPPING	VIEW
ALTER FUNCTION	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER METHOD	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER NICKNAME, modificación del nombre local o el tipo local	R[33]	R	-	-	-	R	-	-	A	-	R	-	-	-	-	-	R
ALTER NICKNAME, modificación de una opción de columna o una opción de apodo	-	-	-	-	-	-	-	-	A	-	R	-	-	-	-	-	-
ALTER NICKNAME, adición, modificación o eliminación de una restricción	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER PROCEDURE	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER SERVER	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER TABLE DROP CONSTRAINT	C	-	-	-	-	-	-	-	A[1]	-	-	-	-	-	-	-	-
ALTER TABLE DROP PARTITIONING KEY	-	-	-	-	-	-	-	R[20]	A[1]	-	-	-	-	-	-	-	-
ALTER TYPE ADD ATTRIBUTE	-	-	-	-	R	-	-	-	A[23]	-	R[24]	-	-	-	-	-	R[14]
ALTER TYPE ALTER METHOD	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
ALTER TYPE DROP ATTRIBUTE	-	-	-	-	R	-	-	-	A[23]	-	R[24]	-	-	-	-	-	R[14]
ALTER TYPE ADD METHOD	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALTER TYPE DROP METHOD	-	-	-	-	-	R[27]	-	-	-	-	-	-	-	-	-	-	-
CREATE METHOD	-	-	-	-	-	-	-	-	A[28]	-	-	-	-	-	-	-	-
CREATE TYPE	-	-	-	-	-	-	-	-	A[29]	-	-	-	-	-	-	-	-
DROP ALIAS	-	R	-	-	-	-	-	-	A[3]	-	R[3]	-	X[3]	-	-	-	X[3]
DROP BUFFERPOOL	-	-	-	-	-	-	-	-	-	-	-	R	-	-	-	-	-
DROP DATABASE PARTITION GROUP	-	-	-	-	-	-	-	-	-	-	-	C	-	-	-	-	-
DROP FUNCTION	R	R[7]	R	-	R	R[7]	-	-	X	-	R	-	R	-	-	-	R
DROP FUNCTION MAPPING	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-
DROP INDEX	R	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	R[17]
DROP INDEX EXTENSION	-	R	-	R	-	-	-	-	-	-	-	-	-	-	-	-	-

Tabla 10. Dependencias (continuación)

Tipo de objeto → / Sentencia ↓	CONSTRAINT	FUNCTION	FUNCMAPPING	INDEX	INDEXTENSION	METHOD	NICKNAME	NODEGROUP	PACKAGE[31]	SERVER	TABLE	TABLESPACE	TRIGGER	TYPE	TYPEMAPPING	USERMAPPING	VIEW
DROP METHOD	R	R[7]	R	-	R	R	-	-	X/A[30]	-	R	-	R	-	-	-	R
DROP NICKNAME	-	R	-	C	-	R	-	-	A	-	C[11]	-	-	-	-	-	X[16]
DROP PROCEDURE	-	R[7]	-	-	-	R[7]	-	-	A	-	-	-	R	-	-	-	-
DROP SEQUENCE	-	R	-	-	-	R	-	-	A	-	-	-	R	-	-	-	-
DROP SERVER	-	C[21]	C[19]	-	-	-	C	-	A	-	-	-	-	-	C[19]	C	-
DROP TABLE[32]	C	R	-	C	-	-	-	-	A[9]	-	RC[11]	-	X[16]	-	-	-	X[16]
DROP TABLE HIERARCHY	C	R	-	C	-	-	-	-	A[9]	-	RC[11]	-	X[16]	-	-	-	X[16]
DROP TABLESPACE	-	-	-	C[6]	-	-	-	-	-	-	CR[6]	-	-	-	-	-	-
DROP TRANSFORM	-	R	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
DROP TRIGGER	-	-	-	-	-	-	-	-	A[1]	-	-	-	X[26]	-	-	-	-
DROP TYPE	R[13]	R[5]	-	-	R	-	-	-	A[12]	-	R[18]	-	R[13]	R[4]	-	-	R[14]
DROP VIEW	-	R	-	-	-	-	-	-	A[2]	-	-	-	X[16]	-	-	-	X[15]
DROP VIEW HIERARCHY	-	R	-	-	-	-	-	-	A[2]	-	-	-	X[16]	-	-	-	X[16]
DROP WRAPPER	-	-	C	-	-	-	-	-	-	C	-	-	-	-	C	-	-
REVOKE un privilegio[10]	-	CR[25]	-	-	-	CR[25]	-	-	A[1]	-	CX[8]	-	X	-	-	-	X[8]

1 Esta dependencia está implícita en la dependencia de una tabla con estas restricciones, activadores o clave de particionamiento.

2 Si un paquete tiene una sentencia INSERT, UPDATE o DELETE que actúa en una vista, el paquete tiene una utilización de inserción, actualización o supresión en la tabla base principal de la vista. En el caso de UPDATE, el paquete tiene una utilización de actualización en cada columna de la tabla base principal que se modifica por UPDATE.

Si un paquete tiene una sentencia que actúa en una vista con tipo, la creación o eliminación de cualquier vista de la misma jerarquía de vistas invalidará el paquete.

3 Si un paquete, tabla de consultas materializadas, tabla de etapas, vista o activador utiliza un alias, éste pasa a depender del alias y del objeto al que hace referencia el alias. Si el alias está en una cadena, se crea una dependencia en cada alias de la cadena.

Los propios alias no son dependientes de nada. Es posible definir un alias de un objeto que no exista.

4 Un tipo T definido por el usuario puede depender de otro tipo B definido por el usuario si T:

- designa B como tipo de datos de un atributo
- tiene un atributo de REF(B)

- tiene B como supertipo.

5 La eliminación de un tipo de datos se propaga y elimina las funciones y métodos que hacen uso de ese tipo de datos como parámetro o tipo resultante, y los métodos definidos para el tipo de datos. La eliminación de estas funciones y métodos no se ve impedida por el hecho de sean dependientes entre sí. Sin embargo, para las funciones o métodos que utilizan el tipo de datos dentro de sus cuerpos, se aplican semánticas de restricción.

6 Eliminar un espacio de tablas o una lista de espacios de tablas hace que se eliminen todas las tablas que están totalmente contenidas dentro de este espacio de tablas o lista. Sin embargo, si una tabla se fragmenta en espacios de tablas (los índices o columnas largas en diferentes espacios de tablas) y esos espacios de tablas no están en la lista que se ha eliminado, entonces no se puede eliminar ninguno de estos espacios de tablas mientras la tabla exista.

7 Una función puede depender de otra función específica si la función dependiente nombra la función de base en una cláusula SOURCE. Una función o método puede también depender de otra función o método determinados si la rutina dependiente está escrita en SQL y utiliza la rutina base en su cuerpo. Un método externo o una función externa con un parámetro de tipo estructurado o tipo de retorno dependerá también de una o más funciones de transformación.

8 Sólo la pérdida del privilegio SELECT dará lugar a que se elimine una tabla de consultas materializadas o a que una vista se convierta en no operativa. Si la vista que se inhabilita está incluida en una jerarquía de vistas con tipo, también se inhabilitarán todas sus subvistas.

9 Si un paquete tiene una sentencia INSERT, UPDATE o DELETE que actúa en una tabla T, el paquete tiene un uso de inserción, actualización o supresión en T. En el caso de UPDATE, el paquete tiene un uso de actualización en cada columna de T que se modifica por UPDATE.

Si un paquete tiene una sentencia que actúa en una tabla con tipo, la creación o eliminación de cualquier tabla de la misma jerarquía de tablas invalidará el paquete.

10 Las dependencias no existen en el nivel de columna porque los privilegios en las columnas no se pueden revocar individualmente.

Si un paquete, un activador o una vista incluye la utilización de OUTER(*Z*) en la cláusula FROM, existe una dependencia en el privilegio SELECT en cada subtabla o subvista de *Z*. De modo similar, si un paquete, un activador o una vista incluye la utilización de DEREF(*Y*) donde *Y* es un tipo de referencia con una vista o tabla de destino *Z*, hay una dependencia del privilegio SELECT sobre cada subtabla o subvista de *Z*.

11 Una tabla de consultas materializadas depende de las tablas o apodos subyacentes que se han especificado en la selección completa de la definición de tabla.

La semántica de la cascada se aplica a las tablas de consultas materializadas dependientes.

Una subtabla depende de sus supertablas hasta la tabla raíz. Una supertabla no puede eliminarse hasta que se han eliminado todas sus subtablas.

12 Un paquete puede depender de tipos estructurados como resultado de

utilizar el predicado TYPE o la expresión de tratamiento de subtipos (TREAT *expresión* AS *tipo-datos*). El paquete depende de los subtipos de cada tipo estructurado especificado en el lado derecho del predicado TYPE o de la expresión TREAT. La eliminación o creación de un tipo estructurado que altera los subtipos de los que el paquete depende causa la invalidación.

Se invalidan todos los paquetes que dependen de métodos que se han definido en supertipos del tipo que está eliminándose y que pueden elegirse para la alteración temporal.

13 Una restricción de comprobación o un activador depende de un tipo si el tipo se utiliza en cualquier parte dentro de la restricción o del activador. No hay dependencia de los subtipos de un tipo estructurado utilizado en un predicado TYPE dentro de una restricción de comprobación o un activador.

14 Una vista depende de un tipo si el tipo se utiliza en cualquier parte dentro de la definición de vista (esto incluye el tipo de la vista con tipo). No hay dependencia de los subtipos de un tipo estructurado utilizado en un predicado TYPE dentro de una definición de vista.

15 Una subvista depende de su supervista hasta la vista raíz. No se puede eliminar una supervista hasta que se hayan eliminado todas sus subvistas. Consulte el número [16] para obtener dependencias de vista adicionales.

16 Un activador o una vista también depende de la tabla de destino o vista de destino de una operación de eliminación de referencia o función DEREF. Un activador o una vista con una cláusula FROM que incluya OUTER(*Z*) depende de todas las subtablas o subvistas de *Z* que existían en el momento de crearse el activador o la vista.

17 Una vista con tipo puede depender de la existencia de un índice de unicidad para asegurar la exclusividad de la columna de identificador de objeto.

18 Una tabla puede depender de un tipo de datos definido por el usuario (diferenciado o estructurado) porque:

- el tipo se utiliza como tipo de una columna
- el tipo se utiliza como tipo de la tabla
- el tipo se utiliza como atributo de un tipo de la tabla
- el tipo se utiliza como tipo destino de un tipo de referencia que es el tipo de una columna de la tabla o un atributo del tipo de la tabla
- el tipo es utilizado, directa o indirectamente, por un tipo que es la columna de la tabla.

19 La eliminación de un servidor produce la eliminación en cascada de las correlaciones de funciones y tipo de correlaciones creadas para ese servidor nombrado.

20 Si la clave de particionamiento se ha definido en una tabla de un grupo de particiones de base de datos de varias particiones, se necesita la clave de particionamiento.

21 Si una función de tabla dependiente OLE DB tiene ″R″ objetos dependientes (véase DROP FUNCTION), el servidor no se puede eliminar.

22 Una función o método SQL puede depender de los objetos referenciados por su cuerpo.

23 Cuando se elimina un atributo A de tipo TA de *nombre-tipo* T, las sentencias DROP siguientes son efectivas:

```
Método mutador: DROP METHOD A (TA) FOR T
Método observador: DROP METHOD A () FOR T
ALTER TYPE T
   DROP METHOD A(TA)
   DROP METHOD A()
```

24 Una tabla puede depender de un atributo de un tipo de datos estructurado definido por el usuario en los casos siguientes:

1. La tabla es una tabla con tipo que está basada en *nombre-tipo* o en cualquiera de sus subtipos.
2. La tabla tiene una columna de un tipo que, directa o indirectamente, hace referencia a *nombre-tipo*.

25 La sentencia REVOKE en un privilegio SELECT para una tabla o vista que se utiliza en el cuerpo de una función o método de SQL da lugar a que se realice un intento de eliminar el cuerpo de la función o método, si el cuerpo de la función o método definido ya no dispone del privilegio SELECT. Si se utiliza un cuerpo de función o método de este tipo en el cuerpo de una vista, activador o método, no podrá eliminarse y, como resultado de ello, la sentencia REVOQUE quedará restringida. En otro caso, la revocación se propaga y elimina esas funciones.

26 Un activador depende de un activador INSTEAD OF cuando éste modifica la vista en la que se ha definido el activador INSTEAD OF y el activador INSTEAD OF se activa.

27 No puede eliminarse una declaración de método de un método original que otros métodos alteran temporalmente (SQLSTATE -2).

28 Si el método del cuerpo de método que está creándose se ha declarado de modo que altere temporalmente a otro método, se invalidan todos los paquetes que dependen del método alterado temporalmente y de los métodos que alteran temporalmente a este método en los supertipos del método que está creándose.

29 Cuando se crea un nuevo subtipo de un tipo existente, se invalidan todos los paquetes que dependen de los métodos que se han definido en los supertipos del tipo que está creándose y que pueden elegirse para la alteración temporal (por ejemplo, no los métodos de mutación o de observación).

30 Si el método específico del cuerpo de método que está eliminándose se ha declarado de modo que altere temporalmente a otro método, se invalidan todos los paquetes que dependen del método alterado temporalmente y de los métodos que alteran temporalmente a este método en los supertipos del método específico que está eliminándose.

31 El SQL dinámico colocado en la antememoria tiene la misma semántica que los paquetes.

32 Cuando se descarta una tabla base remota utilizando la sentencia DROP TABLE, se descartan el apodo y la tabla base remota.

33 Una clave primaria o las claves exclusivas a las que una clave foránea no hace referencia no restringen la modificación de un nombre local de apodo ni del tipo local.

Notas:

- ***Compatibilidades***

  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:
    - SYNONYM puede especificarse en lugar de ALIAS
    - PROGRAM puede especificarse en lugar de PACKAGE

- Es válido eliminar una función definida por el usuario mientras se está utilizando. También, un cursor puede abrirse en una sentencia que contenga una referencia a una función definida por el usuario y mientras el cursor está abierto la función puede eliminarse sin provocar que fallen las lecturas del cursor.
- Si se está ejecutando un paquete que depende de una función definida por el usuario, no es posible que otro ID de autorización elimine la función hasta que el paquete complete su unidad de trabajo actual. En dicho momento, se elimina la función y el paquete se convierte en no operativo. La siguiente petición de este paquete dará como resultado un error indicando que el paquete debe volverse a enlazar explícitamente.
- La eliminación de un cuerpo de función (es muy diferente de eliminar la función) puede producirse mientras se está ejecutando una aplicación que necesite el cuerpo de la función. Esto puede ocasionar o no que la sentencia falle, según si el gestor de bases de datos tenga que cargar todavía el cuerpo de la función en el almacenamiento para la sentencia.
- En cualquier tabla eliminada que incluyera archivos actualmente enlazados mediante columnas DATALINK, los archivos se desenlazan y se restaurarán o se suprimirán según la definición de la columna DATALINK.
- Si una tabla que contiene una columna DATALINK se elimina (mediante DROP TABLE o DROP TABLESPACE) mientras los DB2 Data Links Managers configurados en la base de datos no se encuentran disponibles, fallará la operación (SQLSTATE 57050).
- Además de las dependencias registradas para cualquier UDF especificada explícitamente, se registran las dependencias siguientes cuando se solicitan transformaciones implícitamente:
  1. Cuando el parámetro de tipo estructurado o el resultado de una función o método solicita una transformación, se registra una dependencia para la función o método respecto a la función de transformación necesaria TO SQL o FROM SQL.
  2. Cuando una sentencia de SQL incluida en un paquete solicita una función de transformación, se registra una dependencia para el paquete respecto a la función de transformación TO SQL o FROM SQL indicada.

  Debido a que las condiciones descritas anteriormente son los únicos casos en que se registran dependencias debido a la invocación implícita de transformaciones, las funciones, métodos y paquetes son los únicos objetos que pueden tener una dependencia respecto a funciones de transformación invocadas implícitamente. En cambio, las llamadas explícitas a funciones de transformación (en vistas y activadores, por ejemplo) sí que producen las dependencias habituales de estos otros tipos de objetos respecto a funciones de transformación. Como resultado, una sentencia DROP TRANSFORM puede también fallar debido a estas dependencias de tipo "explícito" que los objetos tienen respecto a las transformaciones que están eliminando (SQLSTATE 42893).
- Debido a que los catálogos de dependencias no distinguen entre el depender de una función en calidad de transformación y el depender de una función por llamada explícita, es recomendable no escribir llamadas explícitas a funciones de transformación. En tal caso, el atributo de transformación de la función no se

puede eliminar, o los paquetes se marcan como no operativos, simplemente porque contienen invocaciones explícitas en una expresión SQL.

- Las secuencias creadas por el sistema para las columnas IDENTITY no pueden eliminarse utilizando la sentencia DROP SEQUENCE.
- Cuando se elimina una secuencia, también se eliminan todos los privilegios para la secuencia y se invalida cualquier paquete que haga referencia a la secuencia.
- Para apodos relacionales, la sentencia DROP NICKNAME de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las siguientes condiciones (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Para apodos no relacionales, la sentencia DROP NICKNAME dentro de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Una sentencia SELECT de la misma UOW ya hace referencia a un apodo al que se hace referencia en esta sentencia
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Una sentencia DROP SERVER (SQLSTATE 55006), o las sentencias DROP FUNCTION MAPPING o DROP TYPE MAPPING (SQLSTATE 55007) de una unidad de trabajo (UOW) determinada no pueden procesarse en bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos

Ejemplos:

Ejemplo 1: Elimine la tabla TDEPT.

```
DROP TABLE TDEPT
```

Ejemplo 2: Elimine la vista VDEPT.

```
DROP VIEW VDEPT
```

Ejemplo 3: El ID de autorización HEDGES intenta eliminar un alias.

```
DROP ALIAS A1
```

El alias HEDGES.A1 se elimina de los catálogos.

Ejemplo 4: Hedges intenta eliminar un alias, pero especifica T1 como el nombre-alias, cuando T1 es el nombre de una tabla existente (no el nombre de un alias).

```
DROP ALIAS T1
```

Esta sentencia falla (SQLSTATE 42809).

Ejemplo 5:

Elimine el grupo de particiones de base de datos BUSINESS_OPS. Para eliminar el grupo de particiones de base de datos, primero deben eliminarse los dos espacios de tabla (ACCOUNTING y PLANS) del grupo de particiones de base de datos.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP DATABASE PARTITION GROUP BUSINESS_OPS
```

Ejemplo 6: Pellow desea eliminar la función CENTRE, que ha creado en su esquema PELLOW, usando la signatura para identificar la instancia de la función que se debe eliminar.

```
DROP FUNCTION CENTRE (INT,FLOAT)
```

Ejemplo 7: McBride desea eliminar la función FOCUS92, que ha creado en el esquema PELLOW, utilizando el nombre específico para identificar la instancia de función que se debe eliminar.

```
DROP SPECIFIC FUNCTION PELLOW.FOCUS92
```

Ejemplo 8: Elimine la función ATOMIC_WEIGHT del esquema CHEM, donde se sabe que sólo existe una función con ese nombre.

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

Ejemplo 9: Elimine el activador SALARY_BONUS, que ha dado lugar a que los empleados de la condición especificada recibieran una bonificación en sus salarios.

```
DROP TRIGGER SALARY_BONUS
```

Ejemplo 10: Elimine el tipo de datos diferenciado denominado shoesize, si no se utiliza actualmente.

```
DROP DISTINCT TYPE SHOESIZE
```

Ejemplo 11: Elimine el supervisor de sucesos SMITHPAY.

```
DROP EVENT MONITOR SMITHPAY
```

Ejemplo 12: Elimine el esquema del Ejemplo 2 bajo CREATE SCHEMA utilizando RESTRICT. Tenga en cuenta que primero debe eliminarse la tabla llamada PART.

```
  DROP TABLE PART
DROP SCHEMA INVENTRY RESTRICT
```

Ejemplo 13: Macdonald desea eliminar el procedimiento DESTROY, que ha creado en el esquema EIGLER, utilizando el nombre específico para identificar la instancia de procedimiento que se debe eliminar.

```
DROP SPECIFIC PROCEDURE  EIGLER.DESTROY
```

Ejemplo 14: Elimine el procedimiento OSMOSIS del esquema BIOLOGY, donde se sabe que sólo hay un procedimiento con dicho nombre.

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

Ejemplo 15: El usuario SHAWN ha utilizado un ID de autorización para acceder a la base de datos federada y otro para acceder a la base de datos en la fuente de datos Oracle llamada ORACLE1. Se ha creado una correlación entre las dos autorizaciones, pero SHAWN ya no necesita acceder a la fuente de datos. Eliminar la correlación.

```
DROP USER MAPPING FOR  SHAWN SERVER ORACLE1
```

Ejemplo 16: Se ha suprimido un índice de una tabla de fuente de datos al que un apodo hace referencia. Elimine la especificación de índice que se ha creado para dejar que optimizador conozca este índice.

```
DROP INDEX INDEXSPEC
```

Ejemplo 17: Eliminación del grupo de transformación MYSTRUCT1 .

```
DROP TRANSFORM MYSTRUCT1 FOR POLYGON
```

Ejemplo 18: Eliminación del método BONUS para el tipo de datos EMP en el esquema PERSONNEL.

```
DROP METHOD BONUS (SALARY DECIMAL(10,2)) FOR PERSONNEL.EMP
```

Ejemplo 19: Elimine la secuencia ORG_SEQ, con restricciones.

```
DROP SEQUENCE ORG_SEQ
```

Ejemplo 20: Se ha creado la tabla remota EMPLOYEE en un sistema federado utilizando un DDL transparente. Ya no es necesario acceder a la tabla. Descarte la tabla remota EMPLOYEE.

```
DROP TABLE EMPLOYEE
```

Ejemplo 21: Descarte la correlación de funciones BONUS_CALC y restituya la correlación de funciones por omisión (si existe).

```
DROP FUNCTION MAPPING BONUS_CALC
```

Tareas relacionadas:

- "Disabling a default function mapping" en la publicación *Federated Systems Guide*
- "Dropping a user-defined function mapping" en la publicación *Federated Systems Guide*
- "Dropping remote tables using transparent DDL" en la publicación *Federated Systems Guide*

Información relacionada:

- "CREATE TRIGGER" en la página 422
- "CREATE VIEW" en la página 466
- "CREATE FUNCTION MAPPING" en la página 268

Ejemplos relacionados:

- "dbstat.sqb -- Reorganize table and run statistics (MF COBOL)"
- "dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)"
- "tbconstr.sqC -- How to create, use, and drop constraints (C++)"

- “tbcreate.sqC -- How to create and drop tables (C++)”
- “tbtrig.sqC -- How to use a trigger on a table (C++)”
- “DbSeq.java -- How to create, alter and drop a sequence in a database (JDBC)”
- “TbConstr.java -- How to create, use and drop constraints (JDBC)”
- “TbCreate.java -- How to create and drop tables (JDBC)”
- “TbTemp.java -- How to use Declared Temporary Table (JDBC)”
- “TbTrig.java -- How to use triggers (JDBC)”
- “UDFDrop.db2 -- How to uncatalog the Java UDFs contained in UDFsrv.java ”
- “spdrop.db2 -- How to uncatalog the stored procedures contained in spserver.sqc (C)”
- “tbconstr.sqc -- How to create, use, and drop constraints (C)”
- “tbcreate.sqc -- How to create and drop tables (C)”
- “tbtemp.sqc -- How to use a declared temporary table (C)”
- “tbtrig.sqc -- How to use a trigger on a table (C)”
- “TbConstr.sqlj -- How to create, use and drop constraints (SQLj)”
- “TbCreate.sqlj -- How to create and drop tables (SQLj)”
- “TbTrig.sqlj -- How to use triggers (SQLj)”

END DECLARE SECTION

La sentencia END DECLARE SECTION marca el final de una sección de declaración de variables de lenguaje principal.

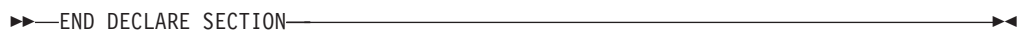
Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. No debe especificarse en REXX.

Autorización:

No se necesita.

Sintaxis:



Descripción:
La sentencia END DECLARE SECTION puede codificarse en el programa de aplicación donde pueda haber declaraciones de acuerdo a las normas del lenguaje principal. Indica el final de una sección de declaración de variables de lenguaje principal. Una sección de variables del lenguaje principal empieza con una sentencia BEGIN DECLARE SECTION.

Las sentencias BEGIN DECLARE SECTION y END DECLARE SECTION deben especificarse por pares y no pueden anidarse.

Las declaraciones de variables de lenguaje principal pueden especificarse utilizando la sentencia de SQL INCLUDE. De lo contrario, la sección de declaración de variables de lenguaje principal no debe contener ninguna sentencia que no sean declaraciones de variables de lenguaje principal.

Las variables del lenguaje principal a las que se hace referencia en las sentencias de SQL deben declararse en una sección de declaración de variables del lenguaje principal en todos los lenguajes del sistema principal que no sean REXX. Además, la declaración de cada variable debe preceder a la primera referencia a la variable.

Las variables declaradas fuera de una sección de declaración no deben tener el mismo nombre que las variables declaradas dentro de una sección de declaración.

Información relacionada:

- "BEGIN DECLARE SECTION" en la página 105

Ejemplos relacionados:

- "advsql.sqb -- How to read table data using CASE (MF COBOL)"
- "prepbind.sqb -- Precompile and bind an embedded SQL program to a database (MF COBOL)"
- "dtlob.sqc -- How to use the LOB data type (C)"
- "spclient.sqc -- Call various stored procedures (C)"
- "tut_read.sqc -- How to read tables (C)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "spclient.sqC -- Call various stored procedures (C++)"

END DECLARE SECTION

- “tut_read.sqC -- How to read tables (C++)”

EXECUTE

La sentencia EXECUTE ejecuta una sentencia de SQL preparada.

Invocación:

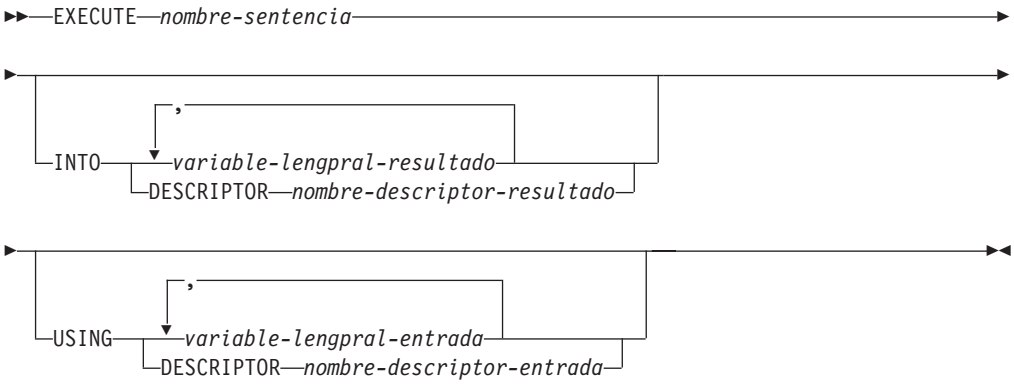
Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

Para las sentencias donde el control de autorizaciones se realiza durante la ejecución (sentencias DDL, GRANT y REVOKE), los privilegios del ID de autorización de la sentencia deben incluir aquellos necesarios para ejecutar la sentencia de SQL especificada por la sentencia PREPARE. El ID de autorización de la sentencia puede verse afectado por la opción de enlace DYNAMICRULES.

Para las sentencias donde el control de autorizaciones se realiza al preparar la sentencia (DML), no es necesaria ninguna autorización para utilizar esta sentencia.

Sintaxis:



Descripción:

nombre-sentencia
Identifica la sentencia preparada que se debe ejecutar. El *nombre-sentencia* debe identificar una sentencia que se ha preparado anteriormente, y la sentencia que se ha preparado anteriormente no puede ser una sentencia SELECT.

INTO
Introduce una lista de variables del lenguaje principal que se utilizan para recibir valores de los marcadores de parámetro de salida (signos de interrogación) en la sentencia preparada.

Para una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos OUT e INOUT para el procedimiento almacenado son marcadores de parámetro de salida. Si en la sentencia aparece algún marcador de parámetro de salida, debe especificarse la cláusula INTO (SQLSTATE 07007).

variable-lengpral-resultado, **...**
Identifica una variable del lenguaje principal que se declara en el programa de acuerdo a las normas para la declaración de variables del lenguaje principal. El número de variables debe ser el mismo que el número de

marcadores de parámetro de salida de la sentencia preparada. La variable número *n* corresponde al marcador de parámetro número *n* de la sentencia preparada. Las variables de localizador y las variables de referencia a archivo, cuando deben utilizarse, pueden proporcionarse como el destino de los marcadores de parámetro.

DESCRIPTOR *nombre-descriptor-resultado*
Identifica una SQLDA de salida que debe contener una descripción válida de las variables del lenguaje principal.

Antes de procesar la sentencia EXECUTE, el usuario debe establecer los campos siguientes en la SQLDA de entrada:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR.

Si deben incluirse datos de salida de tipo de datos estructurados o LOB, deberán existir dos entradas de SQLVAR para cada marcador de parámetro.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

USING
Introduce una lista de las variables del lenguaje principal para las que se sustituyen valores para los marcadores de parámetro de entrada (signos de interrogación) en la sentencia preparada.

Para una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos IN e INOUT para el procedimiento almacenado son los marcadores de parámetro de entrada. Para todas las demás sentencias dinámicas, todos los marcadores de parámetro son marcadores de parámetro de entrada. Si en la sentencia aparece algún marcador de parámetro de salida, debe especificarse la cláusula USING (SQLSTATE 07004).

variable-lengpral-entrada, **...**
Identifica una variable del lenguaje principal que se declara en el programa de acuerdo a las normas para la declaración de variables del lenguaje principal. El número de variables debe ser el mismo que el número de marcadores de parámetro de entrada de la sentencia preparada. La variable número *n* corresponde al marcador de parámetro número *n* de la sentencia preparada. Las variables de localizador y las variables de referencia a archivo, cuando deben utilizarse, pueden proporcionarse como la fuente de los valores de los marcadores de parámetro.

DESCRIPTOR *nombre-descriptor-entrada*
Identifica una SQLDA de entrada que debe contener una descripción válida de variables del lenguaje principal.

Antes de procesar la sentencia EXECUTE, el usuario debe establecer los campos siguientes en la SQLDA de entrada:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que 16 + SQLN*(N), donde N es la longitud de una ocurrencia SQLVAR.

Si deben incluirse datos de entrada de tipo de datos estructurados o LOB, deberán existir dos entradas de SQLVAR para cada marcador de parámetro.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

Notas:

- Antes de ejecutarse la sentencia preparada, cada marcador de parámetro de entrada se sustituye eficazmente por el valor de su correspondiente variable del lenguaje principal. Para un marcador de parámetros con tipo, los atributos de la variable de destino son aquellos especificados por la especificación CAST. Para un marcador de parámetros sin tipo, los atributos de la variable de destino se determinan de acuerdo al contexto del marcador de parámetros.

  Supongamos que V es una variable del lenguaje principal de entrada que corresponde al marcador de parámetro P. El valor de V se asigna a la variable de destino para P de acuerdo con las normas para la asignación de un valor a una columna. Por lo tanto:

  – V debe ser compatible con el destino.
  – Si V es una serie, su longitud no debe ser mayor que el atributo de longitud del destino.
  – Si V es un número, el valor absoluto de su parte correspondiente al entero no debe ser mayor que el valor absoluto máximo de la parte correspondiente a los enteros del destino.
  – Si los atributos de V no son idénticos a los atributos del destino, el valor se convierte para ajustarse a los atributos del destino.

  Cuando se ejecuta la sentencia preparada, el valor que se ha utilizado en lugar de P es el valor de la variable de destino para P. Por ejemplo, si V es CHAR(6) y el destino es CHAR(8), el valor que se ha utilizado en lugar de P es el valor de V con dos blancos.
- Para una sentencia CALL dinámica, tras haberse ejecutado la sentencia preparada, el valor que se devuelve de cada argumento OUT e INOUT se asigna a la variable del lenguaje principal que corresponde al marcador de parámetro de salida que se ha utilizado para el argumento. Para un marcador de parámetros con tipo, los atributos de la variable de destino son aquellos especificados por la especificación CAST. Para un marcador de parámetro sin tipo, los atributos de la variable de destino son los que especifica la definición del parámetro del procedimiento almacenado.

  Supongamos que V es una variable del lenguaje principal de salida que corresponde al marcador de parámetro P, que se utiliza para el argumento A de

un procedimiento almacenado. El valor de A se asigna a V de acuerdo con las normas para la recuperación de un valor de una columna. Por lo tanto:

- V debe ser compatible con A.
- Si V es una serie de caracteres, su longitud no debe ser menor que la longitud de A o el valor de A se truncará.
- Si V es un número, el valor absoluto máximo de su parte integrante no debe ser menor que el valor absoluto de la parte integrante de A.
- Si los atributos de V no son idénticos a los atributos de A, el valor de A se convierte para ajustarse a los atributos de V.

- ***Colocación en antememoria de sentencias de SQL dinámico:*** La información necesaria para ejecutar sentencias de SQL dinámico y estáticas se coloca en la antememoria del paquete de bases de datos cuando se hace referencia por primera vez a las sentencias de SQL estático o cuando se preparan por primera vez las sentencias de SQL dinámico. Esta información permanece en la antememoria del paquete hasta que se convierte en no válida, el espacio de antememoria es necesario para otra sentencia o se cierra la base de datos.

  Cuando se ejecuta o prepara una sentencia de SQL, la información del paquete relevante a la aplicación que emite la petición se carga del catálogo del sistema en la antememoria del paquete. La sección ejecutable real de la sentencia de SQL individual también se coloca en la antememoria: las secciones de SQL estático se leen desde el catálogo del sistema y se colocan en la antememoria del paquete cuando se hace referencia por primera vez a la sentencia; las secciones de SQL dinámico se colocan directamente en la antememoria tras haberse creado. Las secciones de SQL dinámico pueden crearse mediante una sentencia explícita como, por ejemplo, PREPARE o EXECUTE IMMEDIATE. Una vez creadas, las secciones de las sentencias de SQL dinámico pueden volver a crearse por medio de una preparación implícita de la sentencia que realiza el sistema si la sección original se ha suprimido por razones relacionadas con la gestión del espacio o si ha dejado de ser válida debido a que se han realizado cambios en el entorno.

  Cada sentencia de SQL se coloca en antememoria en el nivel de la base de datos y las aplicaciones pueden compartirla. Las sentencias de SQL estático se comparten entre las aplicaciones que utilizan el mismo paquete; las sentencias de SQL dinámico se comparten entre las aplicaciones que utilizan el mismo entorno de compilación y exactamente el mismo texto de sentencia. El texto de cada sentencia de SQL que una aplicación emite se coloca en antememoria localmente dentro de la aplicación para poder utilizarlo si se necesita una preparación implícita. Cada sentencia PREPARE del programa de aplicación puede poner en antememoria una sentencia. Todas las sentencias EXECUTE IMMEDIATE de un programa de aplicación comparten el mismo espacio y sólo existe una sentencia colocada en antememoria para todas estas sentencias EXECUTE IMMEDIATE al mismo tiempo. Si se emite múltiples veces la misma sentencia PREPARE o cualquier sentencia EXECUTE IMMEDIATE con una sentencia de SQL distinta cada vez, sólo se pondrá en antememoria la última sentencia para volverla a utilizar. La utilización óptima de la antememoria consiste en emitir varias sentencias PREPARE distintas una vez durante el inicio de la aplicación y en emitir posteriormente una sentencia EXECUTE o OPEN en función de las necesidades.

  Con la antememoria de sentencias de SQL dinámico, una vez creada una sentencia, puede volverse a utilizar en múltiples unidades de trabajo sin necesidad de preparar la sentencia de nuevo. Si se producen cambios en el entorno, el sistema volverá a compilar la sentencia en función de las necesidades.

  Los sucesos siguientes son ejemplos de cambios en el entorno o en objetos de datos que pueden dar lugar a que las sentencias dinámicas colocadas en

antememoria se preparen implícitamente en la siguiente petición PREPARE, EXECUTE, EXECUTE IMMEDIATE u OPEN:

- ALTER FUNCTION
- ALTER METHOD
- ALTER NICKNAME
- ALTER PROCEDURE
- ALTER SERVER
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE METHOD
- CREATE PROCEDURE
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP (todos los objetos)
- RUNSTATS en cualquier tabla o índice
- Cualquier acción que dé lugar a que una vista pase a ser no operativa
- UPDATE de estadísticas en cualquier tabla del catálogo del sistema
- SET CURRENT DEGREE
- SET PATH
- SET QUERY OPTIMIZATION
- SET SCHEMA
- SET SERVER OPTION

La lista siguiente resalta el funcionamiento que se puede esperar de las sentencias de SQL dinámico en antememoria:

- *Peticiones de PREPARE:* Las preparaciones posteriores de la misma sentencia no incurrirán en el coste de compilar la sentencia si la sección todavía es válida. Se devolverán las estimaciones de coste y cardinalidad para la sección en antememoria actual. Estos valores pueden diferir de los valores devueltos de las sentencias PREPARE anteriores para la misma sentencia de SQL. No habrá necesidad de emitir una sentencia PREPARE subsiguiente a una sentencia COMMIT o ROLLBACK.
- *Peticiones de EXECUTE:* Las sentencias EXECUTE pueden incurrir ocasionalmente en el coste de preparar implícitamente la sentencia si se ha convertido en no válida desde la sentencia PREPARE original. Si una sección se prepara implícitamente, utilizará el entorno actual y no el entorno de la sentencia PREPARE original.
- *Peticiones de EXECUTE IMMEDIATE:* Las sentencias EXECUTE IMMEDIATE posteriores para la misma sentencia no incurrirán en el coste de compilar la sentencia si la sección todavía es válida.
- *Peticiones de OPEN:* Las peticiones de OPEN para cursores definidos dinámicamente pueden incurrir ocasionalmente en el coste de preparar implícitamente la sentencia si ya no es válida desde la sentencia PREPARE

original. Si una sección se prepara implícitamente, utilizará el entorno actual y no el entorno de la sentencia PREPARE original.
- *Peticiones de FETCH:* No debe esperarse ningún cambio en el funcionamiento.
- *ROLLBACK:* Sólo se invalidarán las sentencias de SQL dinámico preparadas o implícitamente preparadas durante la unidad de trabajo afectada por la operación de retrotracción.
- *COMMIT:* Las sentencias de SQL dinámico no se invalidarán, pero se liberará cualquier bloqueo que se haya adquirido. Los cursores que no se hayan definido como cursores WITH HOLD se cerrarán y se liberarán sus bloqueos. Los cursores abiertos como WITH HOLD mantendrán sus bloqueos de paquete y de sección para proteger la sección activa durante y después del proceso de confirmación.

Si se produce une error durante una preparación implícita, se devolverá un error para la petición que provoca la preparación implícita (SQLSTATE 56098).

Ejemplos:

Ejemplo 1: En este ejemplo C, se prepara y ejecuta una sentencia INSERT con marcadores de parámetros. Las variables del lenguaje principal `h1 - h4` corresponden al formato de TDEPT.

```
strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
  .
  .
(Compruebe la ejecución satisfactoria y ponga valores en :h1, :h2, :h3, :h4)
  .
  .
EXEC SQL EXECUTE DEPT_INSERT USING :h1, :h2,
:h3, :h4;
```

Ejemplo 2: La sentencia EXECUTE utiliza una SQLDA.

```
EXECUTE S3 USING DESCRIPTOR :sqlda3
```

Ejemplo 3: Con un procedimiento almacenado para otorgar una bonificación a un empleado:

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
                             IN DEPTNO INTEGER,
                             OUT CHEQUE INTEGER,
                             INOUT BONUS DEC(6,0))
...
```

Realice una llamada dinámica al procedimiento remoto desde una aplicación en C. El procedimiento almacenado toma las variables del lenguaje principal siguientes como entrada:

- *employee*, el número de ID del empleado
- *dept*, el número del departamento
- *bonus*, la bonificación deseada para el empleado

El procedimiento almacenado devuelve los valores siguientes a las variables del lenguaje principal:

- *cheque_no*, el número de ID del cheque
- *bonus*, el importe real de la bonificación (tras realizarse cualquier ajuste necesario)

```
strcpy (s, "CALL GIVE_BONUS(?, ?, ?, ?)");
EXEC SQL PREPARE DO_BONUS FROM :s;
  .
```

```
  .
/* Comprobar la ejecución satisfactoria y colocar valores en
   :employee, :dept y :bonus */
  .
  .
EXEC SQL EXECUTE DO_BONUS INTO :cheque_no, :bonus
                          USING :employee, :dept, :bonus;
  .
  .
/* Comprobar la ejecución satisfactoria y procesar los
   valores devueltos en :cheque_no y :bonus */
```

Información relacionada:

- "Identificadores" en la publicación *Consulta de SQL, Volumen 1*
- "PREPARE" en la página 634
- "SQLDA (área de descriptores de SQL)" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbuse.sqc -- How to use a database (C)"
- "fnuse.sqc -- How to use built-in SQL functions (C)"
- "tut_use.sqc -- How to modify a database (C)"
- "udfcli.sqc -- Call a variety of types of user-defined functions (C)"
- "dbuse.sqC -- How to use a database (C++)"
- "fnuse.sqC -- How to use built-in SQL functions (C++)"
- "tut_use.sqC -- How to modify a database (C++)"
- "udfcli.sqC -- Call a variety of types of user-defined functions (C++)"
- "inpsrv.sqb -- A stored procedure using the GENERAL parameter style (MF COBOL)"

EXECUTE IMMEDIATE

La sentencia EXECUTE IMMEDIATE:

- Prepara una forma ejecutable de una sentencia de SQL a partir de una forma de sentencia de serie de caracteres.
- Ejecuta la sentencia de SQL.

EXECUTE IMMEDIATE combina las funciones básicas de las sentencias PREPARE y EXECUTE. Puede utilizarse para preparar y ejecutar sentencias de SQL que no contengan variables del lenguaje principal ni marcadores de parámetros.

Invocación:

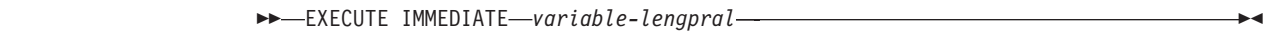
Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

Las normas de autorización son las que se han definido para la sentencia de SQL especificada.

El ID de autorización de la sentencia podría verse afectado por la opción de enlace DYNAMICRULES.

Sintaxis:



Descripción:

variable-lengpral
Debe especificarse una variable de lenguaje principal, y ésta debe identificar una variable del lenguaje principal que se haya descrito en el programa de acuerdo con las normas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres cuyo tamaño sea menor que el tamaño de sentencia máximo de 65 535. Tenga en cuenta que un CLOB(65535) puede contener una sentencia de tamaño máximo, pero un VARCHAR no. El valor de la variable del lenguaje principal identificada se denomina serie de sentencia.

La serie de caracteres de la sentencia debe ser una de las sentencias de SQL siguientes:

- ALTER
- CALL
- COMMENT
- COMMIT
- CREATE
- DECLARE GLOBAL TEMPORARY TABLE
- DELETE
- DROP
- GRANT
- INSERT
- LOCK TABLE

- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

La serie de sentencia no debe incluir marcadores de parámetros ni referencias a variables del lenguaje principal, y no debe empezar por EXEC SQL. No debe contener un terminador de sentencia, a excepción de las sentencias CREATE TRIGGER y CREATE PROCEDURE. Una sentencia CREATE TRIGGER puede contener signos de punto y coma (;) para separar las sentencias de SQL activadas. Una sentencia CREATE PROCEDURE puede contener signos de punto y coma para separar las sentencias de SQL en el cuerpo del procedimiento de SQL. El procedimiento almacenado que se especifica en una sentencia CALL no debe tener ningún parámetro OUT o INOUT (SQLSTATE 07007).

Cuando se ejecuta una sentencia EXECUTE IMMEDIATE, la serie de sentencia se analiza y se comprueba si hay errores. Si la sentencia de SQL no es válida, no se ejecuta y en la SQLCA se informa acerca de la condición de error que evita que pueda ejecutarse. Si la sentencia de SQL es válida, pero se produce un error durante su ejecución, dicha condición de error se informa a la SQLCA.

Notas:

- La puesta de sentencias en antememoria afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE.

Ejemplo:

Utilice sentencias de programa C para mover una sentencia de SQL a la variable del lenguaje principal qstring (char[80]) y prepare y ejecute la sentencia de SQL que se encuentra en la variable del lenguaje principal qstring.

```
if ( strcmp(accounts,"BIG") == 0 )
  strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO < 100");
 else
  strcpy (qstring,"INSERT INTO WORK_TABLE SELECT *
    FROM EMP_ACT WHERE ACTNO >= 100");
   .
   .
   .
EXEC SQL  EXECUTE IMMEDIATE :qstring;
```

Información relacionada:

- "Identificadores" en la publicación *Consulta de SQL, Volumen 1*
- "EXECUTE" en la página 541

Ejemplos relacionados:

- "dbuse.sqc -- How to use a database (C)"
- "dtudt.sqc -- How to create, use, and drop user-defined distinct types (C)"
- "fnuse.sqc -- How to use built-in SQL functions (C)"
- "tbconstr.sqc -- How to create, use, and drop constraints (C)"
- "tbtrig.sqc -- How to use a trigger on a table (C)"
- "tscreate.sqc -- How to create and drop buffer pools and table spaces (C)"
- "dbuse.sqC -- How to use a database (C++)"
- "dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)"
- "dtudt.sqC -- How to create, use, and drop user-defined distinct types (C++)"
- "fnuse.sqC -- How to use built-in SQL functions (C++)"
- "tbconstr.sqC -- How to create, use, and drop constraints (C++)"
- "tbtrig.sqC -- How to use a trigger on a table (C++)"
- "tscreate.sqC -- How to create and drop buffer pools and table spaces (C++)"
- "DtUdt.sqlj -- How to create, use and drop user defined distinct types (SQLj)"
- "expsamp.sqb -- Export and import tables with table data to a DRDA database (MF COBOL)"

EXPLAIN

La sentencia EXPLAIN captura información acerca del plan de acceso elegido para la sentencia explicable suministrada y coloca esta información en las tablas Explain.

Una *sentencia explicable* es una sentencia DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES o VALUES INTO SQL.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

La sentencia que se debe explicar no se ejecuta.

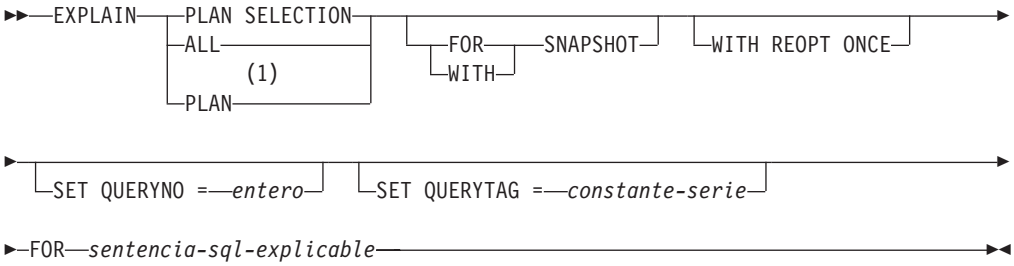
Autorización:

Las normas de autorización son las definidas para la sentencia de SQL especificada en la sentencia EXPLAIN. Por ejemplo, si se ha utilizado una sentencia DELETE como *sentencia-sql-explicable* (consulte la sintaxis de la sentencia a continuación), las normas de autorización para una sentencia DELETE se aplican cuando se explica la sentencia DELETE.

Las normas de autorización para las sentencias EXPLAIN estáticas son las normas que se aplican a las versiones estáticas de la sentencia pasada como la *sentencia-sql-explicable*. Las sentencias EXPLAIN preparadas dinámicamente utilizan las normas de autorización proporcionadas para el parámetro *sentencia-sql-explicable*.

El ID de autorización actual debe tener privilegio de inserción para las tablas Explain.

Sintaxis:



Notas:

1 Sólo se da soporte a la opción PLAN para la tolerancia de la sintaxis de sentencias EXPLAIN existentes de DB2 para MVS. No hay ninguna tabla PLAN. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

Descripción:

PLAN SELECTION
Indica que la información de la fase de selección del plan de compilación SQL se debe insertar en las tablas Explain.

ALL
La especificación de ALL es equivalente a especificar PLAN SELECTION.

PLAN
La opción PLAN proporciona la tolerancia de la sintaxis para las aplicaciones de bases de datos existentes de otros sistemas. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

FOR SNAPSHOT
Esta cláusula indica que sólo va a tomarse una instantánea de Explain y que va a colocarse en la columna SNAPSHOT de la tabla EXPLAIN_STATEMENT. No se capturará ninguna otra información de Explain distinta de la que está presente en las tablas EXPLAIN_INSTANCE y EXPLAIN_STATEMENT.

La información de la instantánea de Explain está pensada para utilizarla con Visual Explain.

WITH SNAPSHOT
Esta cláusula indica que, además de la información de Explain normal, se debe tomar una instantánea de Explain.

El funcionamiento por omisión de la sentencia EXPLAIN es reunir sólo la información de Explain normal y no la instantánea de Explain.

La información de la instantánea de Explain está pensada para utilizarla con Visual Explain.

por omisión (no se especifican FOR SNAPSHOT ni WITH SNAPSHOT)
Pone la información de Explain en las tablas Explain. No se toma ninguna instantánea para utilizarla con Visual Explain.

WITH REOPT ONCE
Esta cláusula indica que la sentencia de SQL explicable que se ha especificado se debe reoptimizar utilizando los valores para las variables del lenguaje principal, los marcadores de parámetros o los registros especiales que se han utilizado previamente para reoptimizar esta sentencia con REOPT ONCE. Las tablas Explain se rellenarán con el nuevo plan de acceso. Si el usuario tiene autorización DBADM, o la variable de registro de base de datos DB2_VIEW_REOPT_VALUES se ha establecido en YES, la tabla EXPLAIN_PREDICATE también se rellenará con los valores si se han utilizado para reoptimizar la sentencia.

SET QUERYNO = *entero*
Asocia el *entero*, por medio de la columna QUERYNO de la tabla EXPLAIN_STATEMENT, a la *sentencia-sql-explicable*. El valor entero suministrado debe ser un valor positivo.

Si no se especifica esta cláusula para una sentencia EXPLAIN dinámica, se asigna un valor por omisión de uno (1). Para una sentencia EXPLAIN estática, el valor por omisión asignado es el número de sentencia asignado por el precompilador.

SET QUERYTAG = *constante-serie*
Asocia la *constante-serie*, por medio de la columna QUERYTAG de la tabla EXPLAIN_STATEMENT, a la *sentencia-sql-explicable*. La *constante-serie* puede ser cualquier serie de caracteres de un máximo de 20 bytes de longitud. Si el valor suministrado es inferior a 20 bytes de longitud, el valor se rellena por la derecha con blancos hasta la longitud necesaria.

Si no se especifica esta cláusula para una sentencia EXPLAIN, se utilizan blancos como el valor por omisión.

FOR *sentencia-sql-explicable*

Especifica la sentencia de SQL que se debe explicar. Esta sentencia puede ser cualquier sentencia DELETE, INSERT, SELECT, SELECT INTO, UPDATE, VALUES o VALUES INTO SQL. Si la sentencia EXPLAIN está incorporada en un programa, la *sentencia-sql-explicable* puede contener referencias a las variables de lenguaje principal (estas variables deben estar definidas en el programa). De manera similar, si se está preparando EXPLAIN dinámicamente, la *sentencia-sql-explicable* puede contener marcadores de parámetros.

La *sentencia-sql-explicable* debe ser una sentencia de SQL válida que podría prepararse y ejecutarse independientemente de la sentencia EXPLAIN. No puede ser un nombre de sentencia ni una variable del lenguaje principal. Las sentencias de SQL que hacen referencia a los cursores definidos a través de CLP no son válidos para utilizarlos con esta sentencia.

Para explicar el SQL dinámico dentro de una aplicación, toda la sentencia EXPLAIN debe estar preparada dinámicamente.

Notas:

La tabla siguiente muestra la interacción de las palabras clave de instantánea y la información de Explain.

Palabra clave especificada	¿Capturar información de Explain?	¿Tomar una instantánea para Visual Explain?
ninguna	Sí	No
FOR SNAPSHOT	No	Sí
WITH SNAPSHOT	Sí	Sí

Si no se ha especificado la cláusula FOR SNAPSHOT ni la cláusula WITH SNAPSHOT, no se toma una instantánea de Explain.

Antes de realizar la invocación de la sentencia EXPLAIN, el usuario deberá haber creado las tablas Explain. La información que genera esta sentencia se almacena en las tablas Explain, en el esquema que se ha designado en el momento de compilarse la sentencia.

Si no se produce ningún error durante la compilación de la *sentencia-sql-explicable* suministrada, entonces no se almacena información en las tablas Explain.

El plan de acceso generado para la *sentencia-sql-explicable* no se guarda y, por lo tanto, no se puede invocar después. La información de Explain para la *sentencia-sql-explicable* se inserta cuando se compila la sentencia EXPLAIN en sí.

Para una sentencia EXPLAIN de SQL estático, la información se inserta en las tablas Explain en tiempo de vinculación o al volver a vincular explícitamente. Durante la precompilación, se comentan las sentencias EXPLAIN estáticas en el archivo fuente de la aplicación modificado. En el momento del enlace, las sentencias EXPLAIN del catálogo SYSCAT.STATEMENTS. Cuando se ejecuta el paquete, la sentencia EXPLAIN no se ejecuta. Tenga en cuenta que los números de sección para todas las sentencias de la aplicación serán secuenciales e incluirán las sentencias EXPLAIN. Una alternativa a utilizar una sentencia EXPLAIN estática es utilizar una combinación de las opciones EXPLAIN y EXPLSNAP BIND/PREP. Las sentencias EXPLAIN estáticas se pueden utilizar para hacer que las tablas Explain se llenen para una sentencia de SQL estática específica entre muchas; simplemente

ponga un prefijo en la sentencia de destino con la sintaxis de sentencia EXPLAIN adecuada y enlace la aplicación sin utilizar las opciones BIND/PREP de Explain. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

Para las sentencias EXPLAIN de SQL con enlace incremental, las tablas Explain se llenan de datos cuando se somete a compilación la sentencia EXPLAIN. Cuando se ejecuta el paquete, la sentencia EXPLAIN no realiza ningún proceso (aunque se ejecutará correctamente). Cuando las tablas Explain se llenan con datos, el calificador de la tabla Explain y el ID de autorización utilizado durante el llenado de datos serán los pertenecientes al propietario del paquete. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

Para sentencias EXPLAIN dinámicas, las tablas Explain se llenan en el momento que la sentencia EXPLAIN se somete a la compilación. Una sentencia Explain puede prepararse con la sentencia PREPARE, pero su ejecución no realizará ningún proceso (aunque la sentencia se ejecutará correctamente). Una alternativa a la emisión de sentencias EXPLAIN dinámicas es utilizar una combinación de los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT para explicar las sentencias de SQL dinámico. La sentencia EXPLAIN debe utilizarse cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de Explain real.

Si la opción de vinculación REOPT se establece en ONCE y el registro especial CURRENT EXPLAIN MODE o CURRENT EXPLAIN SNAPSHOT se establece en REOPT, la ejecución de sentencias de SQL dinámico y estático que contienen variables del lenguaje principal, registros especiales o marcadores de parámetro hará que sólo se capture la información de Explain para la sentencia cuando se reoptimice la sentencia. De forma alternativa, la opción de vinculación REOPT se establece en ALWAYS, la información de Explain se capturará cada vez que se ejecuten estas sentencias.

Ejemplos:

Ejemplo 1: Explique una sentencia SELECT simple y póngale el distintivo QUERYNO = 13.

```
EXPLAIN PLAN SET QUERYNO = 13
  FOR SELECT C1
   FROM T1
```

Ejemplo 2: Explique una sentencia SELECT simple y póngale el distintivo QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
  FOR SELECT C1
   FROM T1
```

Ejemplo 3: Explique una sentencia SELECT simple y póngale los distintivos QUERYNO = 13 y QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
  FOR SELECT C1
   FROM T1
```

Ejemplo 4: Intente obtener información de Explain cuando no existen las tablas Explain.

```
EXPLAIN ALL FOR SELECT C1
   FROM T1
```

Esta sentencia fallará porque no se han definido las tablas Explain (SQLSTATE 42704).

Ejemplo 5: La siguiente sentencia será satisfactoria si se encuentra en la antememoria del paquete y ya se ha compilado utilizando REOPT ONCE.

```
EXPLAIN ALL WITH REOPT ONCE FOR SELECT C1
   FROM T1     WHERE C1 = :<variable del lenguaje principal>
```

Información relacionada:

- "Tabla EXPLAIN_ARGUMENT" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_OBJECT" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_OPERATOR" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_PREDICATE" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_STREAM" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla ADVISE_INDEX" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla ADVISE_WORKLOAD" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_INSTANCE" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_STATEMENT" en la publicación *Consulta de SQL, Volumen 1*
- "Tablas de Explain" en la publicación *Consulta de SQL, Volumen 1*

FETCH

La sentencia FETCH posiciona el cursor en la siguiente fila de su tabla de resultados y asigna los valores de dicha fila a las variables del lenguaje principal.

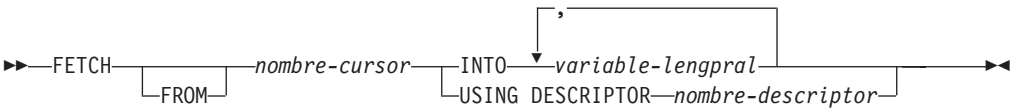
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

Sintaxis:



Descripción:

nombre-cursor
Identifica el cursor que se va a utilizar en una operación de lectura. El *nombre-cursor* debe identificar un cursor declarado, tal como se explica en "DECLARE CURSOR". La sentencia DECLARE CURSOR debe preceder a la sentencia FETCH en el programa fuente. Cuando se ejecuta la sentencia FETCH, el cursor debe estar en el estado abierto.

Si el cursor está situado actualmente en la última fila o después de ella en la tabla de resultados:

- SQLCODE se establece en +100 y SQLSTATE se establece en '02000'.
- El cursor se sitúa después de la última fila.
- Los valores no se asignan a las variables del lenguaje principal.

Si el cursor está situado actualmente antes de una fila, se volverá a situar en dicha fila y se asignarán los valores a las variables del lenguaje principal tal como especifican INTO o USING.

Si el cursor está situado actualmente un una fila que no es la última, se situará en la siguiente fila y se asignarán los valores de dicha fila a las variables del lenguaje principal tal como se especifican INTO o USING.

INTO *variable-lengpral*, **...**
Identifica una o varias variables del lenguaje principal que deben describirse de acuerdo a las normas para la declaración de las variables del lenguaje principal. El primer valor de la fila del resultado se asigna a la primera variable del lenguaje principal de la lista, el segundo valor a la segunda variable del lenguaje principal, etcétera. Para los valores LOB de la lista de selección, el destino puede ser una variable del lenguaje principal normal (si es lo suficientemente grande), una variable localizadora o una variable de referencia a archivos.

USING DESCRIPTOR *nombre-descriptor*
Identifica una SQLDA que debe contener una descripción válida de cero o más variables del lenguaje principal.

Antes de procesar la sentencia FETCH, el usuario debe establecer los campos siguientes en la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA.
- SQLDABC para indicar el número de bytes de almacenamiento asignado para la SQLDA.
- SQLD para indicar el número de variables utilizadas en SQLDA al procesar la sentencia.
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que 16 + SQLN*(N), donde N es la longitud de una ocurrencia SQLVAR.

Si las columnas resultantes de LOB o de tipo estructurado necesitan acomodarse, debe haber dos entradas SQLVAR para cada elemento de la lista de selección (o columna de la tabla de resultados).

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

La variable *n* identificada en la cláusula INTO o descrita en la SQLDA corresponde a la columna *n* de la tabla de resultados del cursor. El tipo de datos de cada variable debe ser compatible con su columna correspondiente.

Cada asignación que se realiza para una variable se realiza de acuerdo con normas específicas. Si el número de variables es menor que el número de valores de la fila, el campo SQLWARN3 de la SQLDA se establece en 'W'. Tenga en cuenta que no hay ningún aviso si hay más variables que el número de columnas del resultado. Si se produce un error de asignación, no se asigna el valor a la variable y no se asignan más valores a las variables. Cualquier valor que ya se haya asignado a las variables continúa asignado.

Notas:

- Un cursor abierto tiene tres posiciones posibles:
  - Antes de una fila
  - En una fila
  - Después de la última fila.
- Si un cursor está en una fila, dicha fila se llama la fila actual del cursor. Un cursor al que se haga referencia en una sentencia UPDATE o DELETE debe estar situado en una fila. Un cursor sólo puede estar en una fila como resultado de una sentencia FETCH.
- Cuando se recuperan datos con localizadores de LOB en situaciones en que no es necesario conservar el localizador entre una sentencia FETCH y otra, es aconsejable emitir una sentencia FREE LOCATOR antes de emitir la siguiente sentencia FETCH, ya que los recursos del localizador son limitados.
- Es posible que se produzca un error que haga que el estado del cursor sea imprevisible.

- Es posible que FETCH no devuelva un aviso. También es posible que el aviso devuelto corresponda a una fila recuperada anteriormente. Esto se produce como resultado de las optimizaciones como, por ejemplo, la utilización de tablas temporales del sistema o de operadores de desplazamiento descendente.
- La puesta de sentencias en antememoria afecta al funcionamiento de la sentencia EXECUTE IMMEDIATE.
- DB2 CLI da soporte a funciones adicionales de recuperación. Por ejemplo cuando la tabla de resultados de un cursor es de sólo lectura, se puede utilizar la función SQLFetchScroll() para situar el cursor en cualquier punto dentro de dicha tabla de resultados.
- Para un cursor actualizable, se obtiene un bloqueo en un fila cuando se capta.
- Si la definición del cursor contiene una sentencia de cambio de datos de SQL o invoca una rutina que modifica datos de SQL, un error durante la operación de recuperación no hace que las filas modificadas se retrotraigan, aunque el error haga que se cierre el cursor.

Ejemplos:

Ejemplo 1: En este ejemplo C, la sentencia FETCH coloca los resultados de la sentencia SELECT en las variables de programa `dnum`, `dname` y `mnum`. Cuando ya no quedan más filas para leer, se devuelve la condición de no encontrado.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

mientras (SQLCODE==0) {
    EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}

EXEC SQL CLOSE C1;
```

Ejemplo 2: La sentencia FETCH utiliza una SQLDA.

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

Información relacionada:

- "DECLARE CURSOR" en la página 483
- "EXECUTE" en la página 541
- "SQLDA (área de descriptores de SQL)" en la publicación *Consulta de SQL, Volumen 1*
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "cursor.sqb -- How to update table data with cursor statically (MF COBOL)"
- "tbread.sqc -- How to read tables (C)"
- "tut_mod.sqc -- How to modify table data (C)"
- "tbread.sqC -- How to read tables (C++)"
- "tut_mod.sqC -- How to modify table data (C++)"
- "TutMod.sqlj -- Modify data in a table (SQLj)"

FLUSH EVENT MONITOR

La sentencia FLUSH EVENT MONITOR graba los valores actuales del supervisor de bases de datos para todos los tipos de supervisores activos asociados con el supervisor de sucesos *nombre-supervisor-sucesos* en el destino de E/S del supervisor de sucesos. Por lo tanto, en cualquier momento está disponible un registro parcial del suceso para los supervisores de sucesos que tienen una frecuencia baja de generación de registros (por ejemplo, supervisor de sucesos de bases de datos). Estos registros se indican en el registro cronológico del supervisor de sucesos mediante un identificador de *registro parcial*.

Cuando se desecha un supervisor de sucesos, sus almacenamientos intermedios internos activos se graban en el objeto de salida del supervisor de sucesos.

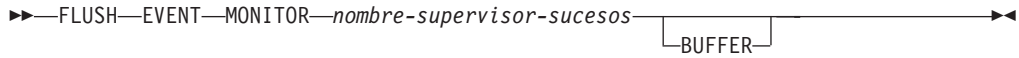
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Los privilegios del ID de autorización deben incluir la autorización SYSADM o DBADM (SQLSTATE 42502).

Sintaxis:



Descripción:

nombre-supervisor-sucesos
Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL.

BUFFER
Indica que se deben grabar los almacenamientos intermedios del supervisor de sucesos. Si se especifica BUFFER, no se generan los registros parciales. Sólo se graban los datos que ya están presentes en los almacenamientos intermedios del supervisor de sucesos.

Notas:

- Cuando se desecha el supervisor de sucesos no se restauran los valores del supervisor de sucesos. Esto significa que el registro del supervisor de sucesos que se habría generado si no se hubiese desechado, se seguirá generando cuando se active el suceso del supervisor normal.

FLUSH PACKAGE CACHE

La sentencia FLUSH PACKAGE CACHE elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes. Esta sentencia da lugar a la invalidación lógica de cualquier sentencia de SQL dinámica colocada en antememoria y hace que DB2 compile implícitamente la siguiente petición para la misma sentencia de SQL.

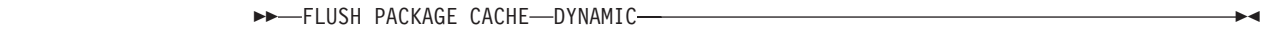
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Los privilegios del ID de autorización de la sentencia deben incluir la autorización SYSADM o la autorización DBADM (SQLSTATE 42502).

Sintaxis:



Notas:

- Esta sentencia afecta a todas las entradas de SQL dinámico colocadas en antememoria que están en la antememoria de paquetes de todas las particiones de base de datos activas.
- Puesto que las sentencias de SQL dinámico colocadas en antememoria se invalidan, la antememoria de paquetes que se utiliza para la entrada colocada en antememoria se liberará si la entrada no está utilizándose cuando se ejecuta la sentencia FLUSH PACKAGE CACHE.
- Cualquier sentencia de SQL dinámica colocada en antememoria que actualmente esté utilizándose podrá seguir existiendo en la antememoria de paquetes hasta que el usuario actual ya no la necesite; el nuevo usuario siguiente de la misma sentencia hará que DB2 realice una preparación implícita de la sentencia y el nuevo usuario ejecutará la nueva versión de la sentencia de SQL dinámica colocada en antememoria.

FOR

La sentencia FOR ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.

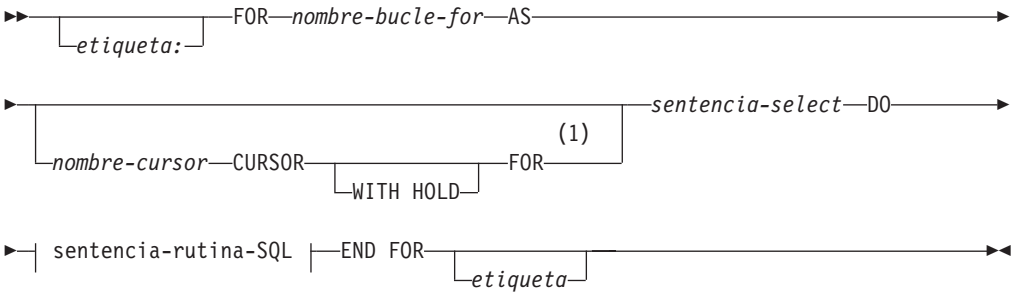
Invocación:

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

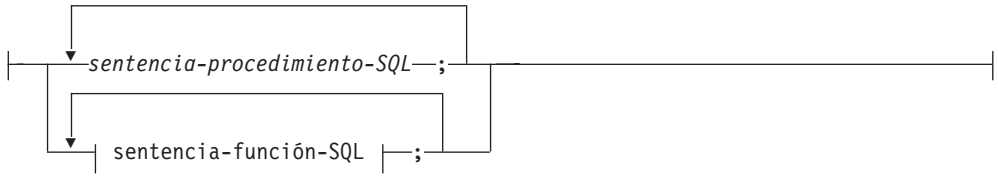
Autorización:

No se requieren privilegios para invocar una sentencia FOR. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia FOR. Para obtener información acerca de la autorización necesaria para utilizar un cursor, consulte la descripción de la sentencia DECLARE CURSOR.

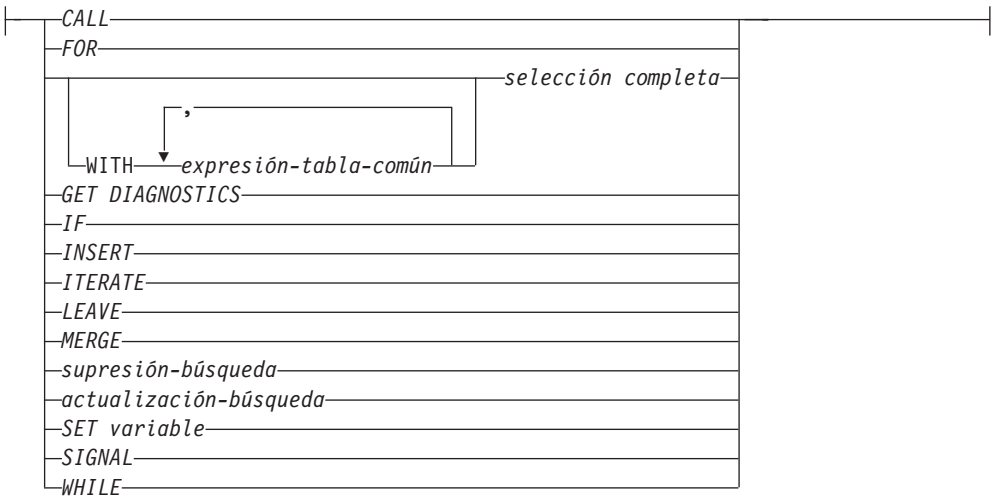
Sintaxis:



sentencia-rutina-SQL:



sentencia-función-SQL:



Notas:

1 Esta opción sólo se puede utilizar en el contexto de un procedimiento SQL.

Descripción:

etiqueta
Especifica la etiqueta de la sentencia FOR. Si se especifica la etiqueta inicial, esa etiqueta puede utilizarse en sentencias LEAVE e ITERATE. Si se especifica la etiqueta final, ésta deberá ser igual que la etiqueta inicial.

nombre-bucle-for
Especifica una etiqueta para la sentencia compuesta implícita que se genera para implementar la sentencia FOR. Sigue las normas para la etiqueta de una sentencia compuesta excepto en que no se puede utilizar con una sentencia ITERATE o LEAVE en la sentencia FOR. El *nombre-bucle-for* se utiliza para calificar los nombres de columna devueltos por la *sentencia-select* especificada.

nombre-cursor
Designa el cursor utilizado para seleccionar filas de la tabla de resultados de la sentencia SELECT. Si no se especifica, DB2 genera un nombre de cursor exclusivo. Para una descripción de la cláusula WITH HOLD, vea la sentencia "DECLARE CURSOR".

sentencia-select
Especifica la sentencia SELECT del cursor. Todas las columnas de la lista de selección deben tener un nombre y no pueden existir dos columnas con el mismo nombre.

En un activador, una función, un método o una sentencia dinámica compuesta, la *sentencia-select* sólo debe constar de una *selección completa* con expresiones de tabla común opcionales.

sentencia-procedimiento-SQL
Especifica una o varias sentencias que se deben invocar para cada fila de la tabla. *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

sentencia-función-SQL
Especifica una o varias sentencias que se deben invocar para cada fila de la tabla. No se da soporte a una operación de actualización-búsqueda,

supresión-búsqueda ni INSERT en apodos. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL.

Normas:

- La lista de selección debe constar de nombres de columna exclusivos y la tabla especificada en la lista de selección debe existir cuando se crea el procedimiento, o debe ser una tabla creada en una sentencia de procedimiento SQL anterior.
- El cursor especificado en una sentencia-for no puede estar referenciado fuera de la sentencia-for y no se puede especificar en una sentencia OPEN, FETCH ni CLOSE.

Ejemplos:

El ejemplo siguiente utiliza la sentencia-for para ejecutar un proceso iterativo sobre la tabla `employee` completa. Para cada fila de la tabla, la variable SQL `fullname` se establece en el primer apellido del empleado, seguido de una coma, el nombre, un espacio en blanco y la inicial del segundo apellido. Cada valor de `fullname` se inserta en la tabla `tnames`.

```
BEGIN ATOMIC
  DECLARE fullname CHAR(40);
  FOR vl AS
    SELECT firstnme, midinit, lastname FROM employee
      DO
      SET fullname = lastname || ',' || firstnme ||' ' || midinit;
      INSERT INTO tnames VALUES (fullname);
  END FOR
END
```

Información relacionada:

- "DECLARE CURSOR" en la página 483
- "SQL compuesto (procedimiento)" en la página 139

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"

FREE LOCATOR

La sentencia FREE LOCATOR elimina la asociación entre una variable localizadora y su valor.

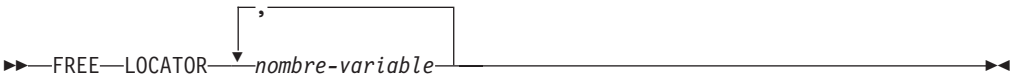
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:

LOCATOR *nombre-variable,* **...**
Identifica una o varias variables localizadoras que deben declararse de acuerdo con las normas para la declaración de variables localizadoras.

La variable-localizadora debe tener actualmente un localizador asignado a ella. Es decir, deberá haberse asignado un localizador durante esta unidad de trabajo (por medio de una sentencia CALL, FETCH, SELECT INTO o VALUES INTO) y no deberá haberse liberado posteriormente (por medio de una sentencia FREE LOCATOR); de lo contrario, se devolverá un error (SQLSTATE 0F001).

Si se especifica más de un localizador, se liberan todos los localizadores que se pueden liberar, sin tener en cuenta los errores detectados en otros localizadores de la lista.

Ejemplo:

En un programa COBOL, libere las variables localizadoras de BLOB, TKN-VIDEO y TKN-BUF, y la variable localizadora de CLOB, LIFE-STORY-LOCATOR.

```
EXEC SQL
FREE LOCATOR :TKN-VIDEO, :TKN-BUF, :LIFE-STORY-LOCATOR
END-EXEC.
```

Ejemplos relacionados:

- "dtlob.c -- How to read and write LOB data"
- "spserver.c -- Definition of various types of stored procedures"
- "dtlob.sqc -- How to use the LOB data type (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "lobeval.sqb -- Demonstrates how to use a Large Object (LOB) (MF COBOL)"

GET DIAGNOSTICS

La sentencia GET DIAGNOSTICS se utiliza para obtener información acerca de la sentencia de SQL ejecutada anteriormente.

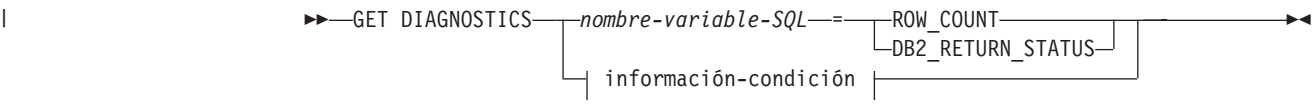
Invocación:

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

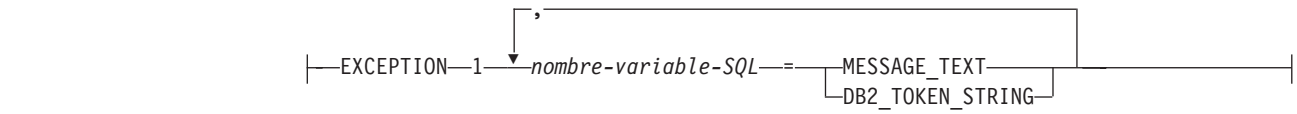
Autorización:

No se necesita.

Sintaxis:



información-condición:



Descripción:

nombre-variable-SQL
Identifica la variable que es el sujeto de la asignación. Si se especifica ROW_COUNT o DB2_RETURN_STATUS, la variable debe ser una variable de enteros. De lo contrario, la variable debe ser CHAR o VARCHAR. Las variables SQL se pueden definir en una sentencia compuesta.

ROW_COUNT
Identifica el número de filas que se asocian a la sentencia de SQL anterior. Si la sentencia de SQL anterior es una sentencia DELETE, INSERT o UPDATE, ROW_COUNT identifica el número de filas que se han calificado para la operación. Si la sentencia anterior es una sentencia PREPARE, ROW_COUNT identifica el número *estimado* de filas de resultados de la sentencia preparada.

DB2_RETURN_STATUS
Identifica el valor de estado devuelto por el procedimiento almacenado asociado a la sentencia de SQL ejecutada anteriormente, siempre que la sentencia fuera una sentencia CALL que invoca un procedimiento que devuelve un estado. Si la sentencia anterior no es una sentencia de este tipo, el valor que se devuelve carece de significado y puede ser cualquier entero.

información-condición
Especifica que va a devolverse la información de error o de aviso para la sentencia de SQL que se ha ejecutado anteriormente. Si se necesita información acerca de un error, la sentencia GET DIAGNOSTICS debe ser la primera sentencia que debe especificarse en el descriptor de contexto que se encargará de manejar el error. Si se necesita información acerca de un aviso y si el descriptor de contexto va a recibir el control de la condición de aviso, la

sentencia GET DIAGNOSTICS debe ser la primera sentencia que debe especificarse en ese descriptor de contexto. Si el descriptor de contexto *no* va a recibir el control de la condición de aviso, la sentencia GET DIAGNOSTICS debe ser la siguiente sentencia que debe ejecutarse. Esta opción sólo puede especificarse en el contexto de un Procedimiento de SQL (SQLSTATE 42601).

MESSAGE_TEXT
Identifica cualquier texto de mensaje de error o de aviso que se devuelve de la sentencia de SQL ejecutada anteriormente. El texto del mensaje se devuelve en el idioma del servidor de bases de datos en el que se ha procesado la sentencia. Si la sentencia se ha completado con un SQLCODE que es cero, para una variable VARCHAR se devuelve una serie de caracteres vacía o, en el caso de una variable CHAR, se devuelven blancos.

DB2_TOKEN_STRING
Identifica cualquier símbolo de mensaje de error o de aviso que se devuelve de la sentencia de SQL ejecutada anteriormente. Si la sentencia se ha completado con un SQLCODE que es cero o si el SQLCODE no tiene ningún símbolo, para una variable VARCHAR se devuelve una serie de caracteres vacía o, en el caso de una variable CHAR, se devuelven blancos.

Notas:

- *Compatibilidades*
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - RETURN_STATUS puede especificarse en lugar de DB2_RETURN_STATUS.
- La sentencia GET DIAGNOSTICS no cambia el contenido del área de diagnósticos (SQLCA). Las variables especiales SQLSTATE o SQLCODE declaradas en un procedimiento SQL se establecen en el SQLSTATE o SQLCODE devuelto por la ejecución de la sentencia GET DIAGNOSTICS.

Ejemplos:

En un procedimiento SQL, ejecute una sentencia GET DIAGNOSTICS para determinar cuántas filas se actualizaron.

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
   LANGUAGE SQL
   BEGIN
      DECLARE SQLSTATE CHAR(5);
      DECLARE rcount INTEGER;
      UPDATE CORPDATA.PROJECT
         SET PRSTAFF = PRSTAFF + 1.5
         WHERE DEPTNO = deptnbr;
      GET DIAGNOSTICS rcount = ROW_COUNT;
-- En este momento, rcount contiene el número de filas que se actualizaron.
...
   END
```

Dentro de un procedimiento SQL, procese el valor de estado devuelto por la invocación de un procedimiento almacenado llamado TRYIT, el cual puede devolver explícitamente un valor positivo, lo que indica un error de usuario, o encontrar errores SQL que producirían un valor de estado de retorno negativo. Si el procedimiento se ejecuta satisfactoriamente, devuelve un valor igual a cero.

```
CREATE PROCEDURE TESTIT ()
   LANGUAGE SQL
   A1:BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
      CALL TRYIT;
```

```
   GET DIAGNOSTICS RETVAL = DB2_RETURN_STATUS;
   IF RETVAL <> 0 THEN
      ...
      LEAVE A1;
   ELSE
      ...
   END IF;
END A1
```

GOTO

La sentencia GOTO se utiliza para definir una bifurcación a una etiqueta definida por el usuario dentro de un procedimiento de SQL.

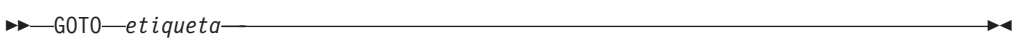
Invocación:

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

Autorización:

No se necesita.

Sintaxis:



Descripción:

etiqueta
Especifica una sentencia con etiqueta donde debe continuar el proceso. La sentencia con etiqueta y la sentencia GOTO deben estar en el mismo ámbito:

- Si la sentencia GOTO se define en una sentencia FOR, la *etiqueta* se debe definir dentro de la misma sentencia FOR, que no sea una sentencia FOR anidada ni una sentencia compuesta anidada.
- Si la sentencia GOTO se define en una sentencia compuesta, la *etiqueta* se debe definir dentro de la misma sentencia compuesta, que no sea una sentencia FOR anidada ni una sentencia compuesta anidada.
- Si la sentencia GOTO se define en un gestor de condiciones, la *etiqueta* se debe definir en el mismo gestor de condiciones, de acuerdo con las demás normas sobre el ámbito
- Si la sentencia GOTO se define fuera de un gestor de condiciones, la *etiqueta* no se debe definir dentro de un gestor de condiciones.

Si la *etiqueta* no está definida dentro de un ámbito que sea accesible para la sentencia GOTO, se produce un error (SQLSTATE 42736).

Notas:

- Es aconsejable utilizar la sentencia GOTO de forma moderada. Esta sentencia interfiere en las secuencias normales de proceso, lo que hace que la rutina sea más difícil de leer y actualizar. Antes de utilizar una sentencia GOTO, determine si en su lugar puede utilizarse otra sentencia, tal como IF o LEAVE, para eludir la necesidad de utilizar una sentencia GOTO.

Ejemplos:

En la sentencia compuesta siguiente, los parámetros *rating* y *v_empno* se pasan al procedimiento que, a continuación, devuelve el parámetro de salida *return_parm* como una duración en forma de fecha. Si el tiempo que el empleado lleva prestando servicios a la empresa es inferior a seis meses, la sentencia GOTO transfiere el control al final del procedimiento y *new_salary* no cambia.

```
CREATE PROCEDURE adjust_salary
  (IN v_empno CHAR(6),
      IN rating INTEGER)
```

```
OUT return_parm DECIMAL (8,2))
   MODIFIES SQL DATA
   LANGUAGE SQL
   BEGIN
  DECLARE new_salary DECIMAL (9,2)
  DECLARE service DECIMAL (8,2)
    SELECT SALARY, CURRENT_DATE - HIREDATE
      INTO new_salary, service
                       FROM EMPLOYEE
      WHERE EMPNO = v_empno
        IF service < 600
          THEN GOTO EXIT
        END IF
      IF rating = 1
      THEN SET new_salary = new_salary + (new_salary * .10)
        ELSE IF rating = 2
      THEN SET new_salary = new_salary + (new_salary * .05)
        END IF
        UPDATE EMPLOYEE
      SET SALARY = new_salary
      WHERE EMPNO = v_empno
    EXIT: SET return_parm = service
      END
```

GRANT (Autorizaciones de base de datos)

Este formato de la sentencia GRANT otorga las autorizaciones que se aplican a toda la base de datos (en lugar de privilegios que se aplican a objetos específicos de la base de datos).

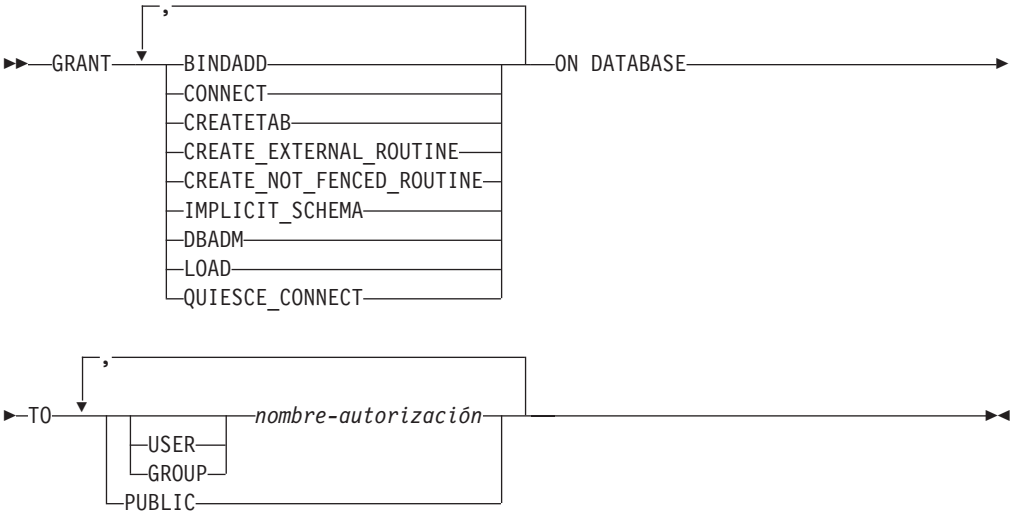
Invocación:

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

Autorización:

Para otorgar la autorización DBADM, se necesita la autorización SYSADM. Para otorgar otras autorizaciones, son necesarias la autorización DBADM o SYSADM.

Sintaxis:



Descripción:

BINDADD
Otorga la autorización para crear paquetes. El creador de un paquete tiene automáticamente el privilegio CONTROL en dicho paquete y conserva este privilegio incluso si se revoca posteriormente la autorización BINDADD.

CONNECT
Otorga la autorización para acceder a la base de datos.

CREATETAB
Otorga la autorización para crear tablas base. El creador de una tabla base tiene automáticamente el privilegio CONTROL en dicha tabla. El creador conserva este privilegio incluso si posteriormente se revoca la autorización CREATETAB.

No se necesita ninguna autorización explícita para la creación de vistas. Una vista se puede crear en cualquier momento si el ID de autorización de la sentencia utilizada para crear la vista tiene el privilegio CONTROL o SELECT en cada tabla base de la vista.

CREATE_EXTERNAL_ROUTINE
Otorga la autorización para registrar rutinas externas. Deberá tenerse cuidado de que las rutinas que se registran de esta forma no generen efectos secundarios negativos. (Para obtener más información, consulte la descripción de la cláusula THREADSAFE en las sentencias de rutina CREATE o ALTER.)

Cuando una rutina externa se ha registrado, ésta sigue existiendo, aunque posteriormente se invoque CREATE_EXTERNAL_ROUTINE.

CREATE_NOT_FENCED_ROUTINE
Otorga la autorización para registrar rutinas que se ejecutan en el proceso del gestor de bases de datos. Deberá tenerse cuidado de que las rutinas que se registran de esta forma no generen efectos secundarios negativos. (Para obtener más información, consulte la descripción de la cláusula FENCED en la sentencias de rutina CREATE o ALTER.)

Cuando una rutina se ha registrado como no limitada, sigue ejecutándose de esta forma, aunque posteriormente se invoque CREATE_NOT_FENCED_ROUTINE.

CREATE_EXTERNAL_ROUTINE se otorga automáticamente a un *nombre-autorización* al que se haya otorgado la autorización CREATE_NOT_FENCED_ROUTINE.

IMPLICIT_SCHEMA
Otorga la autorización para crear implícitamente un esquema.

DBADM
Otorga la autorización de administrador de bases de datos y todas las demás autorizaciones para la base de datos. Un administrador de bases de datos dispone de todos los privilegios para todos los objetos de la base de datos y puede otorgar estos privilegios a otros usuarios.

Nota: Todas las demás autorizaciones de base de datos se otorgan implícita o automáticamente a un *nombre-autorización* al que se haya otorgado autorización DBADM.

LOAD
Otorga autorización para cargar en la base de datos. Esta autorización proporciona al usuario el derecho a utilizar el programa de utilidad LOAD para la base de datos. Por omisión, SYSADM y DBADM también tienen esta autorización. Sin embargo, si un usuario sólo tiene autorización LOAD (no SYSADM ni DBADM), es necesario que el usuario tenga también privilegios para tablas. Además del privilegio LOAD, el usuario debe tener:

- Privilegio INSERT sobre la tabla para realizar una carga en la modalidad INSERT, TERMINATE (para finalizar un LOAD INSERT anterior) o RESTART (para reiniciar un LOAD INSERT anterior)
- Privilegio INSERT y DELETE sobre la tabla para realizar una carga en la modalidad REPLACE, TERMINATE (para finalizar un LOAD REPLACE anterior) o RESTART (para reiniciar un LOAD REPLACE anterior)
- Privilegio INSERT para la tabla de excepciones, si esa tabla se utiliza como parte de LOAD

QUIESCE_CONNECT
Otorga la autorización para acceder a la base de datos mientras está inactiva.

TO
Especifica a quién se otorgan las autorizaciones.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga las autorizaciones a todos los usuarios. DBADM no se puede otorgar a PUBLIC.

Normas:

- Si no se especifica USER ni GROUP, entonces
  - Si el nombre-autorización se ha definido en el sistema operativo únicamente como GROUP, se supone GROUP.
  - Si el nombre-autorización se ha definido en el sistema operativo únicamente como USER, o no se ha definido, se supone USER.
  - Si el nombre-autorización se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - CREATE_NOT_FENCED puede especificarse en lugar de CREATE_NOT_FENCED_ROUTINE

Ejemplos:

Ejemplo 1: Otorgue a los usuarios WINKEN, BLINKEN y NOD la autorización para conectarse a la base de datos.

```
GRANT CONNECT ON DATABASE TO USER WINKEN, USER BLINKEN, USER NOD
```

Ejemplo 2: Otorgue (GRANT) la autorización BINDADD en la base de datos a un grupo llamado D024. Hay un grupo y un usuario llamados D024 en el sistema.

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

Observe que debe especificarse la palabra clave GROUP; de lo contrario, se producirá un error ya que existen tanto un usuario como un grupo llamados D024. Cualquier miembro del grupo D024 podrá enlazar paquetes en la base de datos, pero el usuario D024 no podrá (a menos que también sea miembro del grupo D024, se le haya otorgado la autorización BINDADD previamente o se haya otorgado la autorización BINDADD a otro grupo del que D024 sea miembro).

Información relacionada:

- “GRANT (Privilegios de índice)” en la página 574
- “GRANT (Privilegios de paquete)” en la página 576
- “GRANT (Privilegios de esquema)” en la página 583
- “GRANT (Privilegios de tabla, vista o apodo)” en la página 592
- “GRANT (Privilegios de servidor)” en la página 588

- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "GRANT (Privilegios de rutina)" en la página 579

Ejemplos relacionados:

- "dbauth.sqb -- How to grant and display authorities on a database (MF COBOL)"
- "dbauth.sqc -- How to grant, display, and revoke authorities at database level (C)"
- "dbauth.sqC -- How to grant, display, and revoke authorities at database level (C++)"
- "DbAuth.java -- Grant, display or revoke privileges on database (JDBC)"
- "DbAuth.sqlj -- Grant, display or revoke privileges on database (SQLj)"

GRANT (Privilegios de índice)

Esta forma de la sentencia GRANT otorga el privilegio CONTROL en índices.

Invocación:

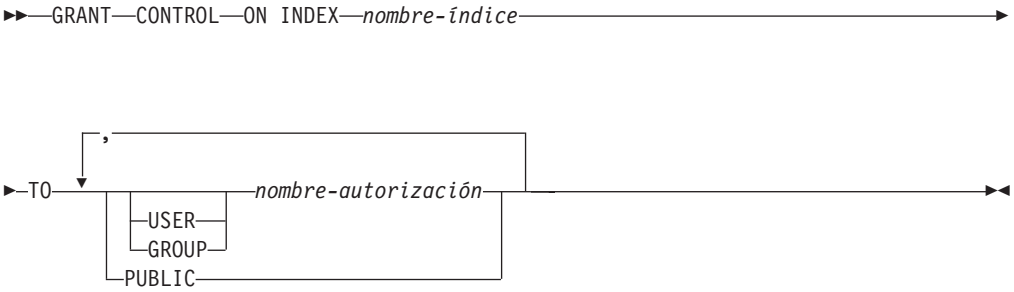
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SYSADM.

Sintaxis:



Descripción:

CONTROL
Otorga el privilegio para eliminar el índice. Esta es la autorización CONTROL para los índices, que se otorga automáticamente a los creadores de índices.

ON INDEX *nombre-índice*
Identifica el índice para el cual se debe otorgar el privilegio CONTROL.

TO
Especifica a quién se otorgan los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a todos los usuarios.

Normas:

- Si no se especifica USER ni GROUP, entonces
  - Si se define el nombre-autorización en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el nombre-autorización en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si el nombre-autorización se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).

Ejemplo:

```
GRANT CONTROL ON INDEX DEPTIDX TO USER USER4
```

Información relacionada:

- "GRANT (Autorizaciones de base de datos)" en la página 570
- "GRANT (Privilegios de paquete)" en la página 576
- "GRANT (Privilegios de esquema)" en la página 583
- "GRANT (Privilegios de tabla, vista o apodo)" en la página 592
- "GRANT (Privilegios de servidor)" en la página 588
- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "GRANT (Privilegios de rutina)" en la página 579

GRANT (Privilegios de paquete)

Esta forma de la sentencia GRANT otorga los privilegios en un paquete.

Invocación:

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

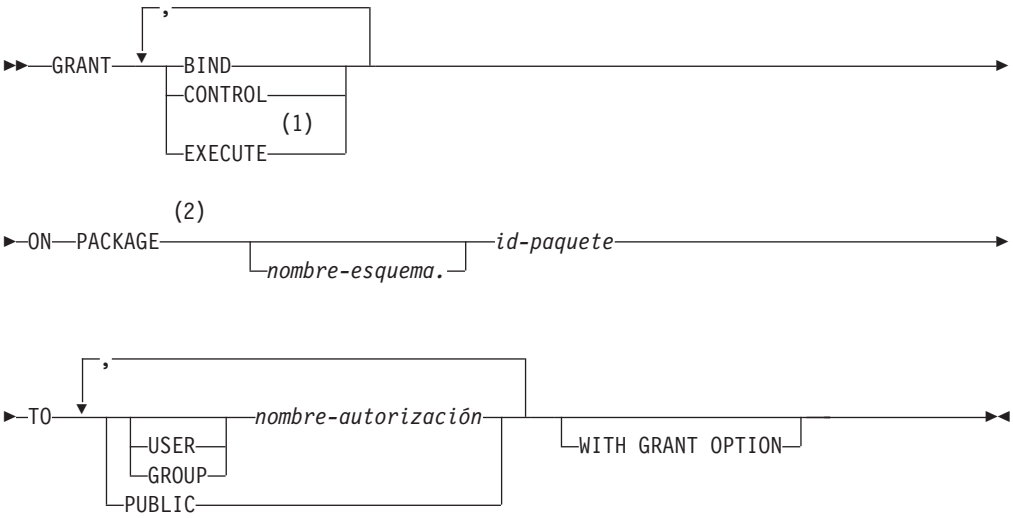
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para el paquete referenciado
- El privilegio WITH GRANT OPTION para cada privilegio identificado en *nombre-paquete*.
- Autorización SYSADM o DBADM.

Para otorgar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Sintaxis:



Notas:

1 RUN se puede utilizar como sinónimo de EXECUTE.

2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

Descripción:

BIND
Otorga el privilegio para enlazar un paquete. El privilegio BIND permite a un usuario volver a emitir el mandato BIND para ese paquete o bien emitir el mandato REBIND. También permite a un usuario crear una nueva versión de un paquete existente.

Además del privilegio BIND, un usuario debe disponer de los privilegios necesarios para cada tabla a la que hagan referencia las sentencias DML estáticas contenidas en un programa. Esto es necesario porque la autorización para las sentencias DML estáticas se comprueba durante el enlace.

CONTROL
Otorga el privilegio para volver a enlazar, eliminar o ejecutar el paquete y extender los privilegios del paquete a otros usuarios. El privilegio CONTROL para paquetes se otorga automáticamente a los creadores de los paquetes. El propietario de un paquete es el enlazador del paquete o el ID especificado con la opción OWNER durante el enlace/precompilación.

BIND y EXECUTE se otorgan automáticamente a un *nombre-autorización* al que se le otorga el privilegio CONTROL.

CONTROL permite otorgar los privilegios anteriores (excepto CONTROL) a otros usuarios.

EXECUTE
Otorga el privilegio para ejecutar el paquete.

ON PACKAGE *nombre-esquema.id-paquete*
Especifica el nombre del paquete en el que se deben otorgar los privilegios. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. La concesión de un privilegio de paquete se aplica a todas las versiones del paquete (es decir, a todos los paquetes que comparten el mismo ID de paquete y esquema de paquete).

TO
Especifica a quién se otorgan los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a todos los usuarios.

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado pueda otorgar (GRANT) los privilegios a otros usuarios.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios que pueden aplicarse, excepto CONTROL (SQLSTATE 01516).

Normas:

- Si no se especifica USER ni GROUP, entonces
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el *nombre-autorización* en el sistema operativo sólo como USER o si no esta definido, se supone USER.

– Si el *nombre-autorización* se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).

Notas:

- Los privilegios de paquete se aplican a todas las versiones de un paquete (es decir, a todos los paquetes que comparten el mismo ID de paquete y esquema de paquete). No es posible restringir el acceso sólo a una única versión. Puesto que el privilegio CONTROL se otorga implícitamente al usuario que enlaza el paquete, si dos usuarios distintos enlazan dos versiones de un paquete, a ambos usuarios se otorgará implícitamente acceso al paquete del otro.

Ejemplos:

Ejemplo 1: Otorgue el privilegio EXECUTE en PACKAGE CORPDATA.PKGA a PUBLIC.

```
GRANT EXECUTE
  ON PACKAGE CORPDATA.PKGA
  TO PUBLIC
```

Ejemplo 2: Otorgue (GRANT) el privilegio EXECUTE en el paquete CORPDATA.PKGA a un usuario denominado EMPLOYEE. No hay ningún grupo ni usuario llamado EMPLOYEE.

```
GRANT EXECUTE ON PACKAGE
  CORPDATA.PKGA TO EMPLOYEE
```

o

```
GRANT EXECUTE ON PACKAGE
  CORPDATA.PKGA TO USER EMPLOYEE
```

Información relacionada:

- “GRANT (Autorizaciones de base de datos)” en la página 570
- “GRANT (Privilegios de índice)” en la página 574
- “GRANT (Privilegios de esquema)” en la página 583
- “GRANT (Privilegios de tabla, vista o apodo)” en la página 592
- “GRANT (Privilegios de servidor)” en la página 588
- “GRANT (privilegios de espacio de tabla)” en la página 590
- “GRANT (Privilegios de secuencia)” en la página 586
- “GRANT (Privilegios de rutina)” en la página 579

GRANT (Privilegios de rutina)

Esta forma de la sentencia GRANT otorga privilegios para una rutina (función, método o procedimiento).

Invocación:

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

Autorización:

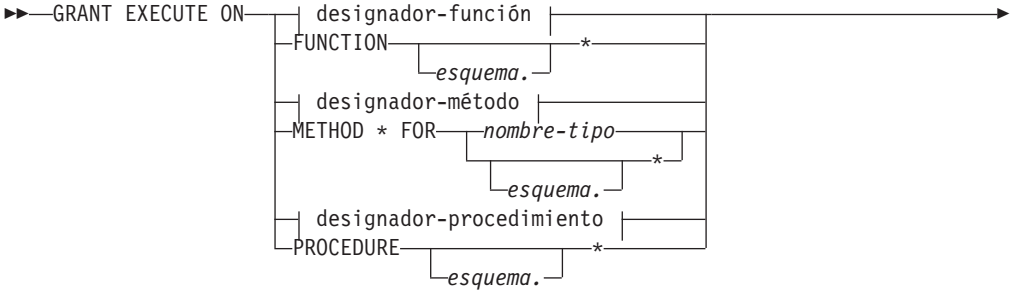
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

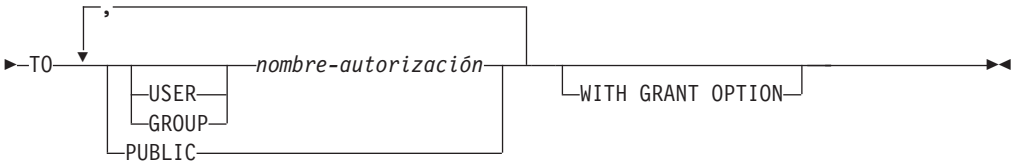
- WITH GRANT OPTION para EXECUTE para la rutina
- Autorización SYSADM o DBADM

Para otorgar todos los privilegios EXECUTE de rutina del esquema o tipo, los privilegios del ID de autorización deben incluir, como mínimo, uno de los siguientes:

- WITH GRANT OPTION para EXECUTE para todas las rutinas existentes y futuras (del tipo especificado) del esquema especificado
- Autorización SYSADM o DBADM

Sintaxis:





Descripción:

EXECUTE
Otorga el privilegio para ejecutar la función, el método o el procedimiento almacenado definido por el usuario que se identifica.

designador-función
Identifica la función de forma exclusiva.

FUNCTION *esquema*.*
Identifica todas las funciones del esquema, incluida cualquier función que pueda crearse en el futuro. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

designador-método
Identifica un método de forma exclusiva.

METHOD*
Identifica todos los métodos del tipo *nombre-tipo*, incluido cualquier método que pueda crearse en el futuro.

FOR *nombre-tipo*
Especifica el nombre del tipo en el que se encuentra el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el valor del registro especial CURRENT SCHEMA se utiliza como calificador de un nombre de tipo no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados. Para identificar todos los tipos del esquema, incluido cualquier tipo que pueda crearse en el futuro, puede utilizase un asterisco (*) en lugar del *nombre-tipo*.

designador-procedimiento
Identifica el procedimiento de forma exclusiva.

PROCEDURE *esquema*.*
Identifica todos los procedimientos del esquema, incluido cualquier procedimiento que pueda crearse en el futuro. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

TO
Especifica a quién se otorga el privilegio EXECUTE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

PUBLIC
Otorga el privilegio EXECUTE a todos los usuarios.

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados puedan otorgar (GRANT) el privilegio EXECUTE a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar el privilegio EXECUTE a otros usuarios si éstos:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar el privilegio EXECUTE desde alguna otra fuente.

Normas:

- No es posible otorgar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Si no se especifica USER ni GROUP, entonces:
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el *nombre-autorización* en el sistema operativo sólo como USER o si no esta definido, se supone USER.
  - Si el *nombre-autorización* se ha definido en el sistema operativo como USER y GROUP, se recibe un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA y no se han otorgado privilegios, se devolverá un mensaje de aviso (SQLSTATE 01007). Si el usuario que otorga privilegios no dispone de ningún privilegio sobre el objeto de la operación de concesión de privilegios, se devolverá un error (SQLSTATE 42501).

Ejemplos:

Ejemplo 1: Otorgue el privilegio EXECUTE para la función CALC_SALARY al usuario JONES. Se da por supuesto que, en el esquema, sólo existe una función con el nombre de función CALC_SALARY.

```
GRANT EXECUTE ON FUNCTION CALC_SALARY TO JONES
```

Ejemplo 2: Otorgue el privilegio EXECUTE para el procedimiento VACATION_ACCR a todos los usuarios del servidor actual.

```
GRANT EXECUTE ON PROCEDURE VACATION_ACCR TO PUBLIC
```

Ejemplo 3: Otorgue el privilegio EXECUTE para la función DEPT_TOTALS al ayudante de administración y otorgue al ayudante la capacidad de otorgar el privilegio EXECUTE para esta función a otros usuarios. La función tiene el nombre específico DEPT85_TOT. Se da por supuesto que el esquema tiene más de una función denominada DEPT_TOTALS.

```
GRANT EXECUTE ON SPECIFIC FUNCTION DEPT85_TOT
  TO ADMIN_A WITH GRANT OPTION
```

Ejemplo 4: Otorgue el privilegio EXECUTE para la función NEW_DEPT_HIRES a HR (Recursos humanos). La función tiene dos parámetros de entrada de tipo INTEGER y CHAR(10) respectivamente. Se da por supuesto que el esquema tiene más de una función denominada NEW_DEPT_HIRES.

```
GRANT EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10)) TO HR
```

Ejemplo 5: Otorgue el privilegio EXECUTE para el método SET_SALARY de tipo EMPLOYEE al usuario JONES.

```
GRANT EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE TO JONES
```

Información relacionada:

- "GRANT (Autorizaciones de base de datos)" en la página 570
- "GRANT (Privilegios de índice)" en la página 574
- "GRANT (Privilegios de paquete)" en la página 576

GRANT (Privilegios de rutina)

- "GRANT (Privilegios de esquema)" en la página 583
- "GRANT (Privilegios de tabla, vista o apodo)" en la página 592
- "GRANT (Privilegios de servidor)" en la página 588
- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "Elementos comunes de la sintaxis" en la página viii

GRANT (Privilegios de esquema)

Esta forma de la sentencia GRANT otorga privilegios en un esquema.

Invocación:

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

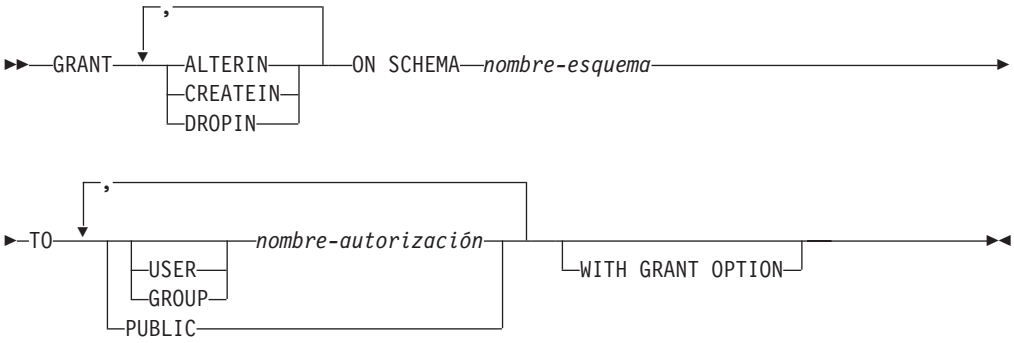
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- WITH GRANT OPTION para cada privilegio identificado de *nombre-esquema*
- Autorización SYSADM o DBADM

Ningún usuario puede otorgar privilegios en los nombres de esquema SYSIBM, SYSCAT, SYSFUN y SYSSTAT (SQLSTATE 42501).

Sintaxis:



Descripción:

ALTERIN
Otorga el privilegio para modificar o comentar todos los objetos del esquema. El propietario de un esquema creado explícitamente recibe automáticamente el privilegio ALTERIN.

CREATEIN
Otorga el privilegio para crear objetos en el esquema. Siguen necesitándose las demás autorizaciones o privilegios necesarios para crear el objeto (como CREATETAB). El propietario de un esquema creado explícitamente recibe automáticamente el privilegio CREATEIN. En un esquema creado implícitamente se otorga automáticamente el privilegio CREATEIN a PUBLIC.

DROPIN
Otorga el privilegio para eliminar todos los objetos del esquema. El propietario de un esquema creado explícitamente recibe automáticamente el privilegio DROPIN.

ON SCHEMA *nombre-esquema*
Identifica el esquema en el que se deben otorgar los privilegios.

TO
Especifica a quién se otorgan los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a todos los usuarios.

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

Si se omite WITH GRANT OPTION, los *nombres-autorización* sólo pueden otorgar los privilegios a otros si:

- tienen la autorización DBADM o
- han recibido la posibilidad de otorgar privilegios por otra fuente.

Normas:

- Si no se especifica USER ni GROUP, entonces
  - Si se define el nombre-autorización en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el nombre-autorización en el sistema operativo sólo como USER o si no está definido, se supone USER.
  - Si el nombre-autorización se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

Ejemplos:

Ejemplo 1: Otorgue al usuario JSINGLETON la posibilidad de crear objetos en el esquema CORPDATA.

```
GRANT CREATEIN ON SCHEMA CORPDATA TO JSINGLETON
```

Ejemplo 2: Otorgue al usuario IHAKES la posibilidad de crear y eliminar objetos en el esquema CORPDATA.

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO IHAKES
```

Información relacionada:

- "GRANT (Autorizaciones de base de datos)" en la página 570

- "GRANT (Privilegios de índice)" en la página 574
- "GRANT (Privilegios de paquete)" en la página 576
- "GRANT (Privilegios de tabla, vista o apodo)" en la página 592
- "GRANT (Privilegios de servidor)" en la página 588
- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "GRANT (Privilegios de rutina)" en la página 579

GRANT (Privilegios de secuencia)

Esta forma de la sentencia GRANT otorga privilegios para una secuencia.

Invocación:

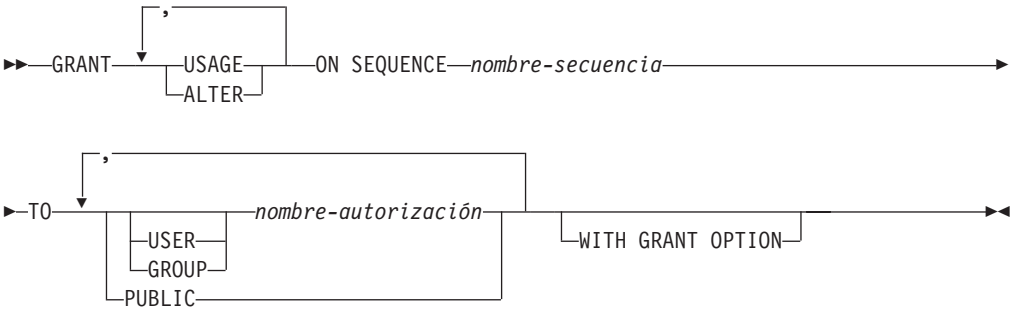
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- WITH GRANT OPTION para cada privilegio identificado en *nombre-secuencia*
- Autorización SYSADM o DBADM

Sintaxis:



Descripción:

USAGE
Otorga el privilegio para poder hacer referencia a una secuencia utilizando una *expresión-nextval* o *expresión-prevval*.

ALTER
Otorga el privilegio de modificar propiedades de secuencia utilizando la sentencia ALTER SEQUENCE.

ON SEQUENCE *nombre-secuencia*
Identifica la secuencia para la que van a otorgarse los privilegios especificados. El nombre de la secuencia, incluido un calificador de esquema implícito o explícito, debe identificar de forma exclusiva a una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704).

TO
Especifica a quién se otorgan los privilegios especificados.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

PUBLIC
Otorga los privilegios especificados a todos los usuarios.

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado pueda otorgar los privilegios indicados a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar los privilegios indicados a otros si:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar los privilegios especificados desde alguna otra fuente.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si el *nombre-autorización* está definido en el sistema operativo sólo como USER, o si no está definido, se supone USER.
  - Si el *nombre-autorización* se ha definido en el sistema operativo como USER y GROUP, se devuelve un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

Ejemplo:

Ejemplo 1: Otorgue a cualquier usuario el privilegio USAGE para una secuencia denominada ORG_SEQ.

```
GRANT USAGE ON SEQUENCE ORG_SEQ TO PUBLIC
```

Ejemplo 2: Otorgue al usuario BOBBY la capacidad de modificar una secuencia denominada GENERATE_ID y otorgar este privilegio a otros usuarios.

```
GRANT ALTER ON SEQUENCE GENERATE_ID TO BOBBY WITH GRANT OPTION
```

Información relacionada:

- “GRANT (Autorizaciones de base de datos)” en la página 570
- “GRANT (Privilegios de índice)” en la página 574
- “GRANT (Privilegios de paquete)” en la página 576
- “GRANT (Privilegios de esquema)” en la página 583
- “GRANT (Privilegios de tabla, vista o apodo)” en la página 592
- “GRANT (Privilegios de servidor)” en la página 588
- “GRANT (privilegios de espacio de tabla)” en la página 590
- “GRANT (Privilegios de rutina)” en la página 579

GRANT (Privilegios de servidor)

Este formato de la sentencia GRANT otorga el privilegio de acceder y utilizar una fuente de datos especificada en la modalidad de paso a través.

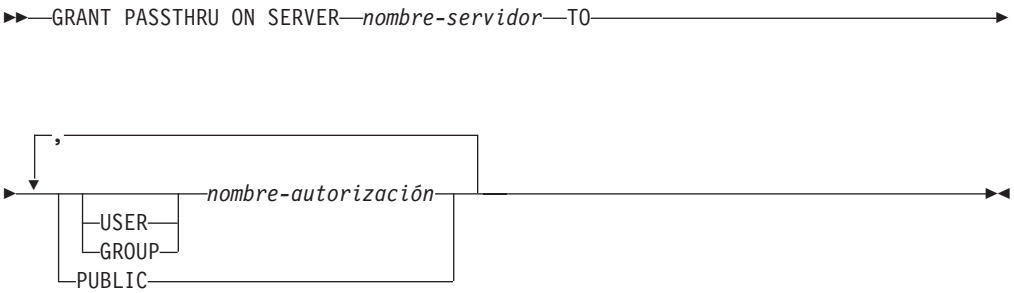
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

nombre-servidor
Designa la fuente de datos para la cual se está otorgando el privilegio que debe utilizarse en la modalidad de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

TO
Especifica a quién se otorga el privilegio.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga a todos los usuarios el privilegio de realizar un paso a través para *nombre-servidor.*

Ejemplos:

Ejemplo 1: Otorgue a R. Smith y a J. Jones el privilegio de paso a través para la fuente de datos SERVALL. Sus ID de autorización son RSMITH y JJONES.

```
   GRANT PASSTHRU ON SERVER SERVALL
TO USER RSMITH,
USER JJONES
```

Ejemplo 2: Otorgue el privilegio de realizar un paso a través para la fuente de datos EASTWING a un grupo cuyo ID de autorización es D024. Existe un usuario cuyo ID de autorización también es D024.

```
GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

Debe especificarse la palabra clave GROUP; de lo contrario se producirá un error porque D024 es el ID de un usuario y el ID del grupo especificado (SQLSTATE 56092). Cualquier miembro del grupo D024 tendrá permitido realizar un paso a través para EASTWING. Por lo tanto, si el usuario D024 pertenece al grupo, este usuario podrá realizar un paso a través para EASTWING.

Información relacionada:

- "GRANT (Autorizaciones de base de datos)" en la página 570
- "GRANT (Privilegios de índice)" en la página 574
- "GRANT (Privilegios de paquete)" en la página 576
- "GRANT (Privilegios de esquema)" en la página 583
- "GRANT (Privilegios de tabla, vista o apodo)" en la página 592
- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "GRANT (Privilegios de rutina)" en la página 579

GRANT (privilegios de espacio de tabla)

Esta forma de la sentencia GRANT otorga privilegios para un espacio de tablas.

Invocación:

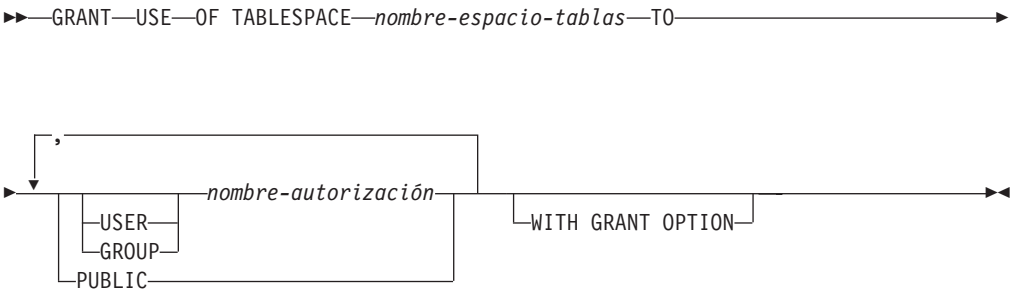
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- WITH GRANT OPTION para utilizar el espacio de tablas
- Autorización SYSADM, SYSCTRL o DBADM

Sintaxis:



Descripción:

USE
Otorga el privilegio para especificar, de forma explícita o por omisión, el espacio de tablas al crear una tabla. La opción GRANT otorga automáticamente el privilegio USE al creador de un espacio de tablas.

OF TABLESPACE *nombre-espacio-tablas*
Identifica el espacio de tablas para el que debe otorgarse el privilegio USE. El espacio de tablas no puede ser SYSCATSPACE (SQLSTATE 42838) ni un espacio de tablas temporal del sistema (SQLSTATE 42809).

TO
Especifica a quién se otorga el privilegio USE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga el privilegio USE a todos los usuarios.

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado otorgue el privilegio USE a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar el privilegio USE a otros usuarios si éstos:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar el privilegio USE desde alguna otra fuente.

Notas:

Si no se especifica USER ni GROUP, entonces

- Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
- Si el *nombre-autorización* está definido en el sistema operativo sólo como USER, o si no está definido, se supone USER.
- Si el *nombre-autorización* se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).

Ejemplos:

Ejemplo 1: Este ejemplo otorga al usuario BOBBY la capacidad para crear tablas en el espacio de tablas PLANS y para otorgar este privilegio a otros usuarios.

```
GRANT USE OF TABLESPACE PLANS TO BOBBY WITH GRANT OPTION
```

Información relacionada:

- “GRANT (Autorizaciones de base de datos)” en la página 570
- “GRANT (Privilegios de índice)” en la página 574
- “GRANT (Privilegios de paquete)” en la página 576
- “GRANT (Privilegios de esquema)” en la página 583
- “GRANT (Privilegios de tabla, vista o apodo)” en la página 592
- “GRANT (Privilegios de servidor)” en la página 588
- “GRANT (Privilegios de secuencia)” en la página 586
- “GRANT (Privilegios de rutina)” en la página 579

GRANT (Privilegios de tabla, vista o apodo)

Esta forma de la sentencia GRANT otorga privilegios en una tabla, vista o apodo.

Invocación:

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

Autorización:

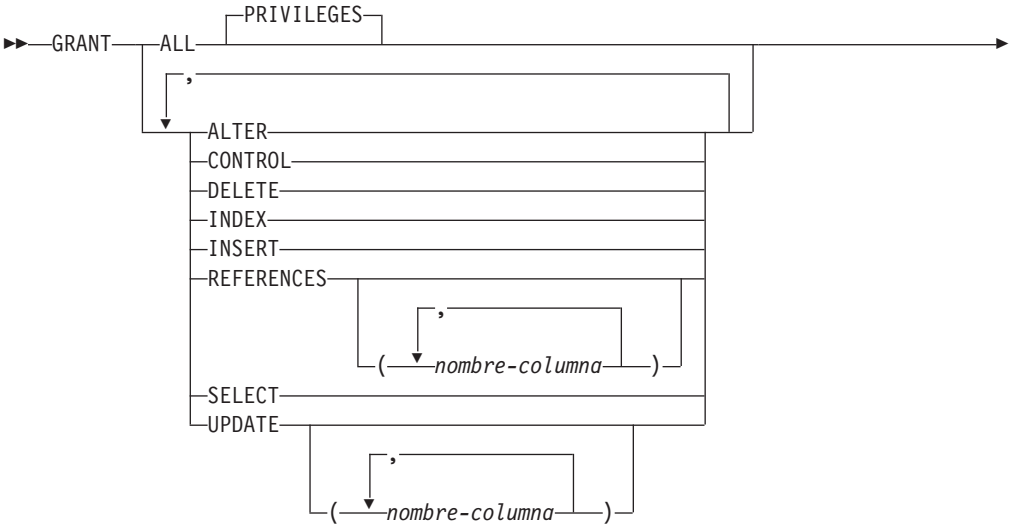
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

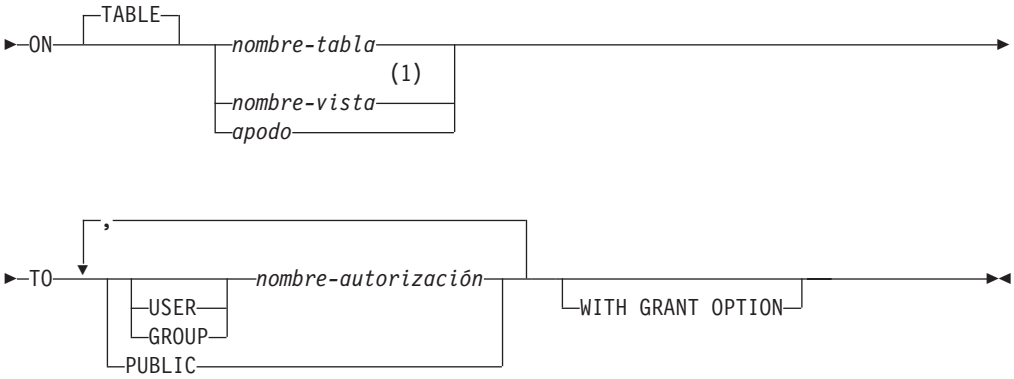
- Privilegio CONTROL para la tabla, vista o apodo referenciado.
- WITH GRANT OPTION para cada privilegio identificado. Si se especifica ALL, el ID de autorización debe tener algún privilegio otorgable en la tabla, vista o apodo identificado.
- Autorización SYSADM o DBADM.

Para otorgar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Para otorgar privilegios sobre tablas y vistas de catálogo son necesarias las autorizaciones SYSADM o DBADM.

Sintaxis:





Notas:

1 Los privilegios ALTER, INDEX y REFERENCES no son aplicables a las vistas.

Descripción:

ALL o **ALL PRIVILEGES**
Otorga todos los privilegios adecuados, excepto CONTROL, en la tabla base, vista o apodo llamado en la cláusula ON.

Si el ID de autorización de la sentencia tiene el privilegio CONTROL en la tabla, vista o apodo, o la autorización DBADM o SYSADM, entonces se otorgan todos los privilegios aplicables al objeto (excepto CONTROL). De lo contrario, los privilegios otorgados son todos los privilegios otorgables que el ID de autorización de la sentencia tenga en la tabla, vista o apodo identificado.

Si no se especifica ALL, debe especificarse una o varias palabras clave en la lista de privilegios.

ALTER
Otorga el privilegio para:

- Añadir columnas a una definición de tabla base.
- Crear o eliminar una clave primaria o una restricción de unicidad en una tabla base.
- Crear o eliminar una clave foránea en una tabla base.

  También es necesario el privilegio REFERENCES en cada columna de la tabla padre.
- Crear o eliminar una restricción de comprobación en una tabla base.
- Crear un activador en una tabla base.
- Añadir, restablecer o eliminar una opción de columna para un apodo.
- Cambiar un nombre de columna de apodo o tipo de datos.
- Añadir o cambiar un comentario en una tabla base o en un apodo.

CONTROL
Otorga:

- Todos los privilegios adecuados de la lista, es decir:
  - ALTER, CONTROL, DELETE, INSERT, INDEX, REFERENCES, SELECT y UPDATE para tablas base
  - CONTROL, DELETE, INSERT, SELECT y UPDATE para vistas
  - ALTER, CONTROL, INDEX y REFERENCES para apodos
- La posibilidad de otorgarlos privilegios anteriores (excepto CONTROL) a otros.

- La posibilidad de eliminar la tabla base, vista o apodo.

  Esta posibilidad no puede extenderse a otros sobre la base de poseer el privilegio CONTROL. La única manera que puede extenderse es otorgando el privilegio CONTROL en sí y esto sólo puede realizarlo alguien con la autorización SYSADM o DBADM.
- La posibilidad de ejecutar el programa de utilidad RUNSTATS en la tabla e índices.
- La posibilidad de ejecutar el programa de utilidad REORG en la tabla.
- La posibilidad de emitir la sentencia SET INTEGRITY para una tabla base, tabla de consultas materializadas o tabla de etapas.

La persona que define una tabla base, una tabla de consultas materializadas, una tabla de etapas o un apodo automáticamente recibe el privilegio CONTROL.

La persona que define una vista recibe automáticamente el privilegio CONTROL si posee el privilegio CONTROL en todas las tablas, vistas y apodos identificados en la selección completa.

DELETE
Otorga el privilegio para suprimir las filas de la tabla o vista actualizable.

INDEX
Otorga el privilegio para crear un índice en una tabla o una especificación de índice en un apodo. Este privilegio no se puede otorgar en una vista. El creador de un índice o de una especificación de índice tiene automáticamente el privilegio CONTROL en el índice o en la especificación de índice (que autoriza al creador a eliminar el índice o la especificación de índice). Así mismo, el creador mantiene el privilegio CONTROL incluso si se revoca el privilegio INDEX.

INSERT
Otorga el privilegio para insertar filas en la tabla o vista actualizable y para ejecutar el programa de utilidad IMPORT.

REFERENCES
Otorga el privilegio para crear y eliminar una clave foránea que haga referencia a la tabla como la tabla padre.

Si el ID de autorización de la sentencia tiene uno de los siguientes:

- Autorización DBADM o SYSADM
- Privilegio CONTROL para la tabla
- REFERENCES WITH GRANT OPTION para la tabla

entonces los usuarios autorizados pueden crear restricciones de referencia utilizando como clave padre todas las columnas de la tabla, incluso las que se han añadido después mediante la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son todos los privilegios REFERENCES de columna otorgables que el ID de autorización de la sentencia tiene en la tabla identificada.

El privilegio puede otorgarse para un apodo, aunque no pueden definirse claves foráneas para apodos de referencia.

REFERENCES (*nombre-columna***,...)**
Otorga el privilegio para crear y eliminar una clave foránea utilizando solamente las columnas especificadas en la lista de columnas como clave padre. Cada *nombre-columna* debe ser un nombre no calificado que identifique

una columna de la tabla identificada en la cláusula ON. El privilegio REFERENCES para columnas no puede otorgarse para tablas con tipo, vistas con tipo ni apodos (SQLSTATE 42997).

SELECT
Otorga el privilegio para:

- Recupera filas de la tabla o vista.
- Crea vistas en la tabla.
- Ejecuta el programa de utilidad EXPORT en la tabla o vista.

UPDATE
Otorga el privilegio para utilizar la sentencia UPDATE sobre la tabla o vista actualizable identificada en la cláusula ON.

Si el ID de autorización de la sentencia tiene uno de los siguientes:

- Autorización DBADM o SYSADM
- Privilegio CONTROL para la tabla o vista
- UPDATE WITH GRANT OPTION en la tabla o vista

entonces la persona o personas a las que se otorga pueden actualizar todas las columnas actualizables de la tabla o vista en las que la persona que otorga tiene el privilegio así como aquellas columnas que se han añadido después utilizando la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son los privilegios UPDATE de columna otorgables que el ID de autorización de la sentencia tiene en la tabla o vista identificada.

UPDATE (*nombre-columna***,...)**
Otorga el privilegio de utilizar la sentencia UPDATE para actualizar solamente las columnas especificadas en la lista de columnas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de la tabla o vista identificada en la cláusula ON. El privilegio UPDATE de nivel de columna no puede otorgarse en las tablas con tipo, vistas con tipo o apodos (SQLSTATE 42997).

ON TABLE *nombre-tabla* o *nombre-vista* o *apodo*
Especifica la tabla, vista o apodo en la que se deben otorgar los privilegios.

No puede otorgarse ningún privilegio para una vista no operativa o para una tabla de consultas materializadas no operativa (SQLSTATE 51024). No pueden otorgarse privilegios para una tabla temporal declarada (SQLSTATE 42995).

TO
Especifica a quién se otorgan los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos. (Se han eliminado las restricciones anteriores que se aplicaban a la concesión de los privilegios correspondientes al ID de autorización del usuario que emite la sentencia.)

Un privilegio que se ha otorgado a un grupo no se utiliza para la comprobación de autorizaciones:

- En las sentencias de DML estáticas de un paquete
- En una tabla base mientras se procesa una sentencia CREATE VIEW

- En una tabla base mientras se procesa una sentencia CREATE TABLE para una tabla de consultas materializadas

En DB2 Universal Database, los privilegios de tabla otorgados a grupos sólo se aplican a las sentencias que se preparan dinámicamente. Por ejemplo, si se ha otorgado el privilegio INSERT en la tabla PROJECT al grupo D204 pero no a UBIQUITY (un miembro de D204), UBIQUITY podría emitir la sentencia:

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

en la que el contenido de la serie es:

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

pero no podría precompilar ni enlazar un programa con la sentencia:

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

PUBLIC
Otorga los privilegios a todos los usuarios. (Se han eliminado las restricciones anteriores que se aplicaban a la utilización de los privilegios que se otorgan a PUBLIC para las sentencias de SQL estático y la sentencia CREATE VIEW.)

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios aplicables excepto CONTROL (SQLSTATE 01516).

Normas:

- Si no se especifica USER ni GROUP, entonces
  - Si se define el *nombre-autorización* en el sistema operativo sólo como GROUP, entonces se supone GROUP.
  - Si se define el *nombre-autorización* en el sistema operativo sólo como USER o si no esta definido, se supone USER.
  - Si el *nombre-autorización* se ha definido en el sistema operativo como ambos, se devuelve un error (SQLSTATE 56092).
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.) Si se ha especificado el privilegio CONTROL, sólo se otorgarán privilegios si el ID de autorización de la sentencia tiene autorización SYSADM o DBADM (SQLSTATE 42501).

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 UDB para OS/390 y z/OS:

  - Se tolera y se pasa por alto la sintaxis siguiente:
    - PUBLIC AT ALL LOCATIONS

- Los privilegios se pueden otorgar independientemente a cada nivel de una jerarquía de tablas. Un usuario con un privilegio sobre una supertabla puede afectar a las subtablas. Por ejemplo, una actualización que especifique la supertabla *T* puede mostrarse como un cambio en una fila en la subtabla *S* de *T* efectuado por un usuario con privilegio UPDATE sobre *T*, pero sin privilegio UPDATE sobre *S*. Un usuario sólo puede operar directamente en la subtabla si sostiene el privilegio necesario sobre la subtabla.
- Otorgar privilegios de apodo no tiene efecto sobre los privilegios de objeto de la fuente de datos (tabla o vista). Normalmente, la tabla o vista necesita privilegios de fuente de datos a los que un apodo hace referencia al intentar recuperar los datos.

Ejemplos:

Ejemplo 1: Otorgue todos los privilegios de la tabla WESTERN_CR a PUBLIC.

```
GRANT ALL ON WESTERN_CR
   TO PUBLIC
```

Ejemplo 2: Otorgue los privilegios adecuados de la tabla CALENDAR para que los usuarios PHIL y CLAIRE puedan leerla e insertar nuevas entradas en ella. No les permita cambiar ni eliminar ninguna de las entradas existentes.

```
GRANT SELECT, INSERT ON CALENDAR
   TO USER  PHIL, USER CLAIRE
```

Ejemplo 3: Otorgue todos los privilegios de la tabla COUNCIL al usuario FRANK y la posibilidad de extender todos los privilegios a otros.

```
GRANT ALL ON COUNCIL
   TO USER FRANK WITH GRANT OPTION
```

Ejemplo 4: Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un usuario llamado JOHN. Hay un usuario llamado JOHN y no hay ningún grupo llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

o

```
GRANT SELECT
  ON CORPDATA.EMPLOYEE TO USER JOHN
```

Ejemplo 5: Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un grupo llamado JOHN. Hay un grupo llamado JOHN y ningún usuario llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

o

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

Ejemplo 6: Otorgue (GRANT) los privilegios INSERT y SELECT en la tabla T1 a un grupo llamado D024 y a un usuario llamado D024.

```
GRANT INSERT, SELECT ON TABLE T1
  TO GROUP D024, USER D024
```

En este caso, tanto los miembros del grupo D024 como el usuario D024 tendrían permitido insertar (INSERT) y seleccionar (SELECT) en la tabla T1. También, se añadirían dos filas a la vista de catálogo SYSCAT.TABAUTH.

Ejemplo 7: Otorgue (GRANT) INSERT, SELECT y CONTROL en la tabla CALENDAR al usuario FRANK. FRANK debe poder pasar los privilegios a otros.

```
GRANT CONTROL ON TABLE CALENDAR
  TO FRANK WITH GRANT OPTION
```

El resultado de esta sentencia es un aviso (SQLSTATE 01516) de que no se ha dado WITH GRAN OPTION a CONTROL. Frank tiene ahora la posibilidad de otorgar cualquier privilegio para CALENDAR, inclusive INSERT y SELECT como era necesario. FRANK no puede otorgar CONTROL en CALENDAR a otros usuarios a menos que tenga la autorización SYSADM o DBADM.

Ejemplo 8: El usuario JON ha creado un apodo para una tabla Oracle que no tiene índice. El apodo es ORAREM1. Más tarde, Oracle DBA ha definido un índice para esta tabla. El usuario SHAWN ahora desea que DB2 sepa que este índice existe, de modo que el optimizador pueda idear estrategias para acceder a la tabla de un modo más eficaz. SHAWN puede informar a DB2 del índice creando una especificación de índice para ORAREM1. Otorgue a SHAWN el privilegio para este apodo, de modo que podrá crear la especificación de índice.

```
GRANT INDEX ON NICKNAME ORAREM1
  TO USER SHAWN
```

Información relacionada:

- "ALTER TABLE" en la página 46
- "GRANT (Autorizaciones de base de datos)" en la página 570
- "GRANT (Privilegios de índice)" en la página 574
- "GRANT (Privilegios de paquete)" en la página 576
- "GRANT (Privilegios de esquema)" en la página 583
- "GRANT (Privilegios de servidor)" en la página 588
- "GRANT (privilegios de espacio de tabla)" en la página 590
- "GRANT (Privilegios de secuencia)" en la página 586
- "GRANT (Privilegios de rutina)" en la página 579

Ejemplos relacionados:

- "tbpriv.sqc -- How to grant, display, and revoke privileges (C)"
- "tbpriv.sqC -- How to grant, display, and revoke privileges (C++)"
- "TbPriv.java -- How to grant, display and revoke privileges on a table (JDBC)"
- "TbPriv.sqlj -- How to grant, display and revoke privileges on a table (SQLj)"

IF

La sentencia IF selecciona una vía de ejecución de acuerdo con la evaluación de una condición.

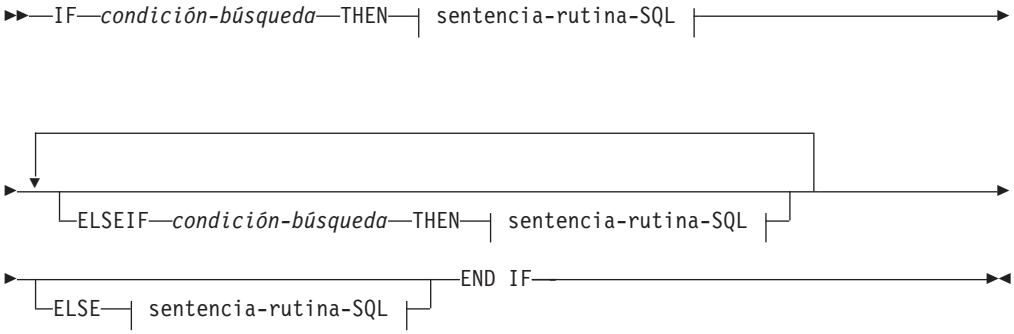
Invocación:

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

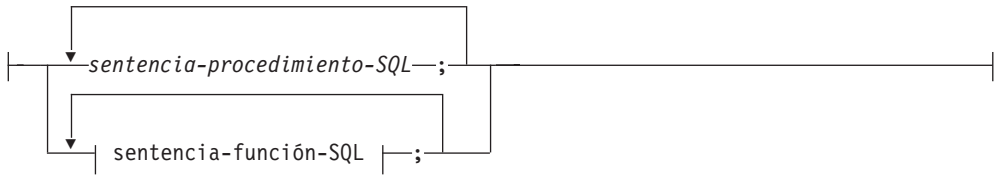
Autorización:

No se requieren privilegios para invocar una sentencia IF. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL y las condiciones de búsqueda incorporadas en la sentencia IF.

Sintaxis:



sentencia-rutina-SQL:



Descripción:

condición-búsqueda
Especifica la condición para la cual debe invocarse una sentencia de SQL. Si la condición es desconocida o falsa, el proceso continúa en la siguiente condición de búsqueda, hasta que una condición sea verdadera o el proceso llegue a la cláusula ELSE.

sentencia-procedimiento-SQL
Especifica la sentencia que debe invocarse si la *condición-búsqueda* anterior es verdadera. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

sentencia-función-SQL
Especifica la sentencia que debe invocarse si la *condición-búsqueda* anterior es verdadera. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL. Consulte sentencia-función-SQL en la descripción de la sentencia FOR.

Ejemplos:

El procedimiento de SQL siguiente acepta dos parámetros IN: un número de empleado (*employee_number*) y una tarifa de empleado (*rating*). Dependiendo del valor de *rating*, la tabla `employee` se actualiza con nuevos valores en las columnas `salary` (salario) y `bonus` (prima).

```
CREATE PROCEDURE UPDATE_SALARY_IF
  (IN employee_number CHAR(6), INOUT rating SMALLINT)
  LANGUAGE SQL
  BEGIN
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE EXIT HANDLER FOR not_found
      SET rating = -1;
    IF rating = 1
      THEN UPDATE employee
      SET salary = salary * 1.10, bonus = 1000
      WHERE empno = employee_number;
    ELSEIF rating = 2
      THEN UPDATE employee
      SET salary = salary * 1.05, bonus = 500
      WHERE empno = employee_number;
    ELSE UPDATE employee
      SET salary = salary * 1.03, bonus = 0
      WHERE empno = employee_number;
    END IF;
  END
```

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"

INCLUDE

La sentencia INCLUDE inserta declaraciones en un programa fuente.

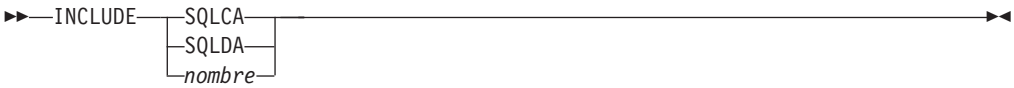
Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable.

Autorización:

No se necesita.

Sintaxis:



Descripción:

SQLCA
Indica que se debe incluir la descripción de un área de comunicaciones SQL (SQLCA).

SQLDA
Indica que se debe incluir la descripción de un área de descriptores SQL (SQLDA).

nombre
Identifica un archivo externo que contiene el texto que se debe incluir en el programa fuente que se está precompilando. Puede ser un identificador SQL sin ninguna extensión de nombre de archivo o un literal entre comillas simples (' '). Un identificador SQL asume la extensión del nombre de archivo del archivo fuente que se está precompilando. Si no se proporciona ninguna extensión de nombre de archivo mediante un literal entrecomillado, entonces no se asume ninguna.

Notas:

- Cuando se precompila un programa, la sentencia INCLUDE se sustituye por las sentencias fuente. Por lo tanto, la sentencia INCLUDE debe especificarse en un punto del programa en el que las sentencias fuente resultantes sean aceptables para el compilador.
- El archivo fuente externo debe estar escrito en el lenguaje principal especificado por el *nombre*. Si es superior a 18 caracteres o contiene caracteres que no están permitidos en un identificador SQL, debe ir entrecomillado. Las sentencias INCLUDE *nombre* pueden anidarse, aunque no cíclicamente (por ejemplo, si A y B son módulos y A contiene una sentencia INCLUDE *nombre*, no es válido que A llame a B y que, a continuación, B llame a A).
- Si la opción de precompilación LANGLEVEL se establece en el valor SQL92E, no debe especificarse INCLUDE SQLCA. Las variables SQLSTATE y SQLCODE pueden estar definidas en la sección de declaraciones de variables de lenguaje principal.

Ejemplo:

Incluya una SQLCA en un programa C.

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

mientras (SQLCODE==0) {
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

(Imprimir resultados)

}

EXEC SQL  CLOSE C1;
```

Información relacionada:

- "SQLDA (área de descriptores de SQL)" en la publicación *Consulta de SQL, Volumen 1*
- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbcfg.sqc -- Configure database and database manager configuration parameters (C)"
- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"
- "dtformat.sqc -- Load and import data format extensions (C)"
- "tbcreate.sqc -- How to create and drop tables (C)"
- "tbident.sqc -- How to use identity columns (C)"
- "dbcfg.sqC -- Configure database and database manager configuration parameters (C++)"
- "tbcreate.sqC -- How to create and drop tables (C++)"

INSERT

La sentencia INSERT inserta filas en una tabla, apodo o vista, o en las tablas, apodos o vistas subyacentes de la selección completa especificada. La inserción de una fila en un apodo inserta la fila en el objeto de fuente de datos al que hace referencia el apodo. La inserción de una fila en una vista también inserta la fila en la tabla en la que se basa la vista, si no se ha definido ningún activador INSTEAD OF para la operación de inserción en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Para ejecutar esta sentencia, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- Privilegio INSERT para la tabla, vista o apodo donde van a insertarse filas
- Privilegio CONTROL para la tabla, vista o apodo donde van a insertarse filas
- Autorización SYSADM o DBADM.

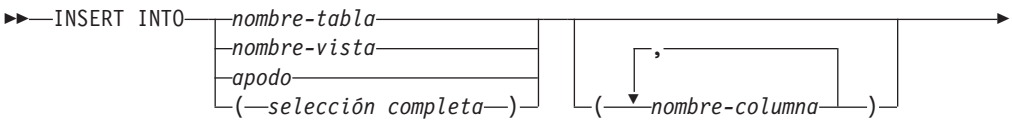
Además, para cada tabla, vista o apodo al que se haga referencia en cualquier selección completa utilizada en la sentencia INSERT, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

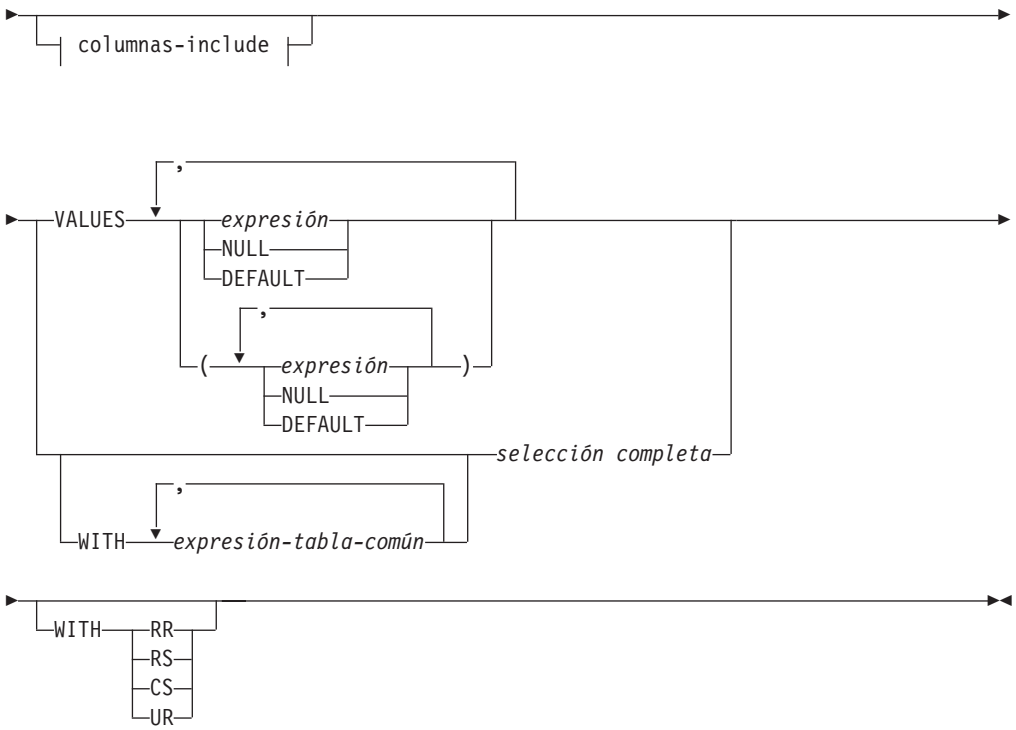
- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

No se comprueban los privilegios GROUP para las sentencias INSERT estáticas.

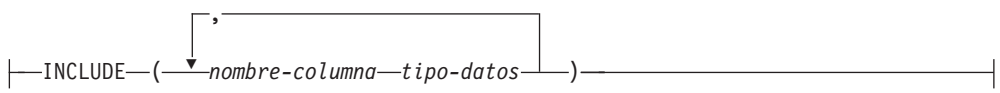
Si el destino de la operación de inserción es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis:





columnas-include:



Descripción:

INTO *nombre-tabla*, *nombre-vista*, *apodo* **o** *(selección completa)*
Identifica el objeto de la operación de inserción. El nombre debe identificar una tabla, vista o apodo que exista en el servidor de aplicaciones, pero no debe identificar una tabla de catálogo, una tabla de consultas materializadas mantenida por el sistema, una vista de una tabla de catálogo o una vista de sólo lectura, a menos que se haya definido un activador INSTEAD OF para la operación de inserción en la vista sujeto. Las filas que se insertan en un apodo se colocan en el objeto de fuente de datos al que hace referencia el apodo.

Si el objeto de la operación de inserción es una selección completa, esta debe ser insertable, según lo definido en el elemento de Notas "Vistas insertables" de la descripción de la sentencia CREATE VIEW.

Si no existe ningún activador INSTEAD OF para la operación de inserción en esta vista, no podrá insertarse un valor en una columna de vista que se haya obtenido de:

- Una constante, expresión o función escalar
- La misma columna de tabla base que otra columna de la vista.

Si el objeto de la operación de inserción es una vista con dichas columnas, debe especificarse una lista de nombres de columna y dicha lista no debe identificar estas columnas.

Una fila podrá insertarse en una vista o una selección completa que se ha definido utilizando UNION ALL si la fila satisface las restricciones de comprobación de exactamente una de las tablas base subyacentes. Si una fila satisface las restricciones de comprobación de más de una tabla, o no satisface la de ninguna tabla, se devolverá un error (SQLSTATE 23513).

(*nombre-columna***,...)**
Especifica las columnas para las que se proporcionan valores de inserción. Cada nombre debe ser un nombre no calificado que identifica una columna de la tabla o vista o una columna de la selección completa.. La misma columna no se puede identificar más de una vez. No puede identificarse una columna que no pueda aceptar la inserción de valores (por ejemplo, una columna basada en una expresión).

La omisión de la lista de columnas es una especificación implícita de una lista en la que cada columna de la tabla o vista, o cada elemento de la lista de selección de la selección completa se identifica en orden de izquierda a derecha. Esta lista se establece cuando se prepara la sentencia y, por lo tanto, no incluye las columnas que se han añadido a la tabla después de preparar la sentencia.

columnas-include
Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia INSERT cuando está anidada en la cláusula FROM de una selección completa. Las *columnas-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

INCLUDE
Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia INSERT. Esta cláusula sólo se puede especificar si la sentencia INSERT está anidada en la cláusula FROM de la selección completa.

nombre-columna
Especifica una columna de la tabla de resultados intermedia de la sentencia INSERT. El nombre no puede coincidir con el nombre de otra columna include ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

tipo-datos
Especifica el tipo de datos de la columna include. El tipo de datos debe ser uno que reciba soporte de la sentencia CREATE TABLE.

VALUES
Introduce una o varias filas de valores que se deben insertar.

Cada variable del lenguaje principal nombrada debe estar descrita en el programa de acuerdo con las normas para la declaración de variables del lenguaje principal.

El número de valores para cada fila debe ser igual al número de nombres de la lista de columnas implícita o explícita y de las columnas identificadas en la cláusula INCLUDE. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

expresión
Una *expresión* puede ser cualquier expresión que esté definida en "Expresiones".

NULL
Especifica el valor nulo y sólo debe especificarse para las columnas con posibilidad de nulos.

DEFAULT
Especifica que se debe utilizar el valor por omisión. El resultado de especificar DEFAULT depende del modo en que se haya definido la columna, del siguiente modo:

- Si la columna se definió como columna generada basándose en una expresión, el sistema genera el valor de la columna de acuerdo con esa expresión.
- Si se utiliza la cláusula IDENTITY, el gestor de bases de datos genera el valor.
- Si se utiliza la cláusula WITH DEFAULT, el valor insertado será el valor tal como se ha definido para la columna (consulte *cláusula-por-omisión* en "CREATE TABLE").
- Si se utiliza la cláusula NOT NULL y no la cláusula GENERATED, o no se utiliza WITH DEFAULT o se utiliza DEFAULT NULL, no se puede especificar la palabra clave DEFAULT para esa columna (SQLSTATE 23502).
- Cuando la inserción se realice en un apodo, la palabra clave DEFAULT se pasará por medio de la sentencia INSERT a la fuente de datos sólo si la fuente de datos da soporte a la palabra clave DEFAULT en su sintaxis de lenguaje de consulta.

WITH *expresión-tabla-común*
Define una expresión de tabla común para utilizarla con la selección completa que va a continuación.

selección completa
Especifica un conjunto de filas nuevas en la forma de la tabla de resultados de una selección completa. Puede haber una, más de una o ninguna. Si la tabla de resultados está vacía, SQLCODE se establece en +100 y SQLSTATE se establece en '02000'.

Cuando el objeto base de INSERT y el objeto base de la selección completa o cualquier subconsulta de la selección completa, son la misma tabla, la selección completa se evalúa por completo antes de insertar alguna fila.

El número de columnas de la tabla de resultados debe ser igual al número de nombres de la lista de columnas. El valor de la primera columna del resultado se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

WITH
Especifica el nivel de aislamiento en el que se ejecuta la selección completa (fullselect).

RR
Lectura repetible

RS
Estabilidad de lectura

CS
Estabilidad del cursor

UR
Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

Normas:

- ***Activadores:*** Las sentencias INSERT pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores insertados. Si una operación de inserción en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se han realizado en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.
- ***Valores por omisión:*** El valor insertado en cualquier columna que no esté en la lista de columnas es el valor por omisión de la columna o nulo. Las columnas que no permiten valores nulos y no están definidas con NOT NULL WITH DEFAULT deben incluirse en la lista de columnas. De manera similar, si se inserta en una vista, el valor insertado en cualquier columna de la tabla base que no esté en la vista es el valor por omisión de la columna o nulo. De ahí que todas las columnas de la tabla base que no estén en la vista deben ser un valor por omisión o permitir valores nulos. El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).
- ***Longitud:*** Si el valor de inserción de una columna es un número, la columna debe ser una columna numérica con la capacidad de representar la parte entera del número. Si el valor de inserción de una columna es una serie, la columna debe ser una columna de serie con un atributo de longitud que como mínimo sea tan grande como la longitud de la serie, o una columna de indicación de fecha y hora si la serie representa una fecha, hora o indicación de la hora.
- ***Asignación:*** Los valores de inserción se asignan a las columnas de acuerdo con normas de asignación específicas.
- ***Validez:*** Si la tabla nombrada, o la tabla base de la vista nombrada, tiene uno o varios índices de unicidad, cada fila insertada en la tabla debe ajustarse a las restricciones impuestas por dichos índices. Si se nombra una vista cuya definición incluye WITH CHECK OPTION, cada fila insertada en la vista debe ajustarse a la definición de la vista. Para obtener información acerca de las normas que rigen esta situación, consulte “CREATE VIEW”.
- ***Integridad de referencia:*** Para cada restricción definida en una tabla, cada valor de inserción que no sea nulo de la clave foránea debe ser igual al valor de clave primaria de la tabla padre.
- ***Restricción de comprobación:*** Los valores de inserción deben cumplir las condiciones de control de las restricciones de comprobación definidas en la tabla. En una sentencia INSERT para una tabla con restricciones de comprobación definidas, se evalúan las condiciones de restricción una vez para cada fila que se inserta.
- ***Enlaces de datos:*** Las sentencias de inserción que incluyan valores DATALINK darán como resultado un intento de enlace del archivo si se incluye un valor de URL (ni una serie vacía ni espacios en blanco) y se define la columna con FILE LINK CONTROL. Los errores producidos en el valor DATALINK o en el enlace del archivo provocarán que falle la inserción (SQLSTATE 428D1 o 57050).

Notas:

- Tras la ejecución de una sentencia INSERT, el valor de la tercera variable de la parte SQLERRD(3) de la SQLCA indica el número de filas que se han pasado a la operación de inserción. En el contexto de una sentencia de procedimiento de

SQL, el valor puede recuperarse utilizando la variable ROW_COUNT de la sentencia GET DIAGNOSTICS. SQLERRD(5) contiene la cuenta de todas las operaciones de inserción, actualización y supresión activadas.

- Salvo que ya existan los bloqueos adecuados, se adquieren uno o varios bloqueos exclusivos durante la ejecución de una sentencia INSERT satisfactoria. Hasta que se liberen los bloqueos, sólo se puede acceder a una fila insertada:
  - El proceso de aplicación que ha realizado la inserción.
  - Otro proceso de aplicación que utilice el nivel de aislamiento UR a través del cursor de sólo lectura, la sentencia SELECT INTO o la subselección utilizada en una subconsulta.
- Para obtener más información acerca del bloqueo, consulte la descripción de las sentencias COMMIT, ROLLBACK y LOCK TABLE.
- Si una aplicación se ejecuta en una base de datos particionada y se enlaza con la opción INSERT BUF, las sentencias INSERT con VALUES que no se procesan utilizando EXECUTE IMMEDIATE pueden ponerse en el almacenamiento intermedio. DB2 supone que tales sentencias INSERT se procesan dentro de un bucle de la lógica de la aplicación. En lugar de ejecutar la sentencia hasta que se completa, intenta almacenar los nuevos valores de fila en uno o varios almacenamientos intermedios. Como resultado las inserciones reales de las filas en la tabla se efectúan posteriormente, de manera asíncrona con la lógica de INSERT de la aplicación. Tenga en cuenta que esta inserción asíncrona puede generar un error relacionado con una sentencia INSERT que se devuelva a otra sentencia de SQL que siga a INSERT en la aplicación.

  Esto tiene la posibilidad de mejorar significativamente el rendimiento de INSERT, pero se utiliza mejor con datos limpios, debido a la naturaleza asíncrona del manejo de errores.
- Cuando se inserta una fila en una tabla que tiene una columna de identidad, DB2 genera un valor para la columna de identidad.
  - Para una columna de identidad definida como GENERATED ALWAYS, DB2 genera siempre el valor.
  - Para una columna definida como GENERATED BY DEFAULT, si no se especifica explícitamente un valor (con una cláusula VALUES o subselección), DB2 genera un valor.

  El primer valor generado por DB2 es el valor de la especificación START WITH para la columna de identidad.
- Cuando se ha insertado un valor para una columna de identidad de tipo diferenciado definida por el usuario, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior al tipo de fuente antes de realizarse el cálculo.)
- Cuando se inserta en una columna de identidad definida como GENERATED ALWAYS, DB2 genera siempre un valor para la columna, y el usuario no debe especificar un valor durante la inserción. Si una columna de identidad definida como GENERATED ALWAYS aparece en la lista de columnas de una sentencia INSERT, y la cláusula VALUES contiene un valor distinto del valor por omisión, se produce un error (SQLSTATE 428C9).

  Por ejemplo, suponga que EMPID es una columna de identidad definida como GENERATED ALWAYS, entonces el mandato:

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
  VALUES (:hv_valid_emp_id, :hv_name, :hv_addr)
```

  da como resultado un error.

- Cuando se inserta en una columna definida como GENERATED BY DEFAULT, DB2 permite especificar un valor real para la columna dentro de la cláusula VALUES o en una subselección. Sin embargo, cuando se especifica un valor en la cláusula VALUES, DB2 no realiza ninguna verificación del valor. Para asegurar la unicidad de los valores, se debe crear un índice de unicidad para la columna de identidad.

  Cuando se inserta en una tabla que tiene una columna de identidad definida como GENERATED BY DEFAULT, y no se especifica una lista de columnas, la cláusula VALUES puede especificar la palabra clave DEFAULT para representar el valor de la columna de identidad. DB2 generará el valor para la columna de identidad.

  ```
  INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
    VALUES (DEFAULT, :hv_name, :hv_addr)
  ```

  En este ejemplo, EMPID está definido como columna de identidad, y por tanto el valor insertado en esta columna es generado por DB2.

- La normas para insertar en una columna de identidad utilizando una subselección son similares a las normas para una inserción mediante una cláusula VALUES. Sólo se puede especificar un valor para una columna de identidad si ésta está definida como GENERATED BY DEFAULT.

  Por ejemplo, suponga que T1 y T2 son tablas que tienen la misma definición, y ambas contienen las columnas *intcol1* e *identcol2* (las dos son de tipo INTEGER y la segunda columna tiene el atributo de identidad). Considere la inserción siguiente:

  ```
  INSERT INTO T2
    SELECT *
    FROM T1
  ```

  Este ejemplo es conceptualmente equivalente a:

  ```
  INSERT INTO T2 (intcol1,identcol2)
    SELECT intcol1, identcol2
    FROM T1
  ```

  En ambos casos, la cláusula INSERT proporciona un valor explícito para la columna de identidad de T2. Esta especificación explícita puede dar un valor para la columna de identidad, pero la columna de identidad de T2 debe estar definida como GENERATED BY DEFAULT. De lo contrario se produce un error (SQLSTATE 428C9).

  Si una tabla tiene una columna definida como GENERATED ALWAYS, todavía es posible propagar todas las demás columnas de una tabla que tenga la misma definición. Por ejemplo, dadas las tablas T1 y T2 descritas anteriormente, se pueden propagar los valores intcol1 desde T1 a T2, mediante el SQL siguiente:

  ```
  INSERT INTO T2 (intcol1)
    SELECT intcol1
    FROM T1
  ```

  Observe que, debido a que identcol2 no está especificado en la lista de columnas, se le proporcionará su valor por omisión (generado).

- Cuando se inserta una fila en una tabla de una sola columna y ésta es una columna de identidad definida como GENERATED ALWAYS, es posible especificar un valor en VALUES mediante la palabra clave DEFAULT. En este caso, la aplicación no proporciona ningún valor para la tabla, y DB2 genera el valor para la columna de identidad.

  ```
  INSERT INTO IDTABLE
    VALUES(DEFAULT)
  ```

En la tabla del ejemplo anterior, formada por una sola columna que tiene el atributo de identidad, para insertar varias filas con una única sentencia INSERT puede utilizarse esta sentencia:

```
INSERT INTO IDTABLE
  VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
```

- Cuando DB2 genera un valor para una columna de identidad, ese valor generado caduca; la próxima vez que sea necesario un valor, DB2 generará uno nuevo. Esto es válido aunque falle o se cancele una sentencia INSERT en la que interviene una columna de identidad.

  Por ejemplo, suponga que se ha creado un índice de unicidad para la columna de identidad. Si al generar un valor para una columna de identidad se detecta una violación de clave duplicada, se produce un error (SQLSTATE 23505) y se considera que el valor generado para la columna de identidad ha caducado. Esto puede ocurrir si la columna de identidad está definida como GENERATED BY DEFAULT y el sistema intenta generar un nuevo valor, pero el usuario ha especificado explícitamente valores para la columna de identidad en sentencias INSERT anteriores. En este caso, el volver a emitir la misma sentencia INSERT puede producir un resultado satisfactorio. DB2 generará el valor siguiente para la columna de identidad y es posible que este valor siguiente sea exclusivo, y que la sentencia INSERT tenga éxito.
- Si al generar un valor para una columna de identidad se excede el valor máximo de la columna (o el valor mínimo en el caso de una secuencia descendente), se produce un error (SQLSTATE 23522). En este caso, el usuario debe eliminar la tabla y crear una nueva con una columna de identidad que tenga un rango mayor (es decir, cambiar el tipo de datos o valor de incremento de la columna para permitir un rango mayor de valores).

  Por ejemplo, una columna de identidad puede haberse definido con el tipo de datos SMALLINT, y posteriormente agotarse los valores que se pueden asignar a la columna. Para redefinir la columna de identidad como INTEGER, es necesario descargar los datos, eliminar la tabla y volver a crearla con una nueva definición para la columna, y luego cargar los datos de nuevo. Cuando se redefine la tabla, es necesario especificar un valor START WITH para la columna de identidad, para el que próximo valor generado por DB2 sea el valor que sigue a continuación en la secuencia original. Para determinar el valor final, emita una consulta utilizando el valor MAX de la columna de identidad (para una secuencia ascendente) o el valor MIN (para una secuencia descendente), antes de descargar los datos.

Ejemplos:

Ejemplo 1: Inserte un nuevo departamento con las siguientes especificaciones en la tabla DEPARTMENT:

- El número de departamento (DEPTNO) es 'E31'
- El nombre de departamento (DEPTNAME) es 'ARCHITECTURE'
- Dirigido por (MGRNO) una persona con el número '00390'
- Informa al departamento (ADMRDEPT) 'E01'.

```
INSERT INTO DEPARTMENT
  VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

Ejemplo 2: Inserte un nuevo departamento en la tabla DEPARTMENT como en el ejemplo 1, pero no asigne ningún director al nuevo departamento.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('E31', 'ARCHITECTURE', 'E01')
```

Ejemplo 3: Inserte dos nuevos departamentos utilizando una sentencia en la tabla DEPARTMENT como en el ejemplo 2, pero no asigne ningún director al nuevo departamento.

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('B11', 'PURCHASING', 'B01'),
         ('E41', 'DATABASE ADMINISTRATION', 'E01')
```

Ejemplo 4: Cree una tabla temporal MA_EMP_ACT con las mismas columnas que la tabla EMP_ACT. Cargue MA_EMP_ACT con las filas de la tabla EMP_ACT con un nuevo número de proyecto (PROJNO) que empieza por las letras 'MA'.

```
CREATE TABLE MA_EMP_ACT
          ( EMPNO CHAR(6)  NOT NULL,
            PROJNO CHAR(6)  NOT NULL,
            ACTNO SMALLINT  NOT NULL,
            EMPTIME DEC(5,2),
            EMSTDATE DATE,
            EMENDATE  DATE )
INSERT INTO MA_EMP_ACT
  SELECT * FROM EMP_ACT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

Ejemplo 5: Utilice una sentencia del programa C para añadir un esqueleto de proyecto a la tabla PROJECT. Obtenga el número de proyecto (PROJNO), nombre de proyecto (PROJNAME), número de departamento (DEPTNO) y empleado responsable (RESPEMP) de las variables del lenguaje principal. Utilice la fecha actual como la fecha de inicio del proyecto (PRSTDATE). Asigne un valor NULL a las restantes columnas de la tabla.

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
  VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

Ejemplo 6: Especifique una sentencia INSERT como *referencia-tabla-cambio-datos* dentro de una sentencia SELECT. Defina una columna include adicional cuyos valores se especifican en la cláusula VALUES, que luego se utiliza como una columna de clasificación para las filas insertadas.

```
SELECT inorder.ordernum
  FROM (INSERT INTO orders(custno)INCLUDE (insertnum integer)
    VALUES(:cnum1, 1), (:cnum2, 2)) InsertedOrders
  ORDER BY insertnum;
```

Información relacionada:

- "Expresiones" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE TABLE" en la página 341
- "CREATE VIEW" en la página 466
- "Consultas de SQL" en la publicación *Consulta de SQL, Volumen 1*
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtlob.sqc -- How to use the LOB data type (C)"
- "tbident.sqc -- How to use identity columns (C)"
- "tbmod.sqc -- How to modify table data (C)"
- "tbtrig.sqc -- How to use a trigger on a table (C)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "tbmod.sqC -- How to modify table data (C++)"
- "tbtrig.sqC -- How to use a trigger on a table (C++)"
- "DtLob.java -- How to use LOB data type (JDBC)"

- "TbIdent.java -- How to use Identity Columns (JDBC)"
- "TbMod.java -- How to modify table data (JDBC)"
- "TbTrig.java -- How to use triggers (JDBC)"
- "TbIdent.sqlj -- How to use Identity Columns (SQLj)"
- "TbMod.sqlj -- How to modify table data (SQLj)"
- "TbTrig.sqlj -- How to use triggers (SQLj)"
- "updat.sqb -- How to update, delete and insert table data (MF COBOL)"

ITERATE

La sentencia ITERATE hace que el flujo de control vuelva al principio de un bucle con etiqueta.

Invocación:

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

Autorización:

No se necesita.

Sintaxis:

Descripción:

etiqueta
Especifica la etiqueta de la sentencia FOR, LOOP, REPEAT o WHILE a la cual DB2 transfiere el flujo de control.

Ejemplos:

Este ejemplo utiliza un cursor para devolver información para un nuevo departamento. Si se ha invocado el descriptor de contexto de condiciones *not_found*, el flujo de control sale del bucle. Si el valor de *v_dept* es 'D11', una sentencia ITERATE vuelve a pasar el flujo del control al principio de la sentencia LOOP. En otro caso, se inserta una nueva fila en la tabla DEPARTMENT.

```
CREATE PROCEDURE ITERATOR()
  LANGUAGE SQL
  BEGIN
    DECLARE v_dept CHAR(3);
    DECLARE v_deptname VARCHAR(29);
    DECLARE v_admdept CHAR(3);
    DECLARE at_end INTEGER DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE c1 CURSOR FOR
      SELECT deptno, deptname, admrdept
      FROM department
      ORDER BY deptno;
    DECLARE CONTINUE HANDLER FOR not_found
      SET at_end = 1;
    OPEN c1;
    ins_loop:
    LOOP
      FETCH c1 INTO v_dept, v_deptname, v_admdept;
      IF at_end = 1 THEN
        LEAVE ins_loop;
      ELSEIF v_dept = 'D11' THEN
        ITERATE ins_loop;
      END IF;
      INSERT INTO department (deptno, deptname, admrdept)
      VALUES ('NEW', v_deptname, v_admdept);
    END LOOP;
    CLOSE c1;
  END
```

LEAVE

La sentencia LEAVE transfiere el control del programa hacia fuera de un bucle o de una sentencia compuesta.

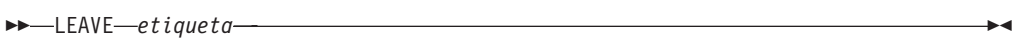
Invocación:

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

Autorización:

No se necesita.

Sintaxis:



Descripción:

etiqueta
Especifica la etiqueta de la sentencia FOR, LOOP, REPEAT, WHILE o sentencia compuesta cuya ejecución debe concluir.

Notas:

- Cuando una sentencia LEAVE transfiere el control hacia fuera de una sentencia compuesta, se cierran todos los cursores abiertos de la sentencia compuesta, excepto los cursores utilizados para devolver conjuntos de resultados.

Ejemplos:

Este ejemplo contiene un bucle que lee datos para el cursor *c1*. Si el valor de la variable de SQL *at_end* no es cero, la sentencia LEAVE transfiere el control fuera del bucle.

```
CREATE PROCEDURE LEAVE_LOOP(OUT counter INTEGER)
     LANGUAGE SQL
     BEGIN
    DECLARE v_counter INTEGER;
    DECLARE v_firstnme VARCHAR(12);
    DECLARE v_midinit CHAR(1);
    DECLARE v_lastname VARCHAR(15);
    DECLARE at_end SMALLINT DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
       DECLARE c1 CURSOR FOR
          SELECT firstnme, midinit, lastname
             FROM employee;
    DECLARE CONTINUE HANDLER for not_found
      SET at_end = 1;
    SET v_counter = 0;
       OPEN c1;
    fetch_loop:
        LOOP
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      IF at_end <> 0 THEN LEAVE fetch_loop;
           END IF;
      SET v_counter = v_counter + 1;
```

```
END LOOP fetch_loop;
SET counter = v_counter;
   CLOSE c1;
    END
```

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"

LOCK TABLE

La sentencia LOCK TABLE impide que procesos de aplicación simultáneos utilicen o cambien una tabla.

Invocación:

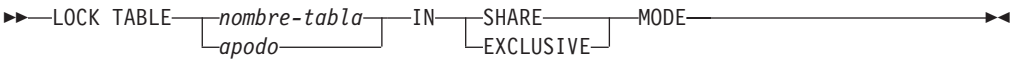
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio SELECT para la tabla
- Privilegio CONTROL para la tabla
- Autorización SYSADM o DBADM.

Sintaxis:



Descripción:

nombre-tabla **o** *apodo*
Identifica la tabla o el apodo. El *nombre-tabla* debe identificar una tabla que exista en el servidor de aplicaciones, pero no debe identificar una tabla de catálogo. No puede ser una tabla temporal declarada (SQLSTATE 42995). Si *nombre-tabla* es una tabla con tipo, debe ser la tabla raíz de la jerarquía de tablas (SQLSTATE 428DR). Cuando se ha especificado un apodo, DB2 bloquea el objeto subyacente (es decir, una tabla o una vista) de la fuente de datos a la que el apodo hace referencia.

IN SHARE MODE
Impide que procesos de aplicación simultáneos ejecuten alguna operación que no sea de sólo lectura en la tabla.

IN EXCLUSIVE MODE
Impide a los procesos de aplicación simultáneos ejecutar cualquier operación en la tabla. Tenga en cuenta que EXCLUSIVE MODE no impide que los procesos de aplicación simultáneos que estén ejecutando en el nivel de aislamiento Lectura no confirmada (UR) ejecuten operaciones de sólo lectura en la tabla.

Notas:

- El bloqueo se utiliza para evitar operaciones simultáneas. No se adquiere necesariamente un bloqueo durante la ejecución de la sentencia LOCK TABLE si ya existe un bloqueo satisfactorio. El bloqueo que impide operaciones simultáneas se conserva como mínimo hasta la terminación de la unidad de trabajo.
- En una base de datos particionada, primero se adquiere un bloqueo de tabla en la primera partición del grupo de particiones de base de datos (la partición que tiene el número más bajo) y, después, en las demás particiones. Si se interrumpe la sentencia LOCK TABLE, la tabla puede estar bloqueada en algunas particiones

y en otras no. Si ocurre esto, emita otra sentencia LOCK TABLE para completar el bloqueo de todas las particiones o emita una sentencia COMMIT o ROLLBACK para liberar los bloqueos actuales.

- Esta sentencia afecta a todas las particiones del grupo de particiones de base de datos.

Ejemplo:

Obtenga un bloqueo en la tabla EMP. No permita que otros programas lean o actualicen la tabla.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

LOOP

La sentencia LOOP repite la ejecución de una sentencia o grupo de sentencias.

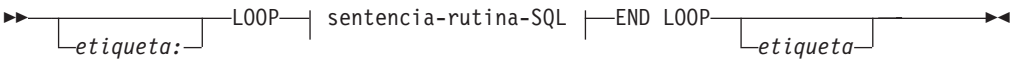
Invocación:

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

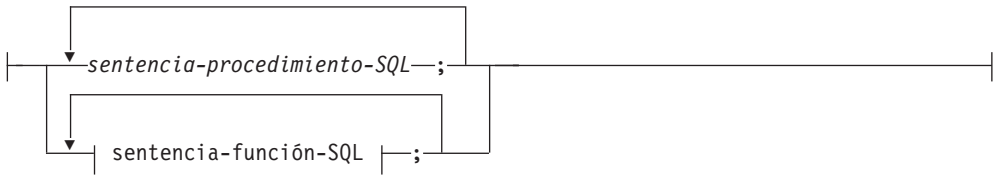
Autorización:

No se requieren privilegios para invocar esta sentencia de SQL. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia LOOP.

Sintaxis:



sentencia-rutina-SQL:



Descripción:

etiqueta
Especifica la etiqueta de la sentencia LOOP. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica la etiqueta final, se debe especificar la etiqueta inicial correspondiente.

sentencia-procedimiento-SQL
Especifica las sentencias de SQL que se deben invocar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

sentencia-función-SQL
Especifica las sentencias de SQL que se deben invocar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL. Consulte sentencia-función-SQL en la descripción de la sentencia FOR.

Ejemplos:

Este procedimiento utiliza una sentencia LOOP para leer valores de la tabla `employee`. Cada vez que se repite el bucle, el parámetro OUT *counter* se incrementa y el valor de *v_midinit* se comprueba para asegurarse de que el valor no es un espacio (' '). Si *v_midinit* es un espacio, la sentencia LEAVE pasa el flujo de control fuera del bucle.

```
CREATE PROCEDURE LOOP_UNTIL_SPACE(OUT counter INTEGER)
  LANGUAGE SQL
  BEGIN
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE v_firstnme VARCHAR(12);
    DECLARE v_midinit CHAR(1);
    DECLARE v_lastname VARCHAR(15);
    DECLARE c1 CURSOR FOR
      SELECT firstnme, midinit, lastname
        FROM employee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET counter = -1;
    OPEN c1;
    fetch_loop:
    LOOP
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      IF v_midinit = ' ' THEN
        LEAVE fetch_loop;
      END IF;
      SET v_counter = v_counter + 1;
    END LOOP fetch_loop;
    SET counter = v_counter;
    CLOSE c1;
  END
```

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139

MERGE

La sentencia MERGE actualiza un destino (una tabla o vista o las tablas o vistas subyacentes de una selección completa) utilizando datos de una fuente (resultado de una referencia de tabla). Las filas del destino que coinciden con la fuente se pueden suprimir o actualizar, según se especifique, y las filas que no existen en el destino se pueden insertar. Si se actualiza, suprime o inserta una fila en una vista, se actualiza, suprime o inserta la fila en las tablas en las que se basa la vista.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Si se especifica una operación de inserción, privilegio INSERT sobre la tabla o vista; si se especifica una operación de suprimir, privilegio DELETE sobre la tabla o vista; y si se especifica una operación de actualizar:
  - Privilegio UPDATE sobre la tabla o vista
  - Privilegio UPDATE sobre cada columna que se va a actualizar
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

El ID de autorización de la sentencia también debe tener al menos uno de los privilegios siguientes:

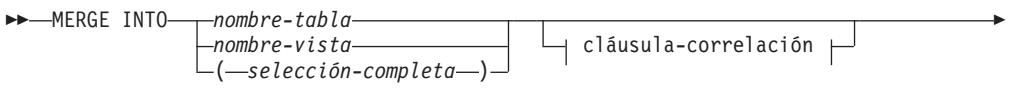
- Privilegio SELECT en cada tabla o vista identificada en la *referencia-tabla*
- Privilegio CONTROL en las tablas o vistas identificadas en la *referencia-tabla*
- Autorización SYSADM o DBADM

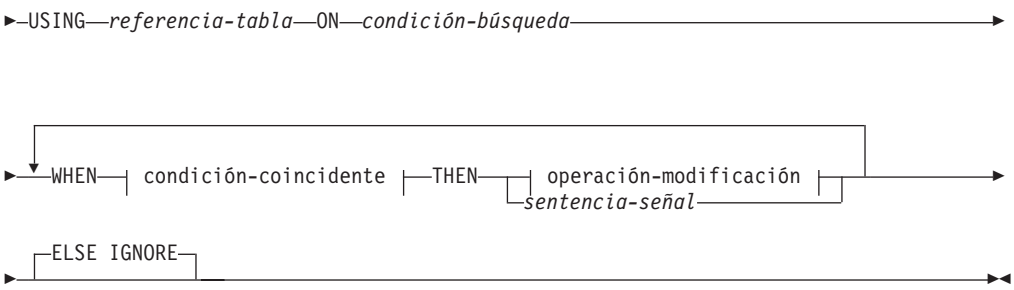
Si *condición-búsqueda*, *operación-inserción* o *cláusula-asignación* incluye una subconsulta, el ID de autorización de la sentencia también debe tener al menos uno de los privilegios siguientes:

- Privilegio SELECT en cada tabla o vista identificada en la subconsulta
- Privilegio CONTROL en las tablas o vistas identificadas en la subconsulta
- Autorización SYSADM o DBADM

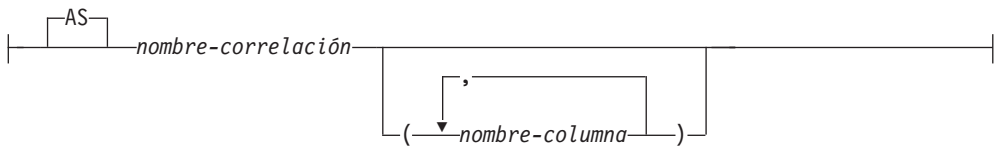
Si se especifica una expresión que se refiere a un función, el conjunto de privilegios debe incluir la autoridad que sea necesaria para ejecutar la función.

Sintaxis:

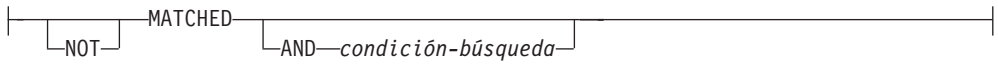




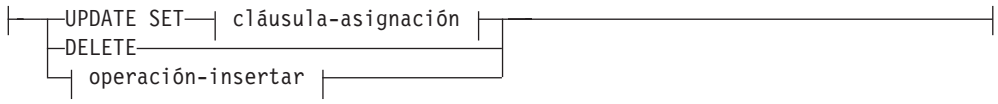
cláusula-correlación:



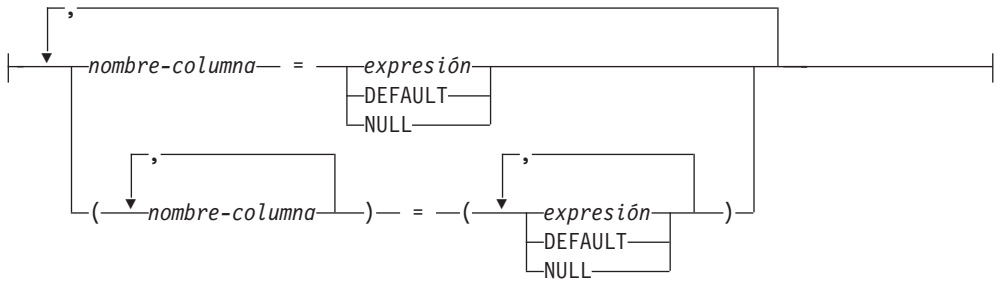
condición-coincidente:



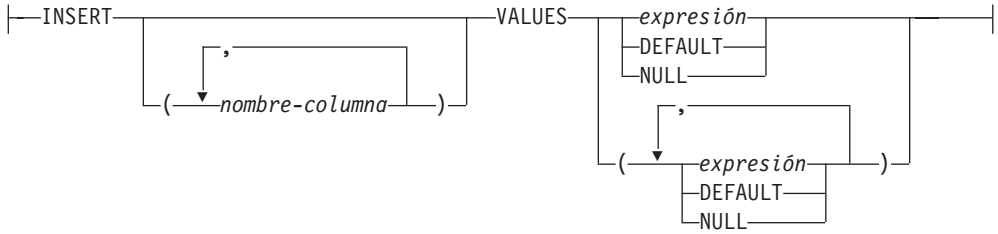
operación-modificación:



cláusula-asignación:



operación-insertar:



Descripción:

nombre-tabla, *nombre-vista* **o** *(selección-completa)*
Identifica el destino de las operaciones de actualización, supresión o inserción de la fusión. El nombre debe identificar una tabla o vista que exista en el servidor actual, pero no debe identificar una tabla de catálogo, una tabla de consultas materializadas mantenida por el sistema, una vista de una tabla de catálogos, una vista de sólo lectura, o una vista que directa o indirectamente contenga una cláusula WHERE que haga referencia a una subconsulta o una rutina definida como NOT DETERMINISTIC o EXTERNAL ACTION (SQLSTATE 42807).

Si el destino de la operación de fusión es una selección completa, la selección completa debe ser actualizable, suprimible o insertable, según lo definido en los elementos de Notas "Vistas actualizables", "Vistas suprimibles" o "Vistas insertables" de la descripción de la sentencia CREATE VIEW.

cláusula-correlación
Se puede utilizar en *condición-búsqueda* o a la derecha de una *cláusula-asignación* para designar una tabla, vista o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte "referencia-tabla" en la descripción de "Subselección".

USING *referencia-tabla*
Especifica un conjunto de filas como una tabla de resultados para fusionarse en el destino. Si la tabla de resultados está vacía, se devuelve un aviso (SQLSTATE 02000).

ON *condición-búsqueda*
Especifica las filas de *referencia-tabla* que se utilizarán en la operación de actualización y supresión de la fusión, y las filas que se utilizarán en la operación de inserción de la fusión.

Cada *nombre-columna* en la condición de búsqueda, que no sea en una subconsulta, debe nombrar una columna de la tabla, vista, o *referencia-tabla* de destino. Cuando la condición de búsqueda incluye una subconsulta en la que la misma tabla es el objeto base de la sentencia MERGE y de la subconsulta, se evalúa completamente la subconsulta antes de que se actualice o inserte ninguna fila.

condición-búsqueda se aplica a cada fila de la tabla de destino y tabla de resultados de *referencia-tabla*. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* sea verdadero, se realiza la operación de actualización o supresión especificada. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* no sea verdadero, se realiza la operación de inserción especificada.

Si la *condición-búsqueda* contiene una subconsulta, la subconsulta puede considerarse ejecutada cada vez que la condición de búsqueda se aplica a una fila de la tabla de resultados de la *referencia-tabla*, y los resultados utilizados en la aplicación de la condición de búsqueda. De hecho, una subconsulta sin referencias correlacionadas se ejecuta una sola vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una o más veces para cada fila de la tabla de resultados de la *referencia-tabla*.

WHEN *condición-coincidente*
Especifica la condición bajo la que se ejecuta la *operación-modificación* o la *sentencia-señal*. Cada *condición-coincidente* se evalúa en el orden de

especificación. Las filas para las que la *condición-coincidente* se evalúa en verdadera no se consideran en las condiciones coincidentes subsiguientes.

MATCHED
Indica la operación a realizar en las filas donde la condición de búsqueda ON es verdadera. Sólo UPDATE, DELETE o *sentencia-señal* se pueden especificar después de THEN.

AND *condición-búsqueda*
Especifica otra condición de búsqueda que se debe aplicar a las filas que han coincidido con la condición de búsqueda ON para la operación a realizar después de THEN.

NOT MATCHED
Indica la operación a realizar en las filas donde la condición de búsqueda ON es falsa o desconocida. Sólo se pueden especificar INSERT o *sentencia-señal* después de THEN.

AND *condición-búsqueda*
Especifica otra condición de búsqueda que se debe aplicar a las filas que no han coincidido con la condición de búsqueda ON para la operación que se debe realizar después de THEN.

THEN *operación-modificación*
Especifica la operación que se debe ejecutar cuando la *condición-coincidente* se evalúa como verdadera.

UPDATE SET
Especifica la operación de actualización que se debe ejecutar para las filas donde la *condición-coincidente* se evalúa en verdadera.

cláusula-asignación
Especifica una lista de actualizaciones de columna.

nombre-columna
Identifica una columna que se debe actualizar. El *nombre-columna* debe identificar una columna de la tabla o vista especificada, pero no una columna de vista derivada de una función escalar, constante o expresión. No se debe especificar una columna más de una vez (SQLSTATE 42701).

Una columna de vista derivada de la misma columna como otra columna de la vista se puede actualizar, pero no se pueden actualizar ambas columnas en la misma sentencia MERGE (SQLSTATE 42701).

expresión
Indica el nuevo valor de la columna. La *expresión* no debe incluir ninguna función de columna (SQLSTATE 42903).

Una *expresión* puede contener referencias a columnas del *nombre-tabla* o *nombre-vista*. Para cada fila que se actualiza, el valor de dicha columna en una expresión es el valor de la columna en la fila antes de que se actualice la fila.

DEFAULT
El valor por omisión asignado a la columna. Se puede especificar DEFAULT sólo para columnas que tengan un valor por omisión. Para obtener información acerca de los valores por omisión de tipos de datos, consulte la descripción de la cláusula DEFAULT en la sentencia “CREATE TABLE”.

Se debe especificar DEFAULT para una columna que se ha definido como GENERATED ALWAYS. Se puede especificar un valor para una columna que se ha definido como GENERATED BY DEFAULT.

NULL
Especifica un valor nulo como el nuevo valor de la columna. Especifique NULL sólo para columnas que pueden contener nulos (SQLSTATE 23502).

DELETE
Especifica la operación de suprimir que se debe ejecutar para las filas donde la *condición-coincidente* se evalúa en verdadera.

operación-insertar
Especifica la operación de insertar que se debe ejecutar para las filas donde la *condición-coincidente* evalúa en verdadera.

INSERT
Presenta una lista de nombres de columna y expresiones de valores de fila que se deben utilizar para la operación de insertar.

El número de valores para la fila en la expresión de valor de fila debe ser igual al número de nombres en la lista de columnas a insertar. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna, etcétera.

(*nombre-columna,...***)**
Especifica las columnas para las que se proporcionan los valores de inserción. Cada nombre debe identificar una columna de la tabla o vista. No se debe identificar la misma columna más de una vez (SQLSTATE 42701). No se debe identificar una columna de vista que no puede aceptar valores de inserción. No se puede insertar un valor en una columna de vista que se derive de:

- Una constante, expresión o función escalar
- La misma columna de tabla base que otra columna de la vista.

Si el objeto de la operación es una vista con dichas columnas, se debe especificar una lista de nombres de columna y dicha lista no debe identificar estas columnas.

La omisión de la lista de columnas es una especificación implícita de una lista en la que cada columna de la tabla o vista se identifica en orden de izquierda a derecha. Esta lista se establece cuando se prepara la sentencia, y, por lo tanto, no incluye las columnas que se han añadido a la tabla después de preparar la sentencia.

VALUES
Introduce una o varias filas de valores que se deben insertar.

expresión
Cualquier expresión que no incluye un nombre de columna (SQLSTATE 42703).

DEFAULT
El valor por omisión asignado a la columna. Se puede especificar DEFAULT sólo para columnas que tengan un valor por omisión. Para obtener información acerca de los valores por omisión de tipos de datos, consulte la descripción de la cláusula DEFAULT en la sentencia “CREATE TABLE”.

Se debe especificar DEFAULT para una columna que se ha definido como GENERATED ALWAYS. Se puede especificar un valor para una columna que se ha definido como GENERATED BY DEFAULT.

NULL
Especifica el valor nulo como el valor de la columna. Especifique NULL sólo para columnas que pueden contener nulos (SQLSTATE 23502).

sentencia-señal
Especifica la sentencia SIGNAL que se ejecutará para devolver un error cuando la *condición-coincidente* se evalúa en verdadera.

ELSE IGNORE
Especifica que no se realizará ninguna acción para las filas donde ninguna *condición-coincidente* se evalúa en verdadera.

Normas:

- Se puede especificar más de una *operación-modificación* (UPDATE SET, DELETE o *operación-insertar*), o *sentencia-señal* en una única sentencia MERGE.
- Sólo se puede trabajar en cada fila del destino una vez. Sólo se puede identificar una fila del destino como MATCHED con una fila en la tabla de resultados de la *referencia-tabla* (SQLSTATE 21506). Una operación de SQL anidado (RI o desencadenante excepto el desencadenante INSTEAD OF) no puede especificar la tabla de destino (o una tabla en la misma jerarquía de tablas) como un destino de una sentencia UPDATE, DELETE, INSERT o MERGE (SQLSTATE 27000).

Para otras normas que afectan a la parte de la operación de actualizar, insertar o suprimir de la sentencia MERGE, consulte la sección ″Normas″ de la descripción de la sentencia correspondiente.

Notas:

- ***Orden de proceso***:
  1. Determine el conjunto de filas que se deben procesar de la fuente y destino. Si se utiliza CURRENT TIMESTAMP en esta sentencia, sólo se realiza una lectura de reloj para toda la sentencia.
  2. Utilice la cláusula ON para clasificar estas filas en MATCHED o NOT MATCHED.
  3. Evalúe cualquier *condición-coincidente* en las cláusulas WHEN.
  4. Evalúe cualquier *expresión* en cualquier *cláusula-asignación* y *operación-insertar*.
  5. Ejecute cada *sentencia-señal*.
  6. Aplique cada *operación-modificación* a las filas aplicables en el orden especificado. Las restricciones y desencadenantes activados por cada *operación-modificación* se ejecutan para la *operación-modificación*. Los desencadenantes a nivel de sentencia se activan incluso si ninguna fila satisface la *operación-modificación*. Cada *operación-modificación* puede afectar a los desencadenantes y restricciones referenciales de cada *operación-modificación* subsiguiente.
- ***Atomicidad a nivel de sentencia***: Si se produce un error durante la ejecución de la sentencia MERGE, se retrotrae toda la sentencia.
- ***Número de filas actualizadas***: Cuando se completa la ejecución de una sentencia MERGE, el valor del elemento ROW_COUNT para GET DIAGNOSTICS y SQLERRD(3) en SQLCA es el número de filas en las que actúa la sentencia MERGE, excluyendo las filas identificadas por la cláusula ELSE IGNORE. El

valor de SQLERRD(3) no incluye el número de filas sobre las que se ha actuado como resultado de restricciones o activadores. El valor de SQLERRD(5) incluye el número de estas filas.

- ***La fila insertada tampoco se puede actualizar***: No se hace ningún intento de actualizar una fila en el destino que aún no existía antes de ejecutarse la sentencia MERGE; es decir, no hay actualizaciones de filas que haya insertado la sentencia MERGE.
- ***Desencadenantes INSTEAD OF***: Si se especifica una vista como destino de la sentencia MERGE, no se debe definir ningún desencadenante INSTEAD OF para la vista, o bien se debe definir un desencadenante INSTEAD OF para cada una de las operaciones de actualizar, suprimir e insertar (SQLSTATE 428FZ).

Ejemplos:

Ejemplo 1: Para actividades cuya descripción ha cambiado, actualice la descripción en la tabla de archivo. Para actividades nuevas, inserte en la tabla de archivo. Tanto la tabla de archivo como la tabla de actividades tienen actividad como clave primaria.

```
MERGE INTO archive ar
USING (SELECT activity, description FROM activities) ac
ON (ar.activity = ac.activity)
WHEN MATCHED THEN
   UPDATE SET
      description = ac.description
WHEN NOT MATCHED THEN
   INSERT
      (activity, description)
      VALUES (ac.activity, ac.description)
```

Ejemplo 2: Utilizando la tabla de envíos, fusione las filas en la tabla de inventario, aumentando la cantidad por número de piezas en la tabla de envíos para filas que coincidan; si no inserte el nuevo *partno* en la tabla de inventario.

```
MERGE INTO inventario AS in
USING (SELECT partno, description, count FROM shipment
       WHERE shipment.partno IS NOT NULL) AS sh
ON (in.partno = sh.partno)
WHEN MATCHED THEN
   UPDATE SET
      description = sh.description,
      quantity = in.quantity + sh.count
WHEN NOT MATCHED THEN
   INSERT
      (partno, description, quantity)
      VALUES (sh.partno, sh.description, sh.count)
```

Ejemplo 3: Utilizando la tabla de transacciones, fusione las filas en una tabla de contabilidad, actualizando el saldo del conjunto de transacciones contra un ID de cuenta e insertando nuevas cuentas desde las transacciones consolidadas donde ya no existen.

```
MERGE INTO account AS a
USING (SELECT id, sum(amount) sum_amount FROM transaction
       GROUP BY id) AS t
ON a.id = t.id
WHEN MATCHED THEN
   UPDATE SET
      balance = a.balance + t.sum_amount
WHEN NOT MATCHED THEN
   INSERT
      (id, balance)
      VALUES (t.id, t.sum_amount)
```

Ejemplo 4: Utilizando la tabla transaction_log, fusione las filas en la tabla employee_file, actualizando el teléfono y la oficina con la fila transaction_log más reciente en función de la hora de la transacción, e insertando la fila new employee_file más reciente donde la fila no existe aún.

```
MERGE INTO employee_file AS e
USING (SELECT empid, phone, office
       FROM (SELECT empid, phone, office,
             ROW_NUMBER() OVER (PARTITION BY empid
             ORDER BY transaction_time DESC) rn
             FROM transaction_log) AS nt
             WHERE rn = 1) AS t
ON e.empid = t.empid
WHEN MATCHED THEN
   UPDATE SET
      (phone, office) =
      (t.phone, t.office)
WHEN NOT MATCHED THEN
   INSERT
      (empid, phone, office)
      VALUES (t.empid, t.phone, t.office)
```

Ejemplo 5: Utilizando los valores proporcionados dinámicamente para una fila de empleado, actualice la tabla maestra de empleado si los datos corresponden a un empleado existente, o inserte la fila si los datos son para un empleado nuevo. El ejemplo siguiente es un fragmento de código de un programa en C.

```
hv1 =
"MERGE INTO employee AS t
USING TABLE(VALUES(CAST (? AS CHAR(6)), CAST (? AS VARCHAR(12)),
                   CAST (? AS CHAR(1)), CAST (? AS VARCHAR(15)),
                   CAST (? AS SMALLINT), CAST (? AS INTEGER)))
                   s(empno, firstnme, midinit, lastname, edlevel, salary)
ON t.empno = s.empno
WHEN MATCHED THEN
   UPDATE SET
      salary = s.salary
WHEN NOT MATCHED THEN
   INSERT
      (empno, firstnme, midinit, lastname, edlevel, salary)
      VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
              s.salary)";
EXEC SQL PREPARE s1 FROM :hv1;
EXEC SQL EXECUTE s1 USING '000420', 'SERGE', 'K', 'FIELDING', 18, 39580;
```

Ejemplo 6: Actualice la lista de actividades organizada por el Grupo A en la tabla de archivos. Suprima todas las actividades caducadas y actualice la información de actividades (description y date) en la tabla de archivo si éstas han cambiado. Para nuevas actividades prevista, inserte en el archivo. Señalice un error si se desconoce la fecha de la actividad. Se debe especificar la fecha de las actividades en la tabla de archivo. Cada grupo tiene una tabla de actividades. Por ejemplo, activities_groupA contiene todas las actividades que organizan, y la tabla de archivo contiene todas las actividades previstas organizadas por grupos diferentes en una empresa. La tabla de archivo (grupo, actividad) como clave primaria, y la fecha no puede contener nulos. Todas las tablas de actividades tienen actividad como clave primaria. La columna last_modified en el archivo está definida con CURRENT TIMESTAMP como valor por omisión.

```
MERGE INTO archive ar
USING (SELECT activity, description, date, last_modified
       FROM activities_groupA) ac
ON (ar.activity = ac.activity) AND ar.group = 'A'
WHEN MATCHED AND ac.date IS NULL THEN
   SIGNAL SQLSTATE '70001'
      SET MESSAGE_TEXT =
```

```
        ac.activity CONCAT ' cannot be modified. Reason: Date is not known'
WHEN MATCHED AND ac.date < CURRENT DATE THEN
   DELETE
WHEN MATCHED AND ar.last_modified < ac.last_modified THEN
   UPDATE SET
      (description, date, last_modified) = (ac.description, ac.date, DEFAULT)
WHEN NOT MATCHED AND ac.date IS NULL THEN
   SIGNAL SQLSTATE '70002'
      SET MESSAGE_TEXT =
         ac.activity CONCAT ' no se puede insertar. Razón: No se conoce la fecha'
WHEN NOT MATCHED AND ac.date >= CURRENT DATE THEN
   INSERT
      (group, activity, description, date)
      VALUES ('A', ac.activity, ac.description, ac.date)
ELSE IGNORE
```

Información relacionada:

- "Expresiones" en la publicación *Consulta de SQL, Volumen 1*
- "Condiciones de búsqueda" en la publicación *Consulta de SQL, Volumen 1*
- "Subselección" en la publicación *Consulta de SQL, Volumen 1*
- "CREATE TABLE" en la página 341
- "DELETE" en la página 497
- "INSERT" en la página 603
- "UPDATE" en la página 761
- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*

OPEN

La sentencia OPEN abre un cursor para que pueda utilizarse para leer filas de la tabla de resultados.

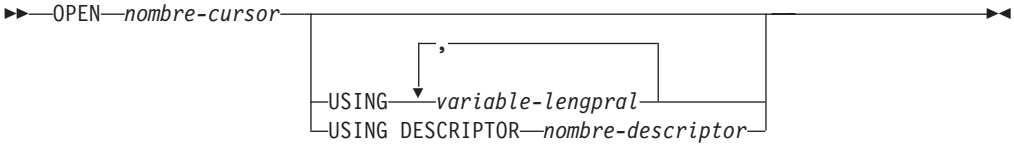
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

Sintaxis:



Descripción:

nombre-cursor
Identifica un cursor que se define en una sentencia DECLARE CURSOR que se ha expresado antes en el programa. Cuando se ejecuta la sentencia OPEN, el cursor debe estar en el estado cerrado.

La sentencia DECLARE CURSOR debe identificar una sentencia SELECT, de una de las siguientes maneras:

- Incluyendo la sentencia SELECT en la sentencia DECLARE CURSOR
- Incluyendo un *nombre-sentencia* que identifique una sentencia SELECT preparada.

La tabla de resultados del cursor se obtiene evaluándose la sentencia SELECT. La evaluación utiliza los valores actuales de cualquier registro especial o expresión PREVIOUS VALUE que se han especificado en la sentencia SELECT y los valores actuales de cualquier variable del lenguaje principal que se han especificado en la sentencia SELECT o en la cláusula USING de la sentencia OPEN. Las filas de la tabla de resultados pueden obtenerse durante la ejecución de la sentencia OPEN, y puede crearse una tabla temporal para contenerlas; o bien pueden obtenerse durante la ejecución de sentencias FETCH posteriores. En cualquier caso, el cursor se coloca en el estado abierto y se posiciona antes de la primera fila de su tabla de resultados. Si la tabla está vacía, el estado del cursor está, efectivamente, "después de la última fila".

USING
Introduce una lista de variables del lenguaje principal cuyos valores se sustituyen por los marcadores de parámetros (signos de interrogación) de una sentencia preparada. Si la sentencia DECLARE CURSOR nombra una sentencia preparada que incluye marcadores de parámetros, debe utilizarse USING. Si la sentencia preparada no incluye ningún marcador de parámetros, se ignora USING.

variable-lengpral
Identifica una variable descrita en el programa de acuerdo a las normas para la declaración de variables del lenguaje principal. El número de variables debe ser igual al número de marcadores de parámetros de las sentencia preparada. La variable *n* corresponde al marcador de parámetro *n* de la sentencia preparada. Cuando sea adecuado, pueden proporcionarse variables localizadoras y variables de referencia a archivos como fuente de valores para marcadores de parámetros.

DESCRIPTOR *nombre-descriptor*
Identifica una SQLDA que debe contener una descripción válida de las variables del lenguaje principal.

Antes de procesar la sentencia OPEN, el usuario debe establecer los campos siguientes en la SQLDA:

- SQLN para indicar el número de apariciones de SQLVAR proporcionadas en la SQLDA
- SQLDABC para indicar el número de bytes de almacenamiento asignados para la SQLDA
- SQLD para indicar el número de variables utilizadas en la SQLDA al procesar la sentencia
- Las apariciones de SQLVAR, para indicar los atributos de las variables.

La SQLDA debe tener suficiente almacenamiento para contener todas las ocurrencias de SQLVAR. Por lo tanto, el valor de SQLDABC debe ser mayor o igual que 16 + SQLN*(N), donde N es la longitud de una ocurrencia SQLVAR.

Si las columnas del resultado LOB necesitan acomodarse, debe haber dos entradas SQLVAR para cada elemento de la lista de selección (o columna de la tabla de resultados).

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

Normas:

- Cuando se evalúa la sentencia SELECT del cursor, cada marcador de parámetros de la sentencia se sustituye efectivamente por su variable del lenguaje principal correspondiente. Para un marcador de parámetros con tipo, los atributos de la variable de destino son aquellos especificados por la especificación CAST. Para un marcador de parámetros sin tipo, los atributos de la variable de destino se determinan de acuerdo al contexto del marcador de parámetros.
- Supongamos que V indique una variable del lenguaje principal que corresponda al marcador de parámetros P. El valor de P se asigna a la variable de destino para P de acuerdo con las normas de asignación de un valor a una columna. Por lo tanto:
  - V debe ser compatible con el destino.
  - Si V es una serie, su longitud (incluidos los blancos finales para series que no son series largas) no debe ser mayor que el atributo de longitud del destino.
  - Si V es un número, el valor absoluto de su parte correspondiente al entero no debe ser mayor que el valor absoluto máximo de la parte correspondiente a los enteros del destino.
  - Si los atributos de V no son idénticos a los atributos del destino, el valor se convierte para ajustarse a los atributos del destino.

Cuando se evalúa la sentencia SELECT del cursor, el valor utilizado en lugar de P es el valor de la variable de destino para P. Por ejemplo, si V es CHAR(6) y el destino es CHAR(8), el valor que se utiliza en lugar de P es el valor de V rellenado con dos blancos.

- La cláusula USING está pensada para una sentencia SELECT preparada que contiene marcadores de parámetros. Sin embargo, también puede utilizarse cuando la sentencia SELECT del cursor forma parte de la sentencia DECLARE CURSOR. En este caso la sentencia OPEN se ejecuta como si cada variable del lenguaje principal de la sentencia SELECT fuese un marcador de parámetros, excepto que los atributos de las variables de destino son iguales a los atributos de las variables del lenguaje principal de una sentencia SELECT. El efecto es alterar temporalmente los valores de las variables del lenguaje principal de la sentencia SELECT del cursor con los valores de las variables del lenguaje principal especificadas en la cláusula USING.
- Las sentencias y rutinas de cambio de datos de SQL que modifican datos SQL incorporados en la definición del cursor se ejecutan por completo y el conjunto de resultados se almacena en una tabla temporal cuando se abre el cursor. Si la ejecución de la sentencia resulta satisfactoria, el campo SQLERRD(3) contiene la suma del número de filas que están cualificadas para operaciones de inserción, actualización y supresión. Si se produce un error durante la ejecución de una sentencia OPEN en la que interviene un cursor que contiene una sentencia de cambio de datos dentro de una selección completa, los resultados de dicha sentencia de cambio de datos se retrotraen.

  La retrotracción explícita de una sentencia OPEN, o la retrotracción a un punto de guardan antes de una sentencia OPEN, cierra el cursor. Si la definición del cursor contiene una sentencia de cambio de datos dentro de la cláusula FROM de una selección completa, los resultados de la sentencia de cambio de datos se retrotraen.

  Los cambios en filas de una tabla que es el destino de una sentencia de cambio de datos anidada dentro de una sentencia SELECT o de una sentencia SELECT INTO se procesan cuando se abre el cursor y no se deshacen si se produce un error durante una operación de recuperación contra el cursor.

Notas:

- ***Estado cerrado de los cursores:*** Todos los cursores de un programa están en estado cerrado cuando se inicia el programa y cuando éste inicia la sentencia ROLLBACK.

  Todos los cursores, excepto los cursores abiertos declarados WITH HOLD, están en estado cerrado cuando el programa emite una sentencia COMMIT.

  Un cursor también puede estar en estado cerrado debido a que se ha ejecutado una sentencia CLOSE o se ha detectado un error que ha originado que la posición del cursor fuese imprevisible.
- Para recuperar filas de la tabla de resultados de un cursor, ejecute una sentencia FETCH cuando el cursor está abierto. La única manera de cambiar el estado de un cursor de cerrado a abierto es ejecutando la sentencia OPEN.
- ***Efecto de las tablas temporales:*** En algunos casos, la tabla de resultados de un cursor se obtiene durante la ejecución de las sentencias FETCH. En otros casos, se utiliza en su lugar el método de tabla temporal. Con este método toda la tabla de resultados se transfiere a una tabla temporal durante la ejecución de la sentencia OPEN. Cuando se utiliza una tabla temporal, los resultados de un programa pueden ser distintos, tal como se indica a continuación:
  - Puede producirse un error durante OPEN que, de lo contrario, no ocurriría hasta alguna sentencia FETCH posterior.

  - Las sentencias INSERT, UPDATE y DELETE que se ejecutan en la misma transacción mientras el cursor está abierto no pueden afectar a la tabla de resultados.
  - Las expresiones NEXT VALUE de la sentencia SELECT se evalúan para cada fila de la tabla de resultados durante la ejecución de OPEN.

  Y, a la inversa, si no se utiliza una tabla temporal, las sentencias INSERT, UPDATE y DELETE que se han ejecutado mientras el cursor estaba abierto pueden afectar a la tabla de resultados si se han emitido desde la misma unidad de trabajo, y las expresiones NEXT VALUE de la tabla de resultados de la sentencia SELECT se evalúan a medida que se busca cada fila. Esta tabla de resultados también puede verse afectada por las operaciones que ha ejecutado la misma unidad de trabajo y el efecto de tales operaciones no siempre es previsible. Por ejemplo, si el cursor C está posicionado en una fila de su tabla de resultados definida como SELECT * FROM T y se inserta una nueva fila en T, el efecto de dicha inserción en la tabla de resultados no es previsible ya que sus filas no están ordenadas. Por lo tanto, una sentencia FETCH C posterior puede recuperar la nueva fila de T o no.

- El poner en antememoria sentencias afecta a los cursores declarados abiertos por la sentencia OPEN.

Ejemplos:

Ejemplo 1: Escriba las sentencias incorporadas en un programa COBOL que:

1. Definan un cursor C1 que se va a utilizar para recuperar todas las filas de la tabla DEPARTMENT para los departamentos que administra el departamento (ADMRDEPT) 'A00'.
2. Coloque el cursor C1 antes de la primera fila que se debe leer.

```
EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT DEPTNO, DEPTNAME, MGRNO
                 FROM DEPARTMENT
                 WHERE ADMRDEPT = 'A00'
END-EXEC.


EXEC SQL  OPEN C1
END-EXEC.
```

Ejemplo 2: Codifique una sentencia OPEN para asociar un cursor DYN_CURSOR con una sentencia de selección definida dinámicamente en un programa C. Suponiendo que se utilizan dos marcadores de parámetros en el predicado de la sentencia de selección, se suministran dos referencias a variables del lenguaje principal con la sentencia OPEN para pasar valores integer y varchar(64) entre la aplicación y la base de datos. (Las definiciones de variables del lenguaje principal relacionadas, la sentencia PREPARE y la sentencia DECLARE CURSOR también se muestran en el ejemplo siguiente.)

```
EXEC SQL  BEGIN DECLARE SECTION;     static short    hv_int;
     char            hv_vchar64[65];
     char            stmt1_str[200];
   EXEC SQL  END DECLARE SECTION;

   EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
   EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

   EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

Ejemplo 3: Codifique una sentencia OPEN en el ejemplo 2, pero en este caso el número de tipos de datos de los marcadores de parámetros de la cláusula WHERE no se conocen.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char    stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

Información relacionada:

- "DECLARE CURSOR" en la página 483
- "EXECUTE" en la página 541
- "PREPARE" en la página 634
- "SQLDA (área de descriptores de SQL)" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dynamic.sqb -- How to update table data with cursor dynamically (MF COBOL)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "tut_read.sqc -- How to read tables (C)"
- "udfemsrv.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "tut_read.sqC -- How to read tables (C++)"
- "udfemsrv.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"

PREPARE

La sentencia PREPARE se utiliza por los programas de aplicación para preparar dinámicamente una sentencia de SQL para ejecución. La sentencia PREPARE crea una sentencia de SQL ejecutable, llamada una *sentencia preparada*, a partir de una forma de sentencia de serie de caracteres, denominada una *serie de sentencia*.

Invocación:

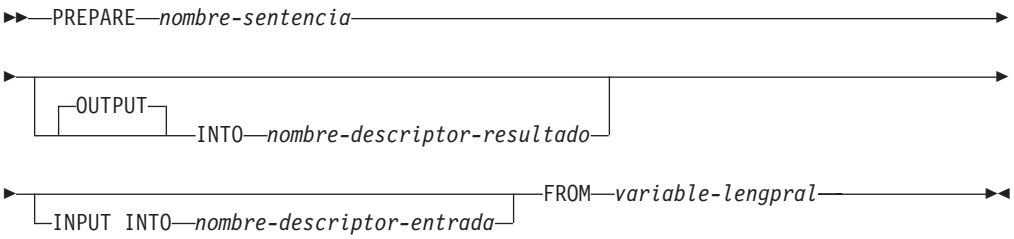
Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

Para las sentencias en las que el control de autorizaciones se realiza al preparar la sentencia (DML), los privilegios del ID de autorización de la sentencia deben incluir los necesarios para ejecutar la sentencia de SQL especificada por la sentencia PREPARE. El ID de autorización de la sentencia puede verse afectado por la opción de enlace DYNAMICRULES.

Para las sentencias en las que el control de autorizaciones se realiza durante la ejecución (sentencias DDL, GRANT y REVOKE), no es necesaria ninguna autorización para utilizar la sentencia; no obstante, se comprueba la autorización cuando se ejecuta la sentencia preparada.

Sintaxis:



Descripción:

nombre-sentencia
Identifica la sentencia preparada. Si el nombre identifica una sentencia preparada existente, se destruye dicha sentencia preparada previamente. El nombre no debe identificar ninguna sentencia preparada que sea la sentencia SELECT de un cursor abierto.

OUTPUT INTO
Si se utiliza OUTPUT INTO y la sentencia PREPARE se ejecuta satisfactoriamente, se colocará información acerca de los marcadores de parámetro de salida de la sentencia preparada en la SQLDA que *nombre-descriptor-resultado* especifica.

nombre-descriptor-resultado
Especifica el nombre de una SQLDA. (Como alternativa a esta cláusula, puede utilizarse la sentencia DESCRIBE.)

INPUT INTO
Si se utiliza INPUT INTO y la sentencia PREPARE se ejecuta satisfactoriamente, se colocará información acerca de los marcadores de parámetro de entrada de la sentencia preparada en la SQLDA que

nombres-descriptor-entrada especifica. Los marcadores de parámetro de entrada se consideran siempre anulables, independientemente de su uso.

nombre-descriptor-entrada
Especifica el nombre de una SQLDA. (Como alternativa a esta cláusula, puede utilizarse la sentencia DESCRIBE.)

FROM
Introduce la serie de la sentencia. La serie de la sentencia es el valor de la variable del lenguaje principal especificada.

variable-lengpral
Especifica una variable del lenguaje principal que se ha descrito en el programa de acuerdo con las normas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres de longitud fija o de longitud variable.

Normas:

- ***Normas para las sentencias:*** La sentencia debe ser una sentencia ejecutable que se pueda preparar dinámicamente. Debe ser una de las sentencias de SQL siguientes:
  - ALTER
  - CALL
  - COMMENT
  - COMMIT
  - CREATE
  - DECLARE GLOBAL TEMPORARY TABLE
  - DELETE
  - DROP
  - EXPLAIN
  - FLUSH EVENT MONITOR
  - FLUSH PACKAGE CACHE
  - GRANT
  - INSERT
  - LOCK TABLE
  - REFRESH TABLE
  - RELEASE SAVEPOINT
  - RENAME TABLE
  - RENAME TABLESPACE
  - REVOKE
  - ROLLBACK
  - SAVEPOINT
  - sentencia-select
  - SET CURRENT DEFAULT TRANSFORM GROUP
  - SET CURRENT DEGREE
  - SET CURRENT EXPLAIN MODE
  - SET CURRENT EXPLAIN SNAPSHOT
  - SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
  - SET CURRENT QUERY OPTIMIZATION
  - SET CURRENT REFRESH AGE

- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

- ***Marcadores de parámetro:*** Aunque la serie de caracteres de una sentencia no puede incluir referencias a las variables del lenguaje principal, sí puede incluir *marcadores de parámetro*. Estos pueden sustituirse por los valores de las variables del lenguaje principal cuando se ejecuta la sentencia preparada. En el caso de una sentencia CALL, también puede utilizarse un marcador de parámetro para los argumentos OUT e INOUT para el procedimiento almacenado. Tras haberse ejecutado CALL, el valor que se ha devuelto para el argumento se asignará a la variable del lenguaje principal que corresponde al marcador de parámetro.

  Un marcador de parámetros es un signo de interrogación (?) que se utiliza donde puede utilizarse una variable del lenguaje principal si la serie de caracteres de la sentencia es una sentencia de SQL estática. Para obtener información acerca de cómo se sustituyen los marcadores de parámetro por valores, consulte "OPEN" y "EXECUTE".

  Existen dos tipos de marcadores de parámetros:

  Marcador de parámetros con tipo
  Es un marcador de parámetros que se especifica junto con su tipo de datos de destino. Tiene el formato general:

  ```
  CAST(? AS tipo de datos)
  ```

  Esta notación no es una llamada a función, sino una "promesa" de que el tipo del parámetro en tiempo de ejecución será el tipo de datos especificado o algún otro tipo de datos que pueda convertirse al tipo de datos especificado. Por ejemplo, en:

  ```
  UPDATE EMPLOYEE
  SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
  WHERE EMPNO = ?
  ```

  el valor del argumento de la función TRANSLATE se proporcionará en tiempo de ejecución. El tipo de datos de dicho valor será VARCHAR(12) o algún tipo que pueda convertirse a VARCHAR(12).

  Marcador de parámetros sin tipo
  Es un marcador de parámetros que se especifica sin su tipo de datos de destino. Consta de un signo de interrogación individual. El tipo de datos de un marcador de parámetros sin tipo se proporciona por el contexto. Por ejemplo, el marcador de parámetros sin tipo del predicado de la sentencia de actualización anterior es el mismo que el tipo de datos de la columna EMPNO.

  Los marcadores de parámetros con tipo pueden utilizarse en sentencias de SQL dinámico donde esté soportada una variable del lenguaje principal y el tipo de datos se base en la promesa realizada en la función CAST.

  Los marcadores de parámetros sin tiempo pueden utilizarse en sentencias de SQL dinámico en las ubicaciones seleccionadas donde estén soportadas las variables del lenguaje principal. Estas ubicaciones y el tipo de datos resultante pueden encontrarse en la Tabla 11 en la página 637. Las ubicaciones se agrupan

en esta tabla por expresiones, predicados, funciones incorporadas y rutinas definidas por el usuario como ayuda para determinar si puede aplicarse o no un marcador de parámetro sin tipo. Cuando se utiliza un marcador de parámetros sin tipo en una función (incluyendo operadores aritméticos, operadores CONCAT y de indicación de fecha y hora) con un nombre de función no calificado, el calificador se establece en 'SYSIBM' con el propósito de la resolución de la función.

Tabla 11. Uso del marcador de parámetros sin tipo

Ubicación del marcador de parámetros sin tipo	**Tipo de datos**
Expresiones (incluidas la lista de selecciones, CASE y VALUES)	
Solo en una lista de selección	Error
Ambos operandos de un solo operador aritmético, después de considerar la prioridad de los operadores y el orden de las normas de la operación. Incluye casos como: `? + ? + 10`	Error
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 parámetros.)	DECIMAL(15,0)
Cualquier otro operando de una expresión de indicación de fecha y hora (por ejemplo 'timecol + ?' o '? - datecol').	Error
Ambos operandos de un operador CONCAT	Error
Un operando de un operador CONCAT en que el otro operando es un tipo de datos de caracteres que no es CLOB	Si un operando es CHAR(n) o VARCHAR(n), donde n es menor que 128, entonces el otro es VARCHAR(254 - n). En todos los demás casos el tipo de datos es VARCHAR(254).
Un operando de un operador CONCAT en que el otro operando es un tipo de datos gráfico que no es DBCLOB.	Si un operando es GRAPHIC(n) o VARGRAPHIC(n), donde n es menor que 64, entonces el otro es VARCHAR(127 - n). En todos los demás casos el tipo de datos es VARCHAR(127).
Un operando de un operador CONCAT en que el otro operando es una serie de gran objeto.	Igual que el del otro operando.

Tabla 11. Uso del marcador de parámetros sin tipo (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Como un valor a la derecha de una cláusula SET de una sentencia UPDATE.	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
La expresión que está a continuación de la palabra clave CASE en una expresión CASE simple	Error
Como mínimo una de las expresiones-resultado en una expresión CASE (tanto Simple como Con búsqueda) con el resto de las expresiones-resultado que sean marcadores de parámetros sin tipo o NULL.	Error
Cualquiera o todas las expresiones que siguen a WHEN en una expresión CASE simple.	El resultado de aplicar las normas para los tipos de datos de resultados a la expresión que sigue a CASE y a las expresiones que siguen a WHEN que no son marcadores de parámetro sin tipo.
Una expresión-resultado de una expresión CASE (tanto Simple como Con búsqueda) en la que por lo menos una expresión-resultado no es NULL y tampoco un marcador de parámetros sin tipo.	El resultado de aplicar las normas para los tipos de datos de resultados a todas las expresiones de resultados que no son NULL o marcadores de parámetro sin tipo.
Solo como una expresión-columna en una cláusula VALUES de una sola fila que no está en la sentencia INSERT.	Error
Solo como una expresión-columna en una cláusula VALUES de múltiples filas que no está dentro de una sentencia INSERT y para la que las expresiones-columna de la misma posición de todas las demás expresiones-fila son marcadores de parámetros sin tipo.	Error
Solo como una expresión-columna en una cláusula VALUES de múltiples filas que no está dentro de una sentencia INSERT y para la que la expresión de la misma posición de como mínimo otra expresión-fila no es un marcador de parámetros sin tipo ni NULL.	El resultado de aplicar las normas para los tipos de datos de resultados en todos los operandos que no son marcadores de parámetro sin tipo.
Solo como una expresión-columna en una cláusula VALUES de una sola fila dentro de una sentencia INSERT.	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.

Tabla 11. Uso del marcador de parámetros sin tipo (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Solo como una expresión-columna en una cláusula VALUES de múltiples filas dentro de una sentencia INSERT.	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo diferenciado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Como un valor a la derecha de una sentencia SET de registro especial	El tipo de datos del registro especial.
Predicados	
Ambos operandos de un operador de comparación	Error
Un operando de un operador de comparación en que el otro operando no es un marcador de parámetros sin tipo.	El tipo de datos del otro operando.
Todos los operandos del predicado BETWEEN	Error
Ó 1o y 2o ó 1o y 3o operandos de un predicado BETWEEN	Igual que el del único marcador sin parámetros.
Las restantes situaciones BETWEEN (es decir, sólo un marcador de parámetro sin tipo)	El resultado de aplicar las normas para los tipos de datos de resultados en todos los operandos que no son marcadores de parámetro sin tipo.
Todos los operandos de un predicado IN	Error
El primer operando de un predicado IN donde la parte derecha no es una subselección; por ejemplo, ? IN (?,A,B) o ? IN (A,?,B,?).	El resultado de aplicar las normas para los tipos de datos de resultados en todos los operandos de la lista IN (los operandos que se encuentran a la derecha de la palabra clave IN) que no son marcadores de parámetro sin tipo.
El primer operando de un predicado IN donde el lado derecho es una selección completa.	El tipo de datos de la columna seleccionada
Cualquiera o todos los operandos de la lista IN del predicado IN	Resultado de aplicar las normas sobre tipos de datos resultantes en todos los operandos del predicado IN (operandos a izquierda y derecha del predicado IN) que no sean marcadores de parámetros sin tipo.
Los tres operandos del predicado LIKE.	La expresión coincidente (operando 1) y la expresión patrón (operando 2) son VARCHAR(32672). La expresión de escape (operando 3) es VARCHAR(2).
La expresión coincidente del predicado LIKE cuando la expresión patrón o la expresión de escape no son un marcador de parámetros sin tipo.	VARCHAR(32672) o VARGRAPHIC(16336) en función del tipo de datos del primer operando que no es un marcador de parámetros sin tipo.

Tabla 11. Uso del marcador de parámetros sin tipo (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
La expresión patrón del predicado LIKE cuando la expresión coincidente o la expresión de escape no es un marcador de parámetros sin tipo.	VARCHAR(32672) o VARGRAPHIC(16336) en función del tipo de datos del primer operando que no es un marcador de parámetros sin tipo. Si el tipo de datos de la expresión coincidente es BLOB, se supone que el tipo de datos de la expresión patrón es BLOB(32672).
La expresión de escape del predicado LIKE cuando la expresión coincidente o la expresión patrón no es un marcador de parámetros sin tipo.	VARCHAR(2) o VARGRAPHIC(1) dependiendo del tipo de datos del primer operando que no es un marcador de parámetros sin tipo. Si el tipo de datos de la expresión coincidente o de la expresión patrón es BLOB, se supone el tipo de datos de la expresión de escape BLOB(1).
Operando del predicado NULL	error
Funciones incorporadas	
Todos los operandos de COALESCE (también llamados VALUE) o NULLIF	Error
Cualquier operando de COALESCE o NULLIF donde al menos el primer operando sea distinto de un marcador de parámetros sin tipo.	El resultado de aplicar las normas para los tipos de datos de resultados en todos los operandos que no son marcadores de parámetro sin tipo.
POSSTR (ambos operandos)	Ambos operandos son VARCHAR(32672).
POSSTR (un operando en que el otro operando es de un tipo de datos de caracteres).	VARCHAR(32672).
POSSTR (un operando en que el otro operando es de un tipo de datos gráficos).	VARGRAPHIC(16336).
POSSTR (el operando de serie-búsqueda cuando el otro operando es BLOB).	BLOB(32672).
SUBSTR (operando 1)	VARCHAR(32672)
SUBSTR (operandos 2 y 3)	INTEGER
El operando 1 de la función escalar TRANSLATE.	Error
Los operandos 2 y 3 de la función escalar TRANSLATE.	VARCHAR(32672) si el primer operando es un tipo de caracteres. VARGRAPHIC(16336) si el primer operando es un tipo gráfico.
El operando 4 de la función escalar TRANSLATE.	VARCHAR(1) si el primer operando es un tipo de caracteres. VARGRAPHIC(1) si el primer operando es un tipo gráfico.
El segundo operando de la función escalar TIMESTAMP.	TIME
Unary minus	DOUBLE-PRECISION
Unary plus	DOUBLE-PRECISION
El primer operando de la función VARCHAR_FORMAT.	TIMESTAMP
El primer operando de la función TIMESTAMP_FORMAT.	VARCHAR (longitud de una serie de caracteres corta)

Tabla 11. Uso del marcador de parámetros sin tipo (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Todos los demás operandos de todas las demás funciones escalares.	Error
El operando de una función de columna.	Error
Rutinas definidas por el usuario	
El argumento de una función	Error
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.

Notas:

- Cuando se ejecuta una sentencia PREPARE, la serie de sentencia se analiza y se comprueba si hay errores. Si la sentencia no es valida, la condición de error se notifica en la SQLCA. Cualquier sentencia EXECUTE u OPEN que hace referencia a esta sentencia también recibirá el mismo error (debido a una preparación implícita realizada por el sistema) a menos que se haya corregido el error.
- Se puede hacer referencia a sentencias preparadas en las siguientes clases de sentencias, con las restricciones que se indican:

En...	La sentencia preparada...
DESCRIBE	puede ser cualquier sentencia
DECLARE CURSOR	debe ser SELECT
EXECUTE	*no* debe ser SELECT

- Una sentencia preparada se puede ejecutar muchas veces. En efecto, si una sentencia preparada no se ejecuta más de una vez y no contiene marcadores de parámetros, es más eficaz utilizar la sentencia EXECUTE IMMEDIATE en lugar de las sentencias PREPARE y EXECUTE.
- La puesta en antememoria de la sentencia afecta a las preparaciones repetidas.

Ejemplos:

Ejemplo 1: Prepare y ejecute una sentencia que no es de selección en un programa COBOL. Suponga que la sentencia está contenida en una variable del lenguaje principal HOLDER y que el programa sustituirá una serie de sentencia de una variable del lenguaje principal basada en algunas instrucciones del usuario. La sentencia que se debe preparar no tiene ningún marcador de parámetros.

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER
  END-EXEC.
EXEC SQL  EXECUTE STMT_NAME
  END-EXEC.
```

Ejemplo 2: Prepare y ejecute una sentencia que no es de selección como en el ejemplo 1, excepto codificada para un programa C. Suponga también que la sentencia que se va a preparar puede contener cualquier cantidad de marcadores de parámetros.

```
EXEC SQL  PREPARE STMT_NAME FROM :holder;
EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

Suponga que se debe preparar la sentencia siguiente:

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

Las columnas de la tabla DEPT se definen de la siguiente manera:

```
DEPT_NO   CHAR(3) NOT NULL, -- número de departamento
DEPTNAME  VARCHAR(29), -- nombre del departamento
MGRNO     CHAR(6), -- número del director
ADMRDEPT  CHAR(3)  -- número del departamento de administración
```

Para insertar el número de departamento G01 que se denomina COMPLAINTS, que no tiene ningún director y que informa al departamento A00, la cláusula INSERT_DA debe tener los valores de la Tabla 12 antes de emitirse la sentencia EXECUTE.

Tabla 12.

Campo de SQLDA	**Valor**
SQLDAID	SQLDA
SQLDABC	192 (Véase la nota 1.)
SQLN	4
SQLD	4
SQLTYPE	452
SQLLEN	3
SQLDATA	*puntero para* G01
SQLIND	(Véase la nota 2.)
SQLNAME	
SQLTYPE	449
SQLLEN	29
SQLDATA	*puntero para* COMPLAINTS
SQLIND	*puntero para* 0
SQLNAME	
SQLTYPE	453
SQLLEN	6
SQLDATA	(Véase la nota 3.)
SQLIND	*puntero para* -1
SQLNAME	
SQLTYPE	453
SQLLEN	3
SQLDATA	*puntero para* A00
SQLIND	*puntero para* 0
SQLNAME	

Tabla 12. (continuación)

Campo de SQLDA	Valor

Notas:

1. Este valor es para una sentencia PREPARE que se ha realizado desde una aplicación de 32 bits. Si la sentencia PREPARE se hubiera realizado en una aplicación de 64 bits, SQLDABC tendría el valor 240.
2. El valor de SQLIND para esta SQLVAR se pasa por alto porque SQLTYPE identifica un tipo de datos sin posibilidad de nulos.
3. El valor de SQLDATA para esta SQLVAR se pasa por alto porque el valor de SQLIND indica que este es un valor NULL.

Información relacionada:

- "Identificadores" en la publicación *Consulta de SQL, Volumen 1*
- "Predicado LIKE" en la publicación *Consulta de SQL, Volumen 1*
- "DESCRIBE" en la página 504
- "EXECUTE" en la página 541
- "OPEN" en la página 629
- "Reglas para los tipos de datos del resultado" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "inpsrv.sqb -- A stored procedure using the GENERAL parameter style (MF COBOL)"
- "trigsql.sqb -- How to use a trigger on a table (MF COBOL)"
- "tbread.sqc -- How to read tables (C)"
- "tut_use.sqc -- How to modify a database (C)"
- "tbread.sqC -- How to read tables (C++)"
- "tut_use.sqC -- How to modify a database (C++)"

REFRESH TABLE

La sentencia REFRESH TABLE renueva los datos de una tabla de consultas materializadas.

Invocación:

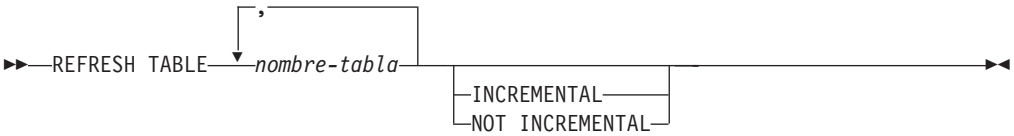
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para la tabla.

Sintaxis:



Descripción:

nombre-tabla
Identifica la tabla que se debe renovar.

El nombre, incluyendo el esquema implícito o explícito, debe identificar una tabla que ya exista en el servidor actual. La tabla debe permitir la sentencia REFRESH TABLE (SQLSTATE 42809). Ello incluye las tablas de consultas materializadas que se han definido con:

- REFRESH IMMEDIATE
- REFRESH DEFERRED

INCREMENTAL
Especifica una renovación incremental de la tabla donde sólo se considera la parte añadida (si existe) de sus tablas subyacentes o el contenido de una tabla de etapas asociada (si existe una y si su contenido es coherente). Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que la definición de la tabla de consultas materializadas debe volver a calcularse por completo), se devolverá un error (SQLSTATE 55019).

NOT INCREMENTAL
Especifica una renovación completa de la tabla en la que vuelve a realizarse el cálculo de la definición de la tabla de consultas materializadas.

Si no se ha especificado INCREMENTAL ni NOT INCREMENTAL, el sistema determinará si es posible realizar el proceso incremental; si no es posible, se realizará una renovación completa. Si existe una tabla de etapas para la tabla de consultas materializadas que va a renovarse, no será posible un proceso incremental porque la tabla de etapas está en estado pendiente, y se devolverá un error (SQLSTATE 428A8). La renovación completa se realizará si la tabla de etapas

o si la tabla de consultas materializadas está en un estado coherente; de lo contrario, para el proceso incremental se utilizará el contenido de la tabla de etapas.

Notas:

- Si se emite REFRESH TABLE en una tabla de consultas materializadas que hace referencia a uno o más apodos, el emisor deberá disponer de autorización para poder realizar selecciones desde las tablas remotas (SQLSTATE 42501).
- Cuando la sentencia se utiliza para renovar una tabla de consultas materializadas REFRESH IMMEDIATE cuyas tablas subyacentes se han cargado, puede que el sistema opte por renovar de manera incremental añadidas de sus tablas subyacentes. Cuando la sentencia se utiliza para renovar una tabla de consultas materializadas REFRESH DEFERRED con una tabla de etapas de soporte, puede que el sistema opte por renovar de manera incremental la tabla de consultas materializadas con las partes añadidas de sus tablas subyacentes que se han capturado en la tabla de etapas. Sin embargo, existen algunas situaciones en las que no es posible esta optimización y es necesario realizar una renovación completa (es decir, volver a calcular la definición de la tabla de consultas materializadas) para garantizar la integridad de los datos. El usuario puede solicitar explícitamente el mantenimiento incremental especificando la opción INCREMENTAL; si no es posible esta optimización, el sistema devuelve un error. Existen determinadas situaciones en las que el sistema hará uso del proceso incremental, dejando al usuario la responsabilidad de comprobar la coherencia.
- Si la tabla de consultas materializadas tiene una tabla de etapas asociada, la tabla de etapas se podará cuando la renovación se haya ejecutado satisfactoriamente.

Información relacionada:

- “SET INTEGRITY” en la página 725
- “CREATE USER MAPPING” en la página 464

RELEASE (Conexión)

La sentencia RELEASE (Connection) establece una o más conexiones en estado pendiente de liberación.

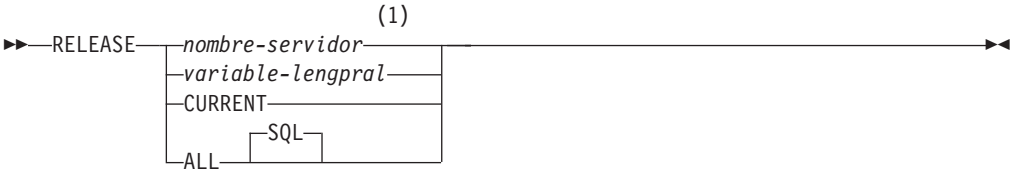
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna.

Sintaxis:



Notas:

1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal o un identificador delimitado.

Descripción:

nombre-servidor o *variable-lengpral*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lengpral* que contenga el *nombre-servidor*.

Si se especifica una *variable-lengpral*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* que está contenido en la *variable-lengpral* debe estar justificado por la izquierda y no debe estar delimitado por comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El alias-basedatos especificado o el alias-basedatos contenido en la variable del lenguaje principal debe identificar una conexión existente del proceso de aplicación. Si el alias-basedatos no identifica ninguna conexión existente, se genera un error (SQLSTATE 08003).

CURRENT
Identifica la conexión actual del proceso de aplicación. El proceso de aplicación debe estar en el estado conectado. Si no se genera un error (SQLSTATE 08003).

ALL o ALL SQL
Indica que todas las conexiones existentes del proceso de aplicación. Este formato de la sentencia RELEASE coloca todas las conexiones existentes del proceso de aplicación en el estado pendiente de liberación. Por lo tanto, todas

las conexiones se destruirán durante la siguiente operación de confirmación. No se produce ningún error ni aviso si no existen conexiones cuando se ejecuta la sentencia.

Ejemplos:

Ejemplo 1: La aplicación ya no necesita la conexión SQL con IBMSTHDB. La sentencia siguiente hará que se destruya durante la siguiente operación de confirmación:

```
EXEC SQL RELEASE IBMSTHDB;
```

Ejemplo 2: La aplicación ya no necesita la conexión actual. La sentencia siguiente hará que se destruya durante la siguiente operación de confirmación:

```
EXEC SQL RELEASE CURRENT;
```

Ejemplo 3: Si una aplicación no tiene necesidad de acceder a las bases de datos después de una confirmación pero va a continuar en ejecución durante un periodo de tiempo, es mejor no unir dichas conexiones innecesariamente. La sentencia siguiente puede ejecutarse antes de la confirmación para asegurar que todas las conexiones se destruyan en la confirmación:

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

La sentencia RELEASE SAVEPOINT sirve para indicar que la aplicación ya no desea mantener el punto de salvaguarda especificado. Después de invocar esta sentencia, ya no es posible hacer una retrotracción hasta el punto de salvaguarda.

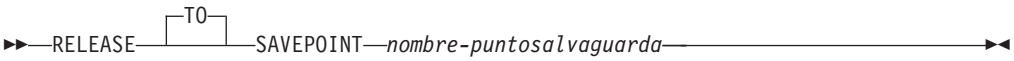
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:

nombre-puntosalvaguarda
Especifica el punto de salvaguarda que se debe liberar. Cualquier punto de salvaguarda anidado en el punto de salvaguarda nombrado también se liberará. La retrotracción a ese punto de salvaguarda, o a cualquier punto de salvaguarda anidado en él, ya no será posible. Si el punto de salvaguarda no existe en el nivel de punto de salvaguarda actual (consulte la sección "Normas" en la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B001). El *nombre-puntosalvaguarda* no puede empezar por 'SYS' (SQLSTATE 42939).

Notas:

- El nombre del punto de salvaguarda que se ha liberado se puede volver a utilizar ahora en otra sentencia SAVEPOINT, sin tener en cuenta si se ha especificado la palabra clave UNIQUE en una sentencia SAVEPOINT anterior especificando este mismo nombre de punto de salvaguarda.

Ejemplo:

Ejemplo 1: Liberación de un punto de salvaguarda llamado SAVEPOINT1.

```
RELEASE SAVEPOINT SAVEPOINT1
```

Información relacionada:

- "SAVEPOINT" en la página 687

RENAME

La sentencia RENAME cambia el nombre de una tabla o de un índice existente.

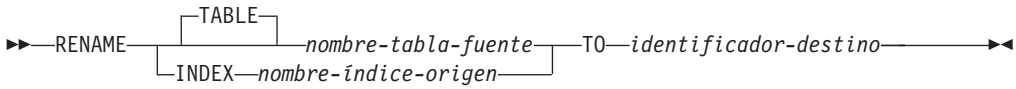
Invocación:

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

Autorización:

Los privilegios del ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM, el privilegio CONTROL para la tabla o índice o el privilegio ALTERIN para el esquema.

Sintaxis:



Descripción:

TABLE *nombre-tabla-fuente*
Identifica la tabla existente que se debe renombrar. El nombre, incluyendo el nombre de esquema, debe identificar una tabla que ya exista en la base de datos (SQLSTATE 42704). No debe ser el nombre de una tabla de catálogo (SQLSTATE 42832), una tabla de consultas materializadas, una tabla con tipo (SQLSTATE 42997), una tabla temporal global declarada (SQLSTATE 42995), un apodo o un objeto que no sea una tabla o un alias (SQLSTATE 42809). La palabra clave TABLE es opcional.

INDEX *nombre-índice-origen*
Especifica el nombre del índice existente cuyo nombre va a cambiarse. El nombre, incluido el nombre de esquema, debe identificar un índice que ya exista en la base de datos (SQLSTATE 42704). No debe ser el nombre de un índice de una tabla temporal global declarada (SQLSTATE 42995). El nombre de esquema no debe ser SYSIBM, SYSCAT, SYSFUN o SYSSTAT (SQLSTATE 42832).

identificador-destino
Especifica el nuevo nombre de la tabla o del índice sin un nombre de esquema. El nombre de esquema del objeto de fuente se utiliza para calificar el nuevo nombre del objeto. El nombre calificado *no* debe identificar una tabla, vista, alias o índice que ya exista en la base de datos (SQLSTATE 42710).

Normas:

Cuando se cambia el nombre de una tabla, la tabla fuente debe ajustarse a las normas siguientes:

- No debe existir una referencia a ésta en ninguna definición de vista o definición de tabla de consultas materializadas existente
- No estar referenciada en ninguna sentencia de SQL activada de activadores existentes ni ser la tabla sujeto de un activador existente

- No estar referenciada en una función SQL
- No tener ninguna restricción de comprobación
- No tener ninguna columna generada distinta de la columna de identidad
- No ser una tabla padre ni una tabla dependiente en ninguna restricción de integridad de referencia
- No ser el ámbito de ninguna columna de referencia existente.

Se devuelve un error (SQLSTATE 42986) si la tabla fuente viola una o más de esas condiciones.

Cuando se cambia el nombre de un índice:

- El índice de origen no debe ser un índice generado por el sistema para una tabla de implementación en la que se basa una tabla con tipo (SQLSTATE 42858).

Notas:

- Las entradas del catálogo se actualizan para reflejar el nuevo nombre de la tabla o índice.
- *Todas* las autorizaciones que se asocian a la tabla fuente o al nombre del índice se *transfieren* al nuevo nombre de tabla o índice (las tablas del catálogo de autorización se actualizan de acuerdo con ello).
- Los índices definidos en la tabla fuente se *transfieren* a la nueva tabla (las tablas de catálogo de índice se actualizan del modo apropiado).
- RENAME TABLE invalida cualquier paquete que dependa de la tabla fuente. RENAME INDEX invalida cualquier paquete que dependa del índice de origen.
- Si se utiliza un alias para el *nombre-tabla-fuente*, éste deberá resolverse de modo que dé como resultado un nombre de tabla. La tabla se renombra dentro del esquema de la tabla. El alias no se cambia por la sentencia RENAME y continúa haciendo referencia al anterior nombre de tabla.
- El nombre de una tabla con una clave primaria o con restricciones de unicidad podrá cambiarse si ninguna clave foránea hace referencia a la clave primaria o a las restricciones de unicidad.

Ejemplos:

Cambie el nombre de la tabla EMP por EMPLOYEE.

```
RENAME TABLE EMP TO EMPLOYEE   RENAME TABLE ABC.EMP TO EMPLOYEE
```

Cambie el nombre del índice NEW-IND por IND.

```
RENAME INDEX NEW-IND TO IND
RENAME INDEX ABC.NEW-IND TO IND
```

RENAME TABLESPACE

La sentencia RENAME TABLESPACE cambia el nombre de un espacio de tablas existente.

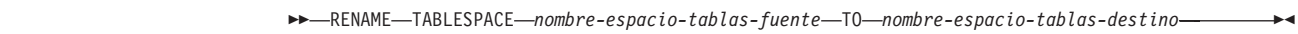
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

nombre-espacio-tablas-fuente
Especifica, en forma de nombre que consta de un solo elemento, el espacio de tablas existente que se debe renombrar. Se trata de un identificador de SQL (ordinario o delimitado). El nombre debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42704).

nombre-espacio-tablas-destino
Especifica el nuevo nombre del espacio de tablas, en forma de nombre formado por un solo elemento. Se trata de un identificador de SQL (ordinario o delimitado). El nombre *no* debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42710) y no puede comenzar con 'SYS' (SQLSTATE 42939).

Normas:

- El espacio de tablas SYSCATSPACE no se puede renombrar (SQLSTATE 42832).
- Los espacios de tablas cuyo estado sea "recuperación pendiente" o "recuperación en proceso" no se pueden renombrar (SQLSTATE 55039)

Notas:

- Cuando se renombra un espacio de tablas, se actualiza su tiempo mínimo de recuperación hasta el momento en que se cambió el nombre. Esto implica que una recuperación hecha a nivel de espacio de tablas debe hacerse como mínimo hasta alcanzar ese momento.
- El nuevo nombre del espacio de tablas se debe utilizar cuando se restaura un espacio de tablas a partir de una imagen de copia de seguridad, cuando el cambio de nombre se hizo después de crear la copia de seguridad.

Ejemplo:

Cambie el nombre del espacio de tablas USERSPACE1 a DATA2000:

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

REPEAT

La sentencia REPEAT ejecuta una sentencia o grupo de sentencias hasta que se cumpla una condición de búsqueda.

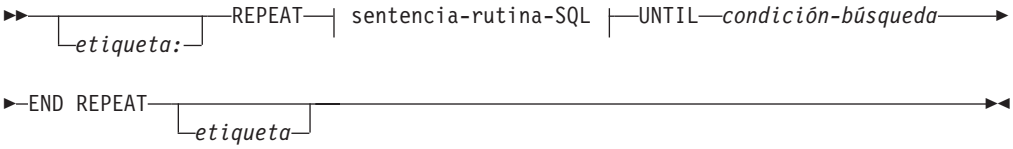
Invocación:

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

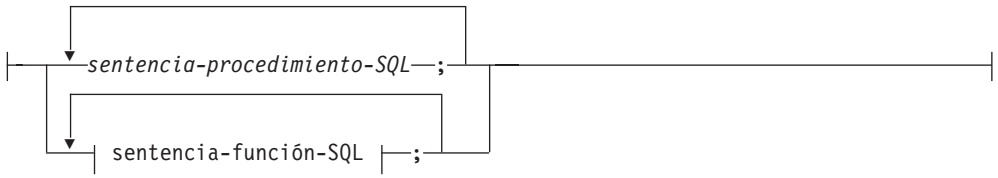
Autorización:

No se requieren privilegios para invocar una sentencia REPEAT. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia REPEAT.

Sintaxis:



sentencia-rutina-SQL:



Descripción:

etiqueta
Especifica la etiqueta de la sentencia REPEAT. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica una etiqueta final, se debe especificar también la etiqueta inicial correspondiente.

sentencia-procedimiento-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

sentencia-función-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL. Consulte sentencia-función-SQL en la descripción de la sentencia FOR.

condición-búsqueda
La *condición-búsqueda* se evalúa tras cada ejecución del bucle REPEAT. Si la condición es verdadera, se sale del bucle. Si la condición es desconocida o falsa, el bucle continúa.

Ejemplos:

Una sentencia REPEAT busca filas en una tabla hasta que se invoca el descriptor de contexto de condiciones *not_found*.

```
CREATE PROCEDURE REPEAT_STMT(OUT counter INTEGER)
  LANGUAGE SQL
  BEGIN
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE v_firstnme VARCHAR(12);
    DECLARE v_midinit CHAR(1);
    DECLARE v_lastname VARCHAR(15);
    DECLARE at_end SMALLINT DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE c1 CURSOR FOR
      SELECT firstnme, midinit, lastname
        FROM employee;
    DECLARE CONTINUE HANDLER FOR not_found
      SET at_end = 1;
    OPEN c1;
    fetch_loop:
    REPEAT
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      SET v_counter = v_counter + 1;
      UNTIL at_end > 0
    END REPEAT fetch_loop;
    SET counter = v_counter;
    CLOSE c1;
  END
```

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139

RESIGNAL

La sentencia RESIGNAL se utiliza para retransmitir una condición de error o de aviso. Da lugar a que se devuelva un error o un aviso especificándose el SQLSTATE, junto con el texto de mensaje opcional.

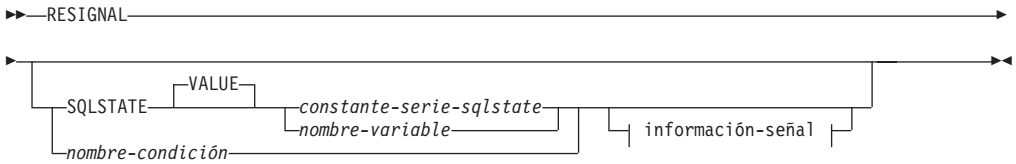
Invocación:

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

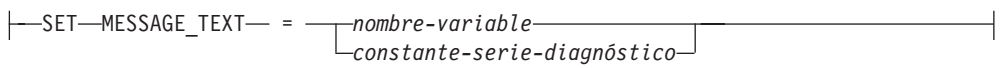
Autorización:

No se necesita.

Sintaxis:



información-señal:



Descripción:

SQLSTATE VALUE *constante-serie-sqlstate*
La constante especificada de tipo serie representa un estado SQL (SQLSTATE). Debe ser una constante de tipo serie de caracteres con 5 caracteres exactamente que siguen las normas aplicables a los SQLSTATE:

- Cada carácter debe pertenecer al conjunto de dígitos (del '0' al '9') o de letras mayúsculas no acentuadas (de la 'A' a la 'Z')
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

Si SQLSTATE no se ajusta a estas normas, se produce un error (SQLSTATE 428B3).

SQLSTATE VALUE *nombre-variable*
El nombre de variable especificado debe ser de tipo CHAR(5). Su valor en tiempo de ejecución de la sentencia debe cumplir las mismas normas que se describen para *constante-serie-sqlstate*. Si SQLSTATE no se ajusta a estas normas, se devuelve un error (SQLSTATE 428B3).

nombre-condición
Especifica el nombre de la condición.

SET MESSAGE_TEXT =
Especifica una serie de caracteres que describe el error o aviso. La serie de caracteres se devuelve en el campo sqlerrmc de la SQLCA. Si la serie tiene más de 70 bytes de longitud, se truncará sin avisar de ello.

nombre-variable
Identifica una variable SQL que se debe declarar dentro de la sentencia compuesta. La variable SQL debe estar definida con el tipo de datos CHAR o VARCHAR.

constante-serie-diagnóstico
Especifica una constante de tipo serie que contiene el texto del mensaje.

Notas:

- Si la sentencia RESIGNAL se especifica sin una cláusula SQLSTATE o un *nombre-condición*, se devuelve la misma condición que ha invocado el descriptor de contexto. El SQLSTATE, el SQLCODE y la SQLCA que se asocian a la condición no cambian.
- Si se emite una sentencia RESIGNAL y se ha especificado un SQLSTATE o un *nombre-condición*, el SQLCODE que se devuelve se basa en el valor de SQLSTATE, tal como se indica a continuación:
  - Si la clase de SQLSTATE especificada es '01' ó '02', se devuelve un aviso o una condición de no encontrado y el SQLCODE se establece en +438.
  - De otro modo, se devuelve una condición de excepción y el SQLCODE se establece en -438.

  Los demás campos de la SQLCA se establecen de la forma siguiente:
  - Los campos sqlerrd se establecen en cero.
  - Los campos sqlwarn se establecen en blancos.
  - El campo sqlerrmc se establece en los 70 primeros bytes de MESSAGE_TEXT.
  - El campo sqlerrml se establece en la longitud de sqlerrmc o bien en cero si no se ha especificado ninguna cláusula SET MESSAGE_TEXT.
  - El campo sqlerrp se establece en ROUTINE.
- Consulte el apartado ″Notas″ en ″Sentencia SIGNAL″ para obtener más información acerca de los valores de SQLSTATE.

Ejemplo:

Este ejemplo detecta los errores de división por cero. La sentencia IF utiliza una sentencia SIGNAL para invocar el gestor de condiciones *overflow*. El gestor de condiciones utiliza una sentencia RESIGNAL para devolver un valor de SQLSTATE diferente a la aplicación cliente.

```
CREATE PROCEDURE divide ( IN numerator INTEGER,
                          IN denominator INTEGER,
                          OUT result INTEGER)
LANGUAGE SQL
BEGIN
  DECLARE overflow CONDITION FOR SQLSTATE '22003';
  DECLARE CONTINUE HANDLER FOR overflow
    RESIGNAL SQLSTATE '22375';
  IF denominator = 0 THEN
    SIGNAL overflow;
  ELSE
    SET result = numerator / denominator;
  END IF;
END
```

Información relacionada:

- “SIGNAL” en la página 758

RETURN

La sentencia RETURN se utiliza para concluir la ejecución de una rutina. Para las funciones o métodos de SQL, devuelve el resultado de la función o método. Para un procedimiento SQL, devuelve opcionalmente un valor de estado de tipo entero.

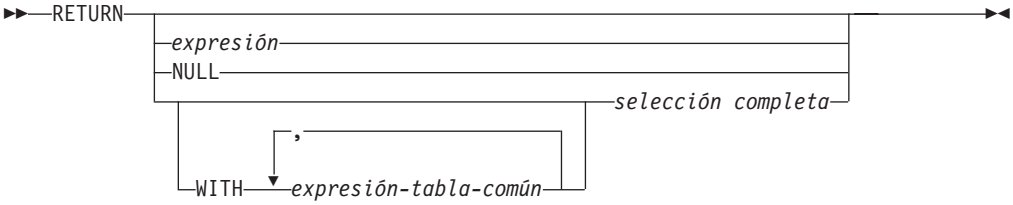
Invocación:

Esta sentencia se puede incluir en una función de SQL, un método de SQL o un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización:

No se requieren privilegios para invocar una sentencia RETURN. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar cualquier expresión o selección completa incluida en la sentencia RETURN.

Sintaxis:



Descripción:

expresión
Especifica un valor devuelto procedente de la rutina:

- Si la rutina es una función o un método, se deberá especificar *expresión*, NULL o *selección completa* (SQLSTATE 42630) y el tipo de datos del resultado debe ser asignable al tipo RETURNS de la rutina (SQLSTATE 42866).
- Si la rutina es una función de tabla, no puede especificarse una expresión escalar (que no sea una selección completa escalar) (SQLSTATE 428F1).
- Si la rutina es un procedimiento, el tipo de datos de *expresión* debe ser INTEGER (SQLSTATE 428E2). Un procedimiento no puede devolver NULL o una *selección completa*.

NULL
Especifica que la función o el método devuelve un valor nulo del tipo de datos definido en la cláusula RETURNS. No se puede especificar NULL para un RETURN de un procedimiento.

WITH *expresión-tabla-común*
Define una expresión de tabla común para utilizarla con la *selección completa* que viene a continuación.

selección completa
Especifica la fila o filas que se deben devolver para la función. El número de columnas de la *selección completa* debe coincidir con el número de columnas del resultado de la función (SQLSTATE 42811). Además, los tipos de columnas estáticas de la *selección completa* deben ser asignables a los tipos de columnas declaradas del resultado de la función, utilizando las normas para la asignación a columnas (SQLSTATE 42866).

No se puede especificar la *selección completa* para un RETURN de un procedimiento.

Si la rutina es una función escalar o un método, la *selección completa* debe devolver una columna (SQLSTATE 42823) y, como máximo, una fila (SQLSTATE 21000).

Si la rutina es una función de fila, debe devolver, cómo máximo, una fila (SQLSTATE 21505). Sin embargo, se pueden devolver una o más columnas.

Si la rutina es una función de tabla, puede devolver cero o más filas con una o más columnas.

Normas:

- La ejecución de un método o una función SQL debe finalizar con una sentencia RETURN (SQLSTATE 42632).
- En una función de fila o tabla SQL que utiliza una *sentencia-dinámica-compuesta*, la única sentencia RETURN permitida es la que se encuentra al final de la sentencia compuesta (SQLSTATE 429BD).

Notas:

- Cuando se devuelve un valor de un procedimiento, quien llama puede acceder al valor:
  - utilizando la sentencia GET DIAGNOSTICS para recuperar el RETURN_STATUS cuando el procedimiento de SQL se ha llamado desde otro procedimiento de SQL
  - utilizando el parámetro encontrado para el marcador de parámetro de valor devuelto en la sintaxis de la cláusula de escape CALL (?=CALL...) en una aplicación CLI
  - directamente desde el campo sqlerrd[0] de la SQLCA, tras procesarse la cláusula CALL de un procedimiento de SQL. Este campo sólo es válido si el SQLCODE es cero o un valor positivo (de lo contrario, debe darse por supuesto un valor de –1).

Ejemplos:

Utilice una sentencia RETURN para salir de un procedimiento almacenado SQL con un valor de estado de cero si la ejecución es satisfactoria y de –200 si no lo es.

```
BEGIN
...
   GOTO FAIL
...
   SUCCESS: RETURN 0
   FAIL: RETURN -200
        END
```

REVOKE (Autorizaciones de base de datos)

Esta forma de la sentencia REVOKE revoca las autorizaciones que se aplican a toda la base de datos.

Invocación:

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

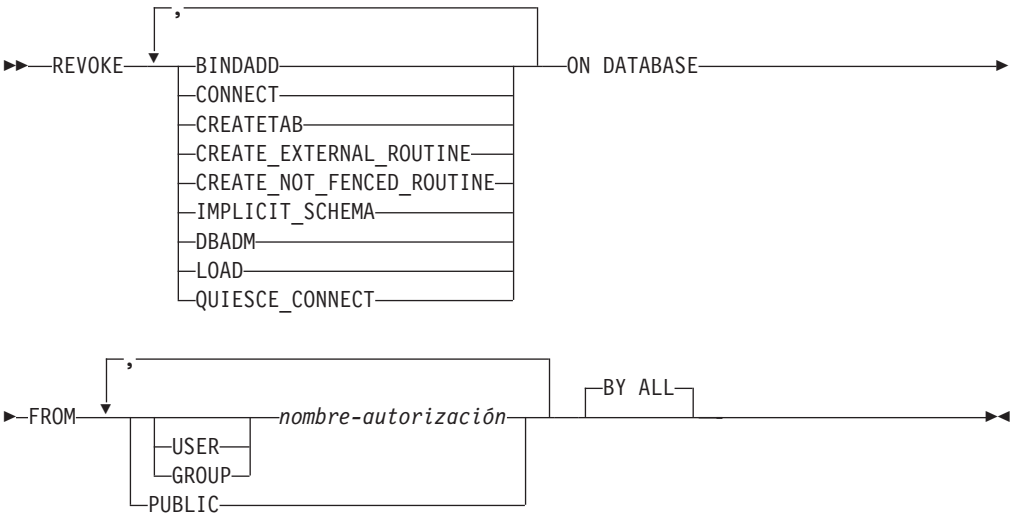
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización DBADM
- Autorización SYSADM

Para revocar la autorización DBADM, se necesita la autorización SYSADM.

Sintaxis:



Descripción:

BINDADD
Revoca la autorización para crear paquetes. El creador de un paquete tiene automáticamente el privilegio CONTROL en dicho paquete y conserva este privilegio incluso si se revoca posteriormente la autorización BINDADD.

La autorización BINDADD no puede revocarse desde un *nombre-autorización* que posea la autorización DBADM sin revocar también la autorización DBADM.

CONNECT
Revoca la autorización para acceder a la base de datos.

La revocación de la autorización CONNECT de un usuario no afecta a ningún privilegio que se haya otorgado a dicho usuario en objetos de la base de datos.

Si con posterioridad se vuelve a otorgar al usuario la autorización CONNECT, todos los privilegios que tenía anteriormente siguen siendo válidos (suponiendo que no se hayan revocado explícitamente).

La autorización CONNECT no puede revocarse de un *nombre-autorización* que contenga la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

CREATETAB
Revoca la autorización para crear tablas. El creador de una tabla tiene automáticamente el privilegio CONTROL en dicha tabla y conserva este privilegio incluso si se revoca la autorización CREATETAB con posterioridad.

La autorización CREATETAB no se puede revocar de un *nombre-autorización* que tiene la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

CREATE_EXTERNAL_ROUTINE
Revoca la autorización para registrar rutinas externas. Cuando una rutina externa se ha registrado, ésta sigue existiendo, aunque posteriormente se revoque la autorización CREATE_EXTERNAL_ROUTINE del ID de autorización que ha registrado la rutina.

La autorización CREATE_EXTERNAL_ROUTINE no puede revocarse de un *nombre-autorización* que tenga autorización DBADM o CREATE_NOT_FENCED_ROUTINE sin que también se revoque la autorización DBADM o CREATE_NOT_FENCED_ROUTINE (SQLSTATE 42504).

CREATE_NOT_FENCED_ROUTINE
Revoca la autorización para registrar rutinas que se ejecutan en el proceso del gestor de bases de datos. Cuando una rutina se ha registrado como no limitada, ésta sigue ejecutándose de esta forma, aunque posteriormente se revoque la autorización CREATE_NOT_FENCED_ROUTINE del ID de autorización que ha registrado la rutina.

La autorización CREATE_NOT_FENCED_ROUTINE no puede revocarse de un *nombre-autorización* que tenga autorización DBADM sin que también se revoque la autorización DBADM (SQLSTATE 42504).

IMPLICIT_SCHEMA
Revoca la autorización para crear implícitamente un esquema. No afecta la posibilidad de crear objetos en los esquemas existentes o de procesar una sentencia CREATE SCHEMA.

DBADM
Revoca la autorización DBADM.

La autorización DBADM no puede revocarse de PUBLIC (porque no puede otorgarse a PUBLIC).

PRECAUCIÓN:
La revocación de la autorización DBADM no revoca automáticamente ningún privilegio del *nombre-autorización* **para los objetos de la base de datos ni revoca ninguna de las demás autorizaciones de base de datos que se han otorgado implícita y automáticamente al otorgarse originalmente la autorización DBADM.**

LOAD
Revoca la autorización para cargar (LOAD) en esta base de datos.

QUIESCE_CONNECT
Revoca la autorización para acceder a la base de datos mientras está inactiva.

FROM
Indica a quién se revocan las autorizaciones.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No es posible revocar las autorizaciones de un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca las autorizaciones de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.DBAUTH tienen GRANTEETYPE de U, se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).

Notas:

- ***Compatibilidades***

  Para mantener la compatibilidad con las versiones anteriores a la Versión 8, la opción CREATE_NOT_FENCED puede sustituirse por CREATE_NOT_FENCED_ROUTINE.
- La revocación de un privilegio específico no revoca necesariamente la capacidad de realizar una acción. Un usuario puede seguir realizando una tarea si PUBLIC o un grupo tienen otros privilegios o si el usuario tiene un grado de autorización más alto como, por ejemplo, DBADM.

Ejemplos:

Ejemplo 1: Dado que USER6 sólo es un usuario y no un grupo, revoque el privilegio para crear las tablas del usuario USER6.

```
REVOKE CREATETAB ON DATABASE FROM USER6
```

Ejemplo 2: Revoque la autorización BINDADD en la base de datos de un grupo denominado D024. Existen dos filas en la vista de catálogo SYSCAT.DBAUTH para el usuario al que se otorga la autorización; una con un GRANTEETYPE que es U y otra con un GRANTEETYPE que es G.

```
REVOKE BINDADD ON DATABASE FROM GROUP D024
```

En este caso, debe especificarse la palabra clave GROUP; de lo contrario, se producirá un error (SQLSTATE 56092).

Información relacionada:

- "REVOKE (Privilegios de índice)" en la página 662
- "REVOKE (Privilegios de paquete)" en la página 664
- "REVOKE (Privilegios de esquema)" en la página 670
- "REVOKE (Privilegios de tabla, vista o apodo)" en la página 679
- "REVOKE (Privilegios de servidor)" en la página 675
- "REVOKE (Privilegios de espacio de tabla)" en la página 677
- "REVOKE (Privilegios de rutina)" en la página 667

Ejemplos relacionados:

- "dbauth.sqc -- How to grant, display, and revoke authorities at database level (C)"
- "dbauth.sqC -- How to grant, display, and revoke authorities at database level (C++)"
- "DbAuth.java -- Grant, display or revoke privileges on database (JDBC)"
- "DbAuth.sqlj -- Grant, display or revoke privileges on database (SQLj)"

REVOKE (Privilegios de índice)

Esta forma de la sentencia REVOKE revoca el privilegio CONTROL en un índice.

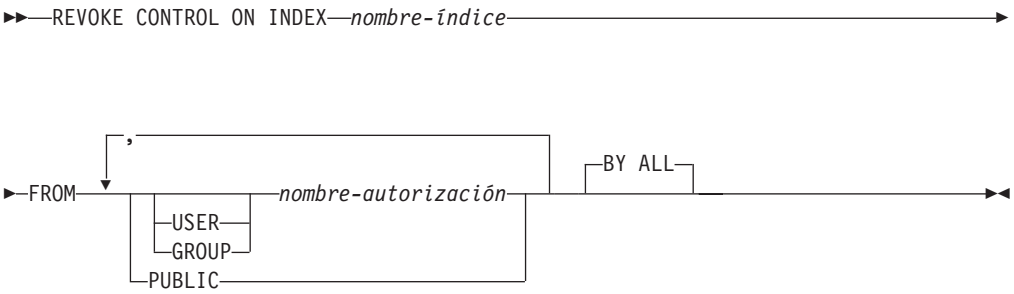
Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener autorización SYSADM o DBADM (SQLSTATE 42501).

Sintaxis:



Descripción:

CONTROL
Revoca el privilegio para eliminar el índice. Este es el privilegio CONTROL para índices, que se otorga automáticamente a los creadores de índices.

ON INDEX *nombre-índice*
Especifica el nombre del índice en el que se debe revocar el privilegio CONTROL.

FROM
Indica de quién se deben revocar los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.INDEXAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).

Notas:

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo tienen otros privilegios o si tienen autorizaciones como ALTERIN en el esquema de un índice.

Ejemplos:

Ejemplo 1: Dado que USER4 es sólo un usuario y no un grupo, revoque el privilegio para eliminar un índice DEPTIDX del usuario USER4.

```
REVOKE CONTROL ON INDEX DEPTIDX FROM USER4
```

Ejemplo 2: Revoque el privilegio para eliminar un índice LUNCHITEMS del usuario CHEF y del grupo WAITERS.

```
REVOKE CONTROL ON INDEX LUNCHITEMS
  FROM USER CHEF, GROUP WAITERS
```

Información relacionada:

- “REVOKE (Autorizaciones de base de datos)” en la página 658
- “REVOKE (Privilegios de paquete)” en la página 664
- “REVOKE (Privilegios de esquema)” en la página 670
- “REVOKE (Privilegios de tabla, vista o apodo)” en la página 679
- “REVOKE (Privilegios de servidor)” en la página 675
- “REVOKE (Privilegios de espacio de tabla)” en la página 677
- “REVOKE (Privilegios de rutina)” en la página 667

REVOKE (Privilegios de paquete)

Esta forma de la sentencia REVOKE revoca los privilegios CONTROL, BIND y EXECUTE en un paquete.

Invocación:

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

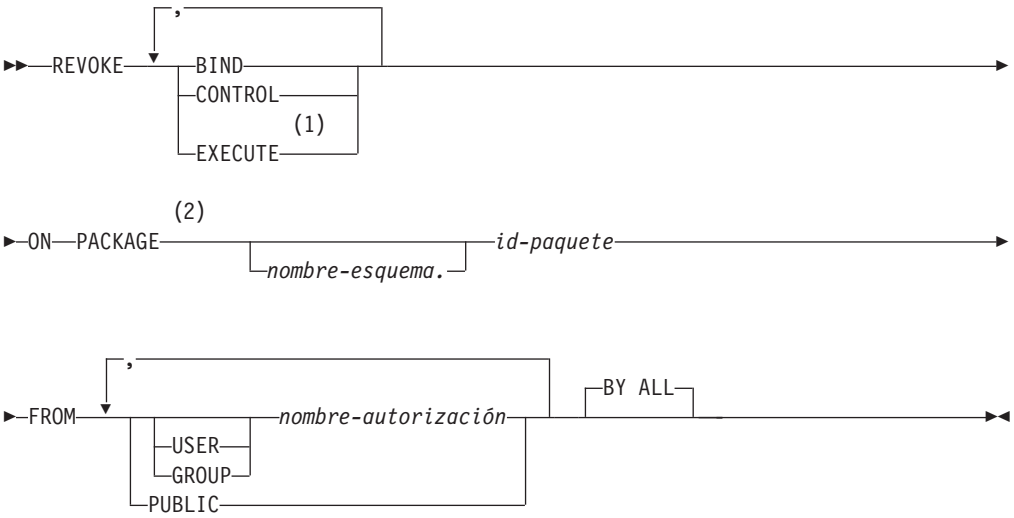
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL para el paquete referenciado
- Autorización SYSADM o DBADM.

Para revocar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Sintaxis:



Notas:

1 RUN se puede utilizar como sinónimo de EXECUTE.

2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

Descripción:

BIND
Revoca el privilegio para ejecutar BIND o REBIND para el paquete al que se hace referencia o bien para añadir una nueva versión del paquete al que se hace referencia.

El privilegio BIND no puede revocarse de un *nombre-autorización* que disponga de privilegio CONTROL para el paquete sin que se revoque también el privilegio CONTROL.

CONTROL
Revoca el privilegio para eliminar el paquete y para extender los privilegios de paquete a otros usuarios.

La revocación de CONTROL no revoca los demás privilegios del paquete.

EXECUTE
Revoca el privilegio para ejecutar el paquete.

El privilegio EXECUTE no puede revocarse de un *nombre-autorización* que tiene el privilegio CONTROL en el paquete sin revocar también el privilegio CONTROL.

ON PACKAGE *nombre-esquema.id-paquete*
Especifica el nombre del paquete para el que van a revocarse privilegios. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. La revocación de un privilegio de paquete se aplica a todas las versiones del paquete.

FROM
Indica de quién se deben revocar los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario de la operación de otorgar en la vista de catálogo SYSCAT.PACKAGEAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).

Notas:

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si

PUBLIC o un grupo poseen otros privilegios, o si tienen privilegios como, por ejemplo, ALTERIN en el esquema de un paquete.

Ejemplos:

Ejemplo 1: Revoque el privilegio EXECUTE en el paquete CORPDATA.PKGA de PUBLIC.

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

Ejemplo 2: Revoque la autorización CONTROL en el paquete RRSP_PKG para el usuario FRANK y para PUBLIC.

```
REVOKE CONTROL
  ON PACKAGE RRSP_PKG
  FROM USER FRANK, PUBLIC
```

Información relacionada:

- "REVOKE (Autorizaciones de base de datos)" en la página 658
- "REVOKE (Privilegios de índice)" en la página 662
- "REVOKE (Privilegios de esquema)" en la página 670
- "REVOKE (Privilegios de tabla, vista o apodo)" en la página 679
- "REVOKE (Privilegios de servidor)" en la página 675
- "REVOKE (Privilegios de espacio de tabla)" en la página 677
- "REVOKE (Privilegios de rutina)" en la página 667

REVOKE (Privilegios de rutina)

Esta forma de la sentencia REVOKE revoca privilegios para una rutina (función, método o procedimiento).

Invocación:

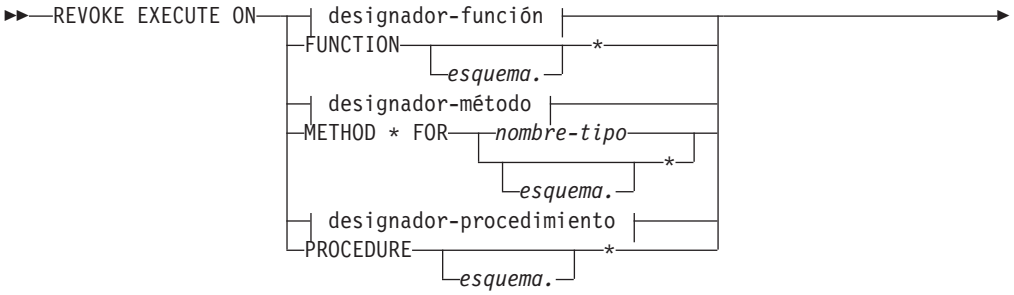
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

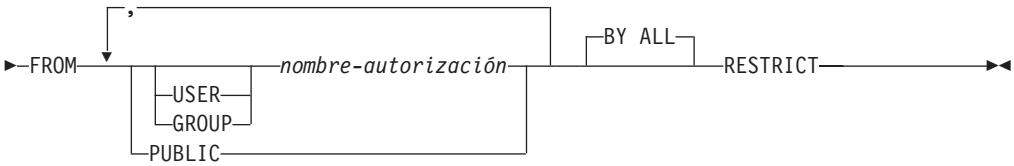
Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM

Sintaxis:





Descripción:

EXECUTE
Revoca el privilegio para ejecutar la función, el método o el procedimiento definido por el usuario que se identifica.

designador-función
Identifica la función de forma exclusiva.

FUNCTION *esquema.**
Identifica el otorgamiento explícito para todas las funciones existentes y futuras del esquema. La revocación del privilegio *esquema*.* no revoca ninguno de los privilegios que se han otorgado para una función específica. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

designador-método
Identifica un método de forma exclusiva.

METHOD*
Identifica el otorgamiento explícito para todos los métodos existentes y futuros para el tipo *nombre-tipo*. La revocación del privilegio * no revoca ninguno de los privilegios que se han otorgado para un método específico.

FOR *nombre-tipo*
Especifica el nombre del tipo en el que se encuentra el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el valor del registro especial CURRENT SCHEMA se utiliza como calificador de un nombre de tipo no calificado. En las sentencias de SQL estático, la opción de precompilación/enlace QUALIFIER especifica implícitamente el calificador para los nombres de tipo no calificados. Para identificar el otorgamiento explícito para todos los métodos existentes y futuros para todos los tipos existentes y futuros del esquema, puede utilizarse un asterisco (*) en lugar del *nombre-tipo*. La revocación del privilegio mediante la utilización de un asterisco para el método y el *nombre-tipo* no revoca ninguno de los privilegios que se han otorgado para un método específico o para todos los métodos para un tipo específico.

designador-procedimiento
Identifica el procedimiento de forma exclusiva.

PROCEDURE *esquema.**
Identifica el otorgamiento explícito para todos los procedimientos existentes y futuros del esquema. La revocación del privilegio *esquema*.* no revoca ninguno de los privilegios que se han otorgado para un procedimiento específico. En las sentencias de SQL dinámico, si no se especifica un esquema, se utilizará el esquema del registro especial CURRENT SCHEMA. En las sentencias de SQL estático, si no se especifica un esquema, se utilizará el esquema de la opción de precompilación/enlace QUALIFIER.

FROM
Especifica a quién se le revoca el privilegio EXECUTE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca el privilegio EXECUTE de todos los usuarios.

BY ALL
Revoca el privilegio EXECUTE de todos los usuarios especificados a los que se ha otorgado explícitamente el privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

RESTRICT
Especifica que el privilegio EXECUTE no puede revocarse si son verdaderas las dos condiciones siguientes (SQLSTATE 42893):

- La rutina especificada se utiliza en una vista, activador, restricción, extensión de índice, función de SQL, método de SQL, grupo de transformación o bien se hace referencia a ésta como SOURCE de una función con origen.
- La pérdida del privilegio EXECUTE puede dar lugar a que el usuario que ha definido la vista, activador, restricción, extensión de índice, función de SQL, método de SQL, grupo de transformación o función con origen ya no pueda ejecutar la rutina especificada.

Normas:

- No es posible revocar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas del usuario al que se ha otorgado el privilegio y que aparecen en las vistas del catálogo SYSCAT.ROUTINEAUTH tienen un GRANTEETYPE que es U, entonces se da por supuesto USER.
  - Si todas las filas tienen un GRANTEETYPE que es G, entonces se da por supuesto GROUP.
  - Si algunas filas tienen U y otras filas tienen G, se genera un error (SQLSTATE 56092).

Ejemplos:

Ejemplo 1: Revoque el privilegio EXECUTE para la función CALC_SALARY del usuario JONES. Se da por supuesto que, en el esquema, sólo existe una función con el nombre de función CALC_SALARY.

```
REVOKE EXECUTE ON FUNCTION CALC_SALARY FROM JONES RESTRICT
```

Ejemplo 2: Revoque el privilegio EXECUTE para el procedimiento VACATION_ACCR de todos los usuarios del servidor actual.

```
REVOKE EXECUTE ON PROCEDURE VACATION_ACCR FROM PUBLIC RESTRICT
```

Ejemplo 3: Revoque el privilegio EXECUTE para la función NEW_DEPT_HIRES de HR (Recursos humanos). La función tiene dos parámetros de entrada de tipo INTEGER y CHAR(10) respectivamente. Se da por supuesto que el esquema tiene más de una función denominada NEW_DEPT_HIRES.

```
REVOKE EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
  FROM HR RESTRICT
```

Ejemplo 4: Revoque el privilegio EXECUTE para el método SET_SALARY para el tipo EMPLOYEE del usuario Jones.

```
REVOKE EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE FROM JONES RESTRICT
```

Información relacionada:

- "REVOKE (Autorizaciones de base de datos)" en la página 658
- "REVOKE (Privilegios de índice)" en la página 662
- "REVOKE (Privilegios de paquete)" en la página 664
- "REVOKE (Privilegios de esquema)" en la página 670
- "REVOKE (Privilegios de tabla, vista o apodo)" en la página 679
- "REVOKE (Privilegios de servidor)" en la página 675
- "REVOKE (Privilegios de espacio de tabla)" en la página 677
- "Elementos comunes de la sintaxis" en la página viii

REVOKE (Privilegios de esquema)

Esta forma de la sentencia REVOQUE revoca los privilegios en un esquema.

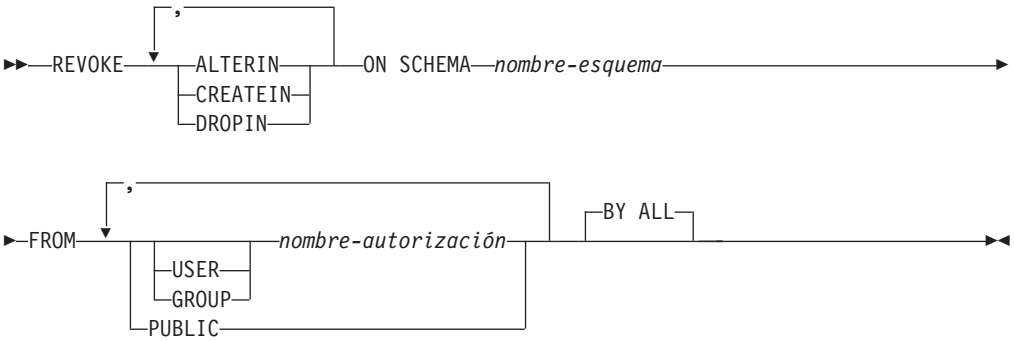
Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener autorización SYSADM o DBADM (SQLSTATE 42501).

Sintaxis:



Descripción:

ALTERIN
Revoca el privilegio para modificar o comentar los objetos del esquema.

CREATEIN
Revoca el privilegio para crear objetos en el esquema.

DROPIN
Revoca el privilegio para eliminar objetos en el esquema.

ON SCHEMA *nombre-esquema*
Especifica el nombre del esquema en el que se deben revocar los privilegios.

FROM
Indica de quién se deben revocar los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas del destinatario de la operación de otorgar en la vista de catálogo SYSCAT.SCHEMAAUTH tienen GRANTEETYPE de U, se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).

Notas:

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo tienen otros privilegios o si tienen una autorización de nivel superior como, por ejemplo, DBADM.

Ejemplos:

Ejemplo 1: Suponiendo que USER4 sea sólo un usuario y no un grupo, revoque el privilegio para crear objetos en el esquema DEPTIDX del usuario USER4.

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

Ejemplo 2: Revoque el privilegio de eliminar objetos en el esquema LUNCH del usuario CHEF y del grupo WAITERS.

```
REVOKE DROPIN ON SCHEMA LUNCH
  FROM USER CHEF, GROUP WAITERS
```

Información relacionada:

- “REVOKE (Autorizaciones de base de datos)” en la página 658
- “REVOKE (Privilegios de índice)” en la página 662
- “REVOKE (Privilegios de paquete)” en la página 664
- “REVOKE (Privilegios de tabla, vista o apodo)” en la página 679
- “REVOKE (Privilegios de servidor)” en la página 675
- “REVOKE (Privilegios de espacio de tabla)” en la página 677
- “REVOKE (Privilegios de rutina)” en la página 667

REVOKE (Privilegios de secuencia)

Esta forma de la sentencia REVOKE revoca los privilegios en una secuencia.

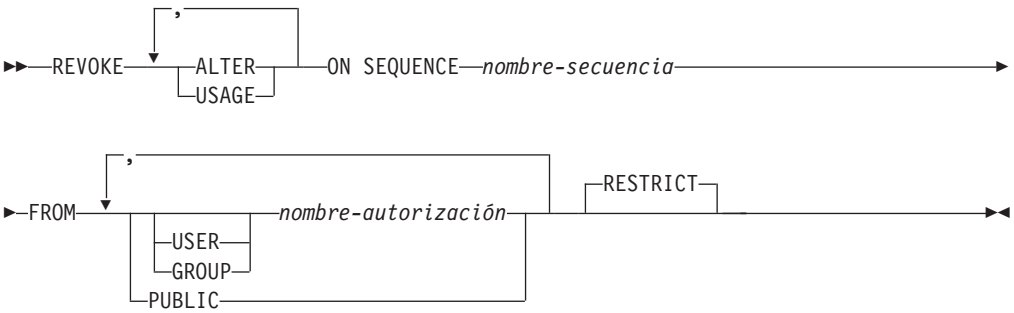
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica. No obstante, si se aplica la opción de vinculación DYNAMICRULES BIND, la sentencia no se puede preparar dinámicamente (SQLSTATE 42509).

Autorización:

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

Sintaxis:



Descripción:

ALTER
Revoca el privilegio de cambiar las propiedades de una secuencia o reiniciar la generación de números de secuencia mediante la sentencia ALTER SEQUENCE.

USAGE
Revoca el privilegio de poder hacer referencia a una secuencia mediante una *expresión-nextval* o *expresión-prevval*.

ON SEQUENCE *nombre-secuencia*
Identifica la secuencia para la que van a revocarse los privilegios especificados. El nombre de la secuencia, incluido un calificador de esquema implícito o explícito, debe identificar de forma exclusiva a una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704).

FROM
Especifica a quién se revocarán los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos. El ID de autorización de la propia sentencia REVOKE no se puede especificar (SQLSTATE 42502).

PUBLIC
Revoca los privilegios especificados a todos los usuarios.

RESTRICT
Esta palabra clave opcional indica que la sentencia dará error si se revoca algún objeto que dependa del privilegio.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas del usuario al que se ha otorgado el privilegio en la vista del catálogo SYSCAT.SEQUENCEAUTH tienen un GRANTEETYPE que es U, entonces se da por supuesto USER.
  - Si todas las filas tienen un GRANTEETYPE que es G, entonces se da por supuesto GROUP.
  - Si algunas filas tienen U y otras tienen G, se devuelve un error (SQLSTATE 56092).

Notas:

- La revocación de un privilegio específico no elimina necesariamente la capacidad de realizar una acción. Un usuario puede continuar si PUBLIC o un grupo al que pertenezca el usuario tienen otros privilegios, o si el usuario tiene un nivel de autorización más alto como, por ejemplo, DBADM.

Ejemplos:

Ejemplo 1: Revoque al usuario ENGLES el privilegio USAGE para una secuencia denominada GENERATE_ID. Hay una fila en la vista de catálogo SYSCAT.SEQUENCEAUTH para esta secuencia y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.

```
REVOKE USAGE ON SEQUENCE GENERATE_ID FROM ENGLES
```

Ejemplo 2: Revoque los privilegios de modificación en la secuencia GENERATE_ID que se han otorgado previamente a todos los usuarios locales. (Lo otorgado a usuarios específicos no se ve afectado.)

```
REVOKE ALTER ON SEQUENCE GENERATE_ID FROM PUBLIC
```

Ejemplo 3: Revoque todos los privilegios en la secuencia GENERATE_ID de los usuarios PELLOW y MLI y del grupo PLANNERS.

```
REVOKE ALTER, USAGE ON SEQUENCE GENERATE_ID
  FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

Información relacionada:

- “REVOKE (Autorizaciones de base de datos)” en la página 658
- “REVOKE (Privilegios de índice)” en la página 662
- “REVOKE (Privilegios de paquete)” en la página 664
- “REVOKE (Privilegios de esquema)” en la página 670
- “REVOKE (Privilegios de tabla, vista o apodo)” en la página 679
- “REVOKE (Privilegios de servidor)” en la página 675
- “REVOKE (Privilegios de espacio de tabla)” en la página 677

- "GRANT (Privilegios de secuencia)" en la página 586
- "REVOKE (Privilegios de rutina)" en la página 667

REVOKE (Privilegios de servidor)

Este formato de la sentencia REVOKE revoca el privilegio para acceder y utilizar una fuente de datos especificada en la modalidad de paso a través.

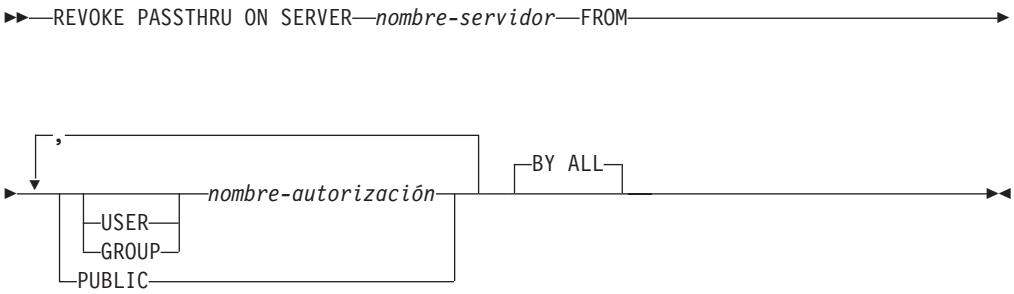
Invocación:

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

Autorización:

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

Sintaxis:



Descripción:

SERVER *nombre-servidor*
Nombra la fuente de datos para la cual se está revocando el privilegio para utilizarse en modalidad de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

FROM
Especifica a quién se revoca el privilegio.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca a todos los usuarios el privilegio de realizar un paso a través para *nombre-servidor*.

BY ALL
Revoca el privilegio de todos los usuarios especificados a los que se ha

otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

Ejemplos:

Ejemplo 1: Revoque el privilegio que USER6 tiene para utilizar la modalidad de paso a través para la fuente de datos MOUNTAIN.

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

Ejemplo 2: Revoque el privilegio que el grupo D024 tiene para usar la modalidad de paso a través para la fuente de datos EASTWING.

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

Los miembros del grupo D024 ya no podrán utilizar su ID de grupo para realizar un paso a través para EASTWING. Pero si cualquier miembro tiene el privilegio para usar el paso a través para EASTWING con su propio ID de usuario, conservará este privilegio.

Información relacionada:

- "REVOKE (Autorizaciones de base de datos)" en la página 658
- "REVOKE (Privilegios de índice)" en la página 662
- "REVOKE (Privilegios de paquete)" en la página 664
- "REVOKE (Privilegios de esquema)" en la página 670
- "REVOKE (Privilegios de tabla, vista o apodo)" en la página 679
- "REVOKE (Privilegios de espacio de tabla)" en la página 677
- "REVOKE (Privilegios de rutina)" en la página 667

REVOKE (Privilegios de espacio de tabla)

Esta forma de la sentencia REVOKE revoca el privilegio USE para una tabla.

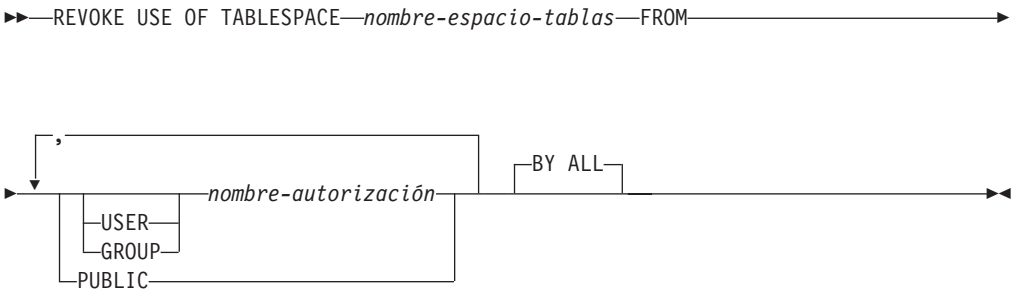
Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener autorización SYSADM, SYSCTRL o DBADM (SQLSTATE 42501).

Sintaxis:



Descripción:

USE
Revoca el privilegio para especificar, de forma explícita o por omisión, el espacio de tablas al crear una tabla.

OF TABLESPACE *nombre-espacio-tablas*
Identifica el espacio de tablas para el que debe revocar el privilegio USE. El espacio de tablas no puede ser SYSCATSPACE (SQLSTATE 42838) ni un espacio de tablas temporal del sistema (SQLSTATE 42809).

FROM
Especifica a quién se revoca el privilegio USE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización
Lista uno o varios ID de autorización.

El ID de autorización de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar los privilegios de un *nombre-autorización* que es igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca el privilegio USE a PUBLIC.

BY ALL
Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas de la vista de catálogo SYSCAT.TBSPACEAUTH correspondientes al usuario autorizado tienen U como GRANTEETYPE, se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se produce un error (SQLSTATE 56092).

Notas:

- La revocación del privilegio USE no revoca necesariamente la capacidad para crear tablas en ese espacio de tablas. Un usuario puede todavía crear tablas en ese espacio de tablas si el privilegio USE se otorga a PUBLIC o a un grupo, o si el usuario tiene una autorización de nivel superior, tal como DBADM.

Ejemplos:

Ejemplo 1: Revocación del privilegio del usuario BOBBY para crear tablas en el espacio de tablas PLANS.

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```

Información relacionada:

- "REVOKE (Autorizaciones de base de datos)" en la página 658
- "REVOKE (Privilegios de índice)" en la página 662
- "REVOKE (Privilegios de paquete)" en la página 664
- "REVOKE (Privilegios de esquema)" en la página 670
- "REVOKE (Privilegios de tabla, vista o apodo)" en la página 679
- "REVOKE (Privilegios de servidor)" en la página 675
- "REVOKE (Privilegios de rutina)" en la página 667

REVOKE (Privilegios de tabla, vista o apodo)

Esta forma de la sentencia REVOKE revoca privilegios en una tabla, vista o apodo.

Invocación:

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

Autorización:

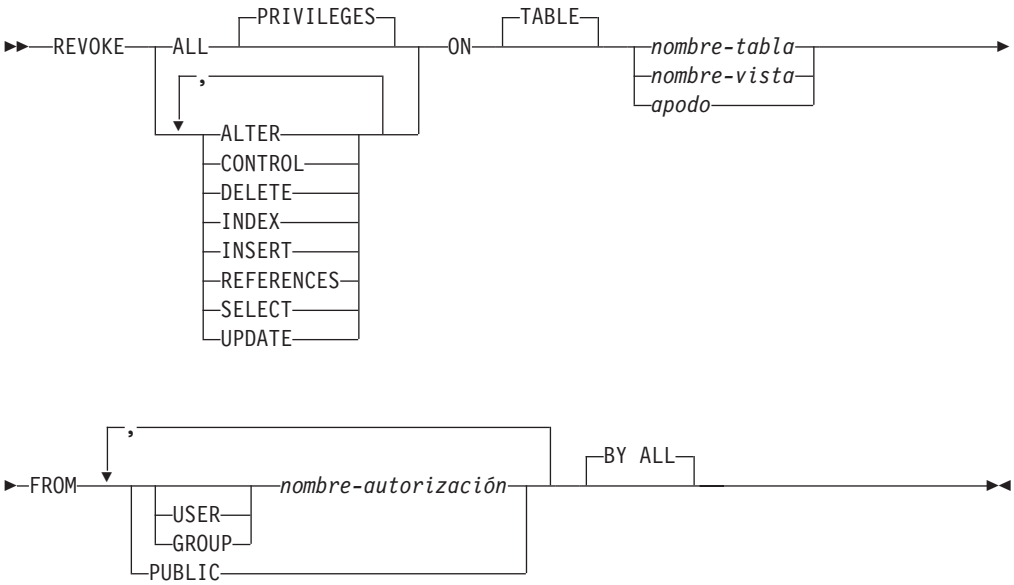
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización SYSADM o DBADM
- Privilegio CONTROL para la tabla, vista o apodo referenciado.

Para revocar el privilegio CONTROL, es necesaria la autorización SYSADM o DBADM.

Para revocar los privilegios sobre tablas y vistas de catálogo es necesaria la autorización SYSADM o DBADM.

Sintaxis:



Descripción:

ALL o **ALL PRIVILEGES**
Revoca todos los privilegios (excepto CONTROL) del nombre-autorización para las tablas, vistas o apodos especificados.

Si no se utiliza ALL, deben utilizarse una o varias de las palabras clave listadas abajo. Cada palabra clave revoca el privilegio descrito, pero sólo si se aplica a las tablas o vistas nombradas en la cláusula ON. No se debe especificar la misma palabra clave más de una vez.

ALTER
Revoca el privilegio para añadir columnas a la definición de la tabla base; crear o eliminar una clave primaria o restricción de unicidad en la tabla; crear o eliminar una clave foránea en la tabla; añadir/cambiar un comentario en la tabla, vista o apodo; crear o eliminar una restricción de comprobación; crear un activador; añadir, restablecer o eliminar una opción de columna para un apodo; o cambiar nombres de columna de apodo o tipos de datos.

CONTROL
Revoca la capacidad para eliminar la tabla base, vista o apodo y para ejecutar el programa de utilidad RUNSTATS sobre la tabla y los índices.

La revocación del privilegio CONTROL en un *nombre-autorización* no revoca otros privilegios otorgados al usuario de dicho objeto.

DELETE
Revoca el privilegio para suprimir filas de la tabla, de la vista que puede actualizarse o del apodo.

INDEX
Revoca el privilegio para crear un índice en la tabla o una especificación de índice en el apodo. El creador de un índice o de una especificación de índice tiene automáticamente el privilegio CONTROL en el índice o en la especificación de índice (que autoriza al creador a eliminar el índice o la especificación de índice). Así mismo, el creador mantiene este privilegio incluso si se revoca el privilegio INDEX.

INSERT
Revoca los privilegios para insertar filas en la tabla, en la vista que puede actualizarse o en el apodo y para ejecutar el programa de utilidad IMPORT.

REFERENCES
Revoca el privilegio para crear o eliminar una clave foránea que haga referencia a la tabla como padre. Cualquier privilegio REFERENCES de nivel de columna también se revoca.

SELECT
Revoca el privilegio para recuperar filas de la tabla o vista, para crear una vista en una tabla y para ejecutar el programa de utilidad EXPORT sobre la tabla o vista.

La revocación del privilegio SELECT puede provocar que algunas vistas se marquen como no operativas. (Para obtener información acerca de las vistas no operativas, consulte “CREATE VIEW”.)

UPDATE
Revoca el privilegio para actualizar filas en la tabla, en la vista que puede actualizarse o en el apodo. Cualquier privilegio UPDATE de nivel de columna también se revoca.

ON TABLE *nombre-tabla* o *nombre-vista* o *apodo*
Especifica la tabla, vista o apodo en los que se deben revocar los privilegios. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

FROM
Indica de quién se deben revocar los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista uno o varios ID de autorización.

El ID de la propia sentencia REVOKE no se puede utilizar (SQLSTATE 42502). No se pueden revocar privilegios para un *nombre-autorización* que sea igual que el ID de autorización de la sentencia REVOKE.

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas:

- Si no se especifica USER ni GROUP, entonces:
  - Si todas las filas para el destinatario en la vista de catálogo SYSCAT.TABAUTH y SYSCAT.COLAUTH tienen GRANTEETYPE de U, entonces se supone USER.
  - Si todas las filas tienen GRANTEETYPE de G, entonces se supone GROUP.
  - Si algunas filas tienen U y otras tienen G, se genera un error (SQLSTATE 56092).

Notas:

- Si se revoca un privilegio del *nombre-autorización* utilizado para crear una vista (se denomina DEFINER de la vista en SYSCAT.VIEWS), dicho privilegio también se revoca de las vistas dependientes.
- Si el usuario que define la vista, DEFINER, pierde un privilegio SELECT para algún objeto del que depende la definición de vista (o si se elimina un objeto del que depende la definición de vista o se convierte en no operativo en el caso de otra vista), la vista se convertirá en no operativa.

  Sin embargo, si DBADM o SYSADM revoca explícitamente todos los privilegios del DEFINER sobre la vista, el registro del DEFINER no aparecerá en SYSCAT.TABAUTH, pero no le sucederá nada a la vista - continuará operativa.
- Los privilegios en vistas no operativas no pueden revocarse.
- Todos los paquetes que dependen de un objeto para el que se revoca un privilegio se marcan como no válidos. Un paquete continúa siendo no válido hasta que se ejecuta satisfactoriamente una operación de enlace lógico o de volver a enlazar lógicamente en la aplicación, o la aplicación se ejecuta y el gestor de bases de datos vuelve a enlazar la aplicación satisfactoriamente (utilizando la información almacenada en los catálogos). Los paquetes marcados como no válidos debido a una revocación pueden volverse a enlazar satisfactoriamente sin ninguna operación de otorgar adicional.

  Por ejemplo, si un paquete propiedad de USER1 contiene SELECT de la tabla T1 y se revoca el privilegio SELECT para la tabla T1 del USER1, el paquete se marcará como no válido. Si se vuelve a otorgar la autorización SELECT, o si el usuario tiene la autorización DBADM, el paquete se vuelve a enlazar satisfactoriamente cuando se ejecuta.
- Paquetes, activadores o vistas que incluyen la utilización de OUTER(*Z*) en la cláusula FROM, dependen de tener el privilegio SELECT en cada subtabla o subvista de *Z*. De modo similar, paquetes, activadores o vistas que incluyen la utilización de DEREF(*Y*) donde *Y* es un tipo de referencia con una vista o tabla de destino *Z*, dependen de tener el privilegio SELECT en cada subtabla o

subvista de Z. Si se revoca uno de estos privilegios SELECT, los paquetes se invalidan y los activadores y vistas pasan a ser no operativas.

- Los privilegios de tabla, vista o apodo no pueden revocarse de un *nombre-autorización* con CONTROL en el objeto sin revocar también el privilegio CONTROL (SQLSTATE 42504).
- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir con su tarea si PUBLIC o un grupo poseen otros privilegios, o si tienen privilegios como, por ejemplo, ALTERIN en el esquema de una tabla o vista.
- Si el usuario que define la tabla de consultas materializadas, DEFINER, pierde un privilegio SELECT para una tabla de la que depende la definición de la tabla de consultas materializadas (o si se elimina una tabla de la que depende la definición de la tabla de consultas materializadas), la tabla de consultas materializadas se convertirá en no operativa.

  Sin embargo, se DBADM o SYSADM revoca explícitamente todos los privilegios para la tabla de consultas materializadas de DEFINER, el registro de SYSTABAUTH para DEFINER se suprimirá, pero no le sucederá nada a la tabla de consultas materializadas - seguirá siendo operativa.
- Revocar privilegios de apodo no tiene efecto sobre los privilegios de objeto de la fuente de datos (tabla o vista).
- La revocación del privilegio SELECT para una tabla o vista a la que se hace referencia directa o indirectamente en una función de SQL o cuerpo de método podría no ejecutarse satisfactoriamente si la función de SQL o el cuerpo de método no puede eliminarse porque algún otro objeto depende de éste (SQLSTATE 42893).
- Si el usuario que define la función de SQL o el cuerpo de método, DEFINER, pierde el privilegio SELECT para algún objeto del que depende la definición de la función o del cuerpo de método (o si se elimina un objeto del que depende la definición de la función o del cuerpo de método), la función o el cuerpo de método se eliminará, a menos que otro objeto dependa de la función o del método (SQLSTATE 42893).

Ejemplos:

Ejemplo 1: Revoque el privilegio SELECT en la tabla EMPLOYEE del usuario ENGLES. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.

```
REVOKE SELECT
  ON TABLE EMPLOYEE
  FROM ENGLES
```

Ejemplo 2: Revoque los privilegios de actualización en la tabla EMPLOYEE que se han otorgado previamente a todos los usuarios locales. Tenga en cuenta que no afecta a los usuarios a los que se les ha otorgado de manera específica.

```
REVOKE UPDATE
  ON EMPLOYEE
  FROM PUBLIC
```

Ejemplo 3: Revoque todos los privilegios en la tabla EMPLOYEE de los usuarios PELLOW y MLI y del grupo PLANNERS.

```
REVOKE ALL
  ON EMPLOYEE
  FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

Ejemplo 4: Revoque el privilegio SELECT en la tabla CORPDATA.EMPLOYEE de un usuario llamado JOHN. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor de GRANTEETYPE es U.

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM JOHN
```

o

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM USER JOHN
```

Tenga en cuenta que el intento de revocar el privilegio de GROUP JOHN daría como resultado un error, ya que el privilegio no estaba otorgado previamente a GROUP JOHN.

Ejemplo 5: Revoque el privilegio SELECT en la tabla CORPDATA.EMPLOYEE de un grupo llamado JOHN. Hay una fila en la vista de catálogo SYSCAT.TABAUTH para esta tabla y el destinatario de la operación de otorgar y el valor GRANTEETYPE es G.

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM JOHN
```

o

```
REVOKE SELECT
  ON CORPDATA.EMPLOYEE FROM GROUP JOHN
```

Ejemplo 6: Revoque el privilegio SHAWN de usuario para crear una especificación de índice en el apodo ORAREM1.

```
REVOKE INDEX
  ON ORAREM1 FROM USER SHAWN
```

Información relacionada:

- “CREATE TABLE” en la página 341
- “CREATE VIEW” en la página 466
- “DROP” en la página 512
- “REVOKE (Autorizaciones de base de datos)” en la página 658
- “REVOKE (Privilegios de índice)” en la página 662
- “REVOKE (Privilegios de paquete)” en la página 664
- “REVOKE (Privilegios de esquema)” en la página 670
- “REVOKE (Privilegios de servidor)” en la página 675
- “REVOKE (Privilegios de espacio de tabla)” en la página 677
- “REVOKE (Privilegios de rutina)” en la página 667

Ejemplos relacionados:

- “tbpriv.sqc -- How to grant, display, and revoke privileges (C)”
- “tbpriv.sqC -- How to grant, display, and revoke privileges (C++)”
- “TbPriv.java -- How to grant, display and revoke privileges on a table (JDBC)”
- “TbPriv.sqlj -- How to grant, display and revoke privileges on a table (SQLj)”

ROLLBACK

La sentencia ROLLBACK se utiliza para restituir los cambios que se han hecho en la base de datos dentro de una unidad de trabajo o punto de salvaguarda.

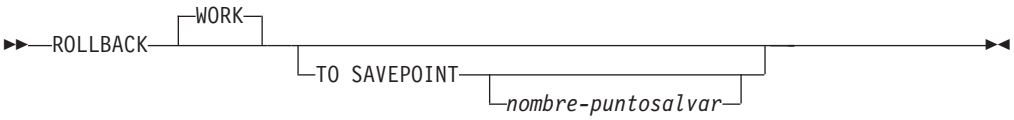
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:
La unidad de trabajo en la que se ejecuta la sentencia ROLLBACK se termina y se inicia una nueva unidad de trabajo. Se restituyen todos los cambios realizados en la base de datos durante la unidad de trabajo.

Sin embargo, las sentencias siguientes no están bajo el control de la transacción y los cambios que realicen serán independientes de la emisión de la sentencia ROLLBACK:

- SET CONNECTION
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT PACKAGESET
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET PASSTHRU

  Nota: Aunque la sentencia SET PASSTHRU no está bajo el control de la transacción, la sesión PASSTHRU que ha iniciado la sentencia está bajo el control de la transacción.
- SET PATH
- SET SCHEMA
- SET SERVER OPTION

La generación de valores de secuencia y de identidad no están bajo el control de la transacción. Los valores generados y consumidos por la *expresión-nextval* o por la inserción de filas en una tabla que contiene una columna de identidad son independientes de la emisión de la sentencia ROLLBACK. Asimismo, la emisión de la sentencia ROLLBACK no afecta al valor devuelto por la *expresión-prevval*, ni la función IDENTITY_VAL_LOCAL.

TO SAVEPOINT
Especifica que debe realizarse una retrotracción parcial (ROLLBACK TO SAVEPOINT). Si no hay ningún punto de salvaguarda activo en el nivel del punto de salvaguarda actual (consulte la sección "Normas" de la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B502). Tras una retrotracción satisfactoria, el punto de salvaguarda continúa existiendo, pero se liberan los puntos de salvaguarda anidados y ya no existen. Se considera que los puntos de salvaguarda anidados, si existen, se han retrotraído y después liberado como parte de la retrotracción hasta el punto de salvaguarda actual. Si no se facilita un *nombre-puntosalvaguarda*, la retrotracción se produce hasta el punto de salvaguarda definido más recientemente dentro del nivel de punto de salvaguarda actual.

Si se omite esta cláusula, la sentencia ROLLBACK retrotrae toda la transacción. Además, se liberan todos los puntos de salvaguarda definidos dentro de la transacción.

nombre-puntosalvar
Especifica el punto de salvaguarda que se debe utilizar en la operación de retrotracción. El *nombre-puntosalvaguarda* especificado no puede empezar por 'SYS' (SQLSTATE 42939). Tras una operación de retrotracción satisfactoria, el punto de salvaguarda nombrado continúa existiendo. Si el nombre de punto de salvaguarda no existe, se devuelve un error (SQLSTATE 3B001). Se deshacen los cambios que se han hecho en datos y esquemas desde que se definió el punto de salvaguarda.

Notas:

- Cuando se ejecuta un ROLLBACK de la unidad de trabajo se liberan todos los bloqueos mantenidos. Se cierran todos los cursores abiertos. Se liberan todos los localizadores de LOB.
- La ejecución de la sentencia ROLLBACK no afecta a las sentencias SET que cambian los valores del registro especial ni a la sentencia RELEASE.
- Si el programa finaliza de forma anómala, la unidad de trabajo se retrotrae implícitamente.
- La colocación de sentencias en antememoria se ve afectada por la operación de retrotracción.
- El efecto que una sentencia ROLLBACK TO SAVEPOINT tiene sobre los cursores depende de las sentencias contenidas en el punto de salvaguarda.
  - Si el punto de salvaguarda contiene un DDL del cual depende un cursor, el cursor se marca como no válido. Los intentos para utilizar ese cursor dan lugar a un error (SQLSTATE 57007).
  - En otro caso:
    - Si se hace referencia al cursor en el punto de salvaguarda, el cursor permanece abierto y se coloca delante de la primera fila lógica de la tabla de resultados. (FETCH debe realizarse antes de emitirse una sentencia UPDATE o DELETE con posición.)
    - En otro caso, el cursor no queda afectado por ROLLBACK TO SAVEPOINT (permanece abierto y posicionado).

- Los nombres de sentencias preparadas dinámicamente siguen siendo válidos, aunque la sentencia puede volver a prepararse implícitamente, como resultado de operaciones DDL que se retrotraen dentro del punto de salvaguarda.
- Las operaciones ROLLBACK TO SAVEPOINT eliminan cualquier tabla temporal declarada que aparezca mencionada dentro del punto de salvaguarda. Si una tabla temporal declarada se modifica dentro del punto de salvaguarda, se suprimen todas las filas de la tabla.
- Después de una sentencia ROLLBACK TO SAVEPOINT se conservan todos los bloqueos.
- Después de una operación ROLLBACK TO SAVEPOINT se conservan todos los localizadores de LOB.

Ejemplo:

Suprima las modificaciones realizadas desde el último punto de confirmación o retrotracción.

```
ROLLBACK WORK
```

Información relacionada:

- “EXECUTE” en la página 541
- “SAVEPOINT” en la página 687

Ejemplos relacionados:

- “delet.sqb -- How to delete table data (MF COBOL)”
- “spclient.sqc -- Call various stored procedures (C)”
- “tut_use.sqc -- How to modify a database (C)”
- “spclient.sqC -- Call various stored procedures (C++)”
- “tut_use.sqC -- How to modify a database (C++)”

SAVEPOINT

Utilice la sentencia SAVEPOINT para definir un punto de salvaguarda dentro de una transacción.

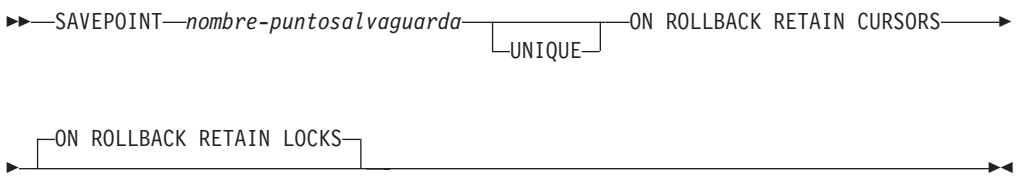
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación (también en procedimientos almacenados) o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:

nombre-puntosalvaguarda
Especifica el nombre de un punto de salvaguarda. El *nombre-puntosalvaguarda* no puede empezar por 'SYS' (SQLSTATE 42939). Si ya se ha definido un punto de salvaguarda con este nombre como UNIQUE dentro de este nivel de punto de salvaguarda, se devuelve un error (SQLSTATE 3B501).

UNIQUE
Especifica que la aplicación no intenta volver a utilizar este nombre de punto de salvaguarda mientras el punto de salvaguarda esté activo dentro del nivel de punto de salvaguarda actual. Si el *nombre-puntosalvaguarda* ya existe en este nivel de punto de salvaguarda, se devuelve un error (SQLSTATE 3B501).

ON ROLLBACK RETAIN CURSORS
Especifica la respuesta del sistema al realizar una retrotracción hasta este punto de salvaguarda, con respecto a las sentencias OPEN CURSOR procesadas después de la sentencia SAVEPOINT. Esta cláusula indica que, siempre que sea posible, los cursores no se ven afectados por una operación de retrotracción al punto de salvaguarda. Para obtener información acerca de las situaciones en las que los cursores pueden verse afectados por la retrotracción hasta el punto de salvaguarda, consulte "ROLLBACK".

ON ROLLBACK RETAIN LOCKS
Especifica la respuesta del sistema al realizar una retrotracción hasta este punto de salvaguarda, con respecto a los bloqueos adquiridos después de definir el punto de salvaguarda. No se hace un seguimiento de los bloqueos adquiridos a partir del punto de salvaguarda y no se retrotraen (liberan) al retrotraer hasta el punto de salvaguarda.

Normas:

- Las sentencias relacionadas con el punto de salvaguarda no se deben utilizar en definiciones de activador (SQLSTATE 42987).

- Un nuevo nivel de punto de salvaguarda empieza cuando se produce una de las siguientes situaciones:
  - Se inicia una nueva unidad de trabajo (UOW).
  - Se llama a un nuevo procedimiento definido con la cláusula NEW SAVEPOINT LEVEL.
  - Se inicia una sentencia de SQL compuesta atómica.
- Un nivel de punto de salvaguarda termina cuando finaliza o se elimina el suceso que ha provocado su creación. Cuando finaliza un nivel de punto de salvaguarda, se liberan todos los puntos de salvaguarda que contiene. Cualquier cursor abierto, acciones DDL o modificaciones de datos se heredan del nivel de punto de salvaguarda padre (es decir, el nivel de punto de salvaguarda en el cual se ha creado el que acaba de terminar) y están sujetos a cualquier sentencia relacionada con el punto de salvaguarda emitida en el nivel de punto de salvaguarda padre.
- Las normas siguientes se aplican a las acciones de un nivel de punto de salvaguarda:
  - Sólo se puede hacer referencia a los puntos de salvaguarda en el nivel de punto de salvaguarda en el que se establecen. No se puede liberar ni destruir un punto de salvaguarda establecido fuera del nivel actual de punto de salvaguarda, ni tampoco realizar una retrotracción hasta él.
  - Todos los puntos de salvaguarda activos establecidos en el nivel de punto de salvaguarda actual se liberan automáticamente cuando finaliza el nivel de punto de salvaguarda.
  - La exclusividad de los nombres de punto de salvaguarda sólo es obligatoria en el nivel del punto de salvaguarda actual. Los nombres de los puntos de salvaguarda que están activos en otros niveles de punto de salvaguarda pueden reutilizarse en el nivel de punto de salvaguarda actual sin que afecte a los puntos de salvaguarda de los otros niveles de punto de salvaguarda.

Notas:

- Cuando se ha emitido una sentencia SAVEPOINT, las operaciones para realizar inserciones, actualizaciones o supresiones en los apodos no están permitidas.
- La omisión de la cláusula UNIQUE especifica que el otro punto de salvaguarda puede reutilizar el *nombre-puntosalvaguarda* en el nivel de punto de salvaguarda. Si un punto de salvaguarda del mismo nombre ya existe en el nivel de punto de salvaguarda, el punto de salvaguarda existente se destruye y se crea un nuevo punto de salvaguarda con el mismo nombre en el punto actual del proceso. El nuevo punto de salvaguarda se considera como el último punto de salvaguarda establecido por la aplicación. Tenga en cuenta que la destrucción de un punto de salvaguarda mediante la reutilización de su nombre por otro punto de salvaguarda simplemente destruye ese punto de salvaguarda y no libera ningún punto de salvaguarda establecido después del punto de salvaguarda destruido. Estos puntos de salvaguarda subsiguientes sólo pueden liberarse mediante la sentencia RELEASE SAVEPOINT, que libera el punto de salvaguarda nombrado y todos los puntos de salvaguarda establecidos después del punto de salvaguarda nombrado.
- Si se especifica la cláusula UNIQUE, el *nombre-puntosalvaguarda* sólo puede reutilizarse después de que se haya liberado un punto de salvaguarda existente con el mismo nombre.
- En un punto de salvaguarda, si un programa de utilidad, una sentencia de SQL o un mandato DB2 realiza confirmaciones intermitentes durante el proceso, el punto de salvaguarda se liberará implícitamente.

- Si la sentencia SET INTEGRITY se retrotrae dentro del punto de salvaguarda, los nombres de sentencias preparadas dinámicamente siguen siendo válidos, aunque la sentencia pueda prepararse de nuevo implícitamente.
- Si las inserciones se colocan en almacenamiento intermedio (es decir, la aplicación se ha compilado previamente con la opción INSERT BUF), el almacenamiento intermedio se vaciará cuando se emitan las sentencias SAVEPOINT, ROLLBACK o RELEASE TO SAVEPOINT.

Ejemplo:

Ejemplo 1: Realice una operación de retrotracción para los puntos de salvaguarda anidados. Primero, cree una tabla denominada DEPARTMENT. Inserte una fila antes de iniciar SAVEPOINT1; inserte otra fila e inicie SAVEPOINT2; después, inserte una tercera fila e inicie SAVEPOINT3.

```
CREATE TABLE DEPARTMENT (
  DEPTNO   CHAR(6),
  DEPTNAME VARCHAR(20),
  MGRNO    INTEGER)

  INSERT INTO DEPARTMENT VALUES ('A20', 'MARKETING', 301)

  SAVEPOINT SAVEPOINT1 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('B30', 'FINANCE', 520)

  SAVEPOINT SAVEPOINT2 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('C40', 'IT SUPPORT', 430)

  SAVEPOINT SAVEPOINT3 ON ROLLBACK RETAIN CURSORS

  INSERT INTO DEPARTMENT VALUES ('R50', 'RESEARCH', 150)
```

En este momento, la tabla DEPARTMENT existe con las filas A20, B30, C40 y R50. Si ahora emite:

```
ROLLBACK TO SAVEPOINT SAVEPOINT3
```

la fila R50 ya no estará en la tabla DEPARTMENT. Si después emite:

```
ROLLBACK TO SAVEPOINT SAVEPOINT1
```

la tabla DEPARTMENT todavía existirá, pero las filas insertadas desde que se ha establecido SAVEPOINT1 (B30 y C40) ya no estarán en la tabla.

Información relacionada:

- "ROLLBACK" en la página 684

SELECT

La sentencia SELECT es una forma de consulta. Puede incorporarse en un programa de aplicación o emitirse interactivamente.

Información relacionada:

- "Subselección" en la publicación *Consulta de SQL, Volumen 1*
- "Sentencia select" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dynamic.sqb -- How to update table data with cursor dynamically (MF COBOL)"
- "static.sqb -- Get table data using static SQL statement (MF COBOL)"
- "tbread.c -- How to read data from tables"
- "tut_read.c -- How to read data from tables"
- "tbread.sqc -- How to read tables (C)"
- "tut_read.sqc -- How to read tables (C)"
- "tbread.sqC -- How to read tables (C++)"
- "tut_read.sqC -- How to read tables (C++)"
- "TbRead.java -- How to read table data (JDBC)"
- "TutRead.java -- Read data in a table (JDBC)"
- "TbRead.sqlj -- How to read table data (SQLj)"
- "TutRead.sqlj -- Read data in a table (SQLj)"

SELECT INTO

La sentencia SELECT INTO produce una tabla de resultados que consta de como máximo una fila y asigna los valores de dicha fila a las variables del lenguaje principal. Si la tabla está vacía, la sentencia asigna +100 a SQLCODE y '02000' a SQLSTATE y no asigna valores a las variables del lenguaje principal. Si más de una fila satisface la condición de búsqueda, se termina el proceso de la sentencia y se produce un error (SQLSTATE 21000).

Invocación:

Esta sentencia sólo se puede incluir en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

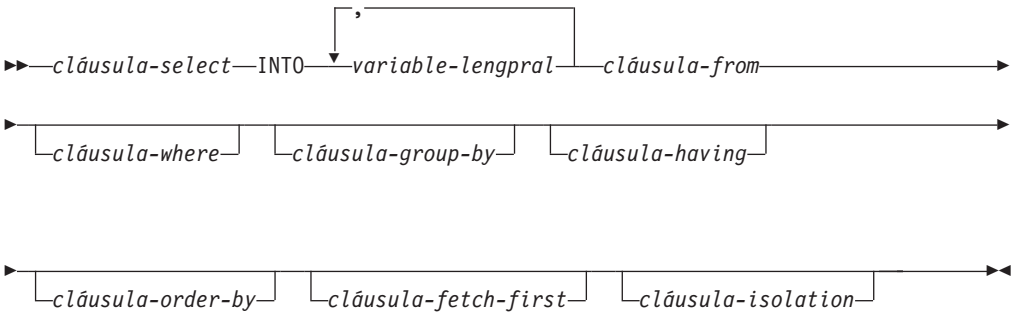
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio SELECT para la tabla, vista o apodo
- Privilegio CONTROL para la tabla, vista o apodo
- Autorización SYSADM o DBADM

Los privilegios GROUP no se comprueban para las sentencias SELECT INTO estáticas.

Si el destino de la sentencia SELECT es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis:



Descripción:
Para obtener una descripción de la *cláusula-select*, la *cláusula-from*, la *cláusula-where*, la *cláusula-group-by*, la *cláusula-having*, la *cláusula-order-by*, la *cláusula-fetch-first* y la *cláusula-isolation*, vea "Consultas" en la publicación *Consulta de SQL, Volumen 1*.

INTO
Introduce una lista de variables del lenguaje principal.

variable-lengpral
Identifica una variable que está descrita en el programa bajo las normas para la declaración de variables del lenguaje principal.

El primer valor de la fila del resultado se asigna a la primera variable de la lista, el segundo valor a la segunda variable, etcétera. Si el número de variables del lenguaje principal es menor que el número de valores de columna, se asigna el valor 'W' al campo SQLWARN3 de la SQLCA.

Cada asignación que se realiza para una variable se realiza secuencialmente y siguiendo la lista. Si se produce un error, no se asigna ningún valor a la variable del lenguaje principal.

Ejemplos:

Ejemplo 1: Este ejemplo C coloca el salario máximo de la tabla EMP en la variable del lenguaje principal MAXSALARY.

```
EXEC SQL SELECT MAX(SALARY)
  INTO :MAXSALARY
  FROM EMP;
```

Ejemplo 2: Este ejemplo C coloca la fila del empleado 528671 (de la tabla EMP) en las variables del lenguaje principal.

```
EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
  FROM EMP
  WHERE EMPNO = '528671';
```

Ejemplo 3: Este ejemplo SQLJ coloca la fila del empleado 528671 (de la tabla EMP) en las variables del lenguaje principal. Esa fila se actualizará más tarde con una actualización de búsqueda y se debe bloquear cuando se ejecuta la consulta.

```
#sql { SELECT * INTO :FIRSTNAME, :LASTNAME, :EMPNO, :SALARY
  FROM EMP
  WHERE EMPNO = '528671'
  WITH RS USE AND KEEP EXCLUSIVE LOCKS };
```

Información relacionada:

- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*
- "Consultas de SQL" en la publicación *Consulta de SQL, Volumen 1*
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbauth.sqc -- How to grant, display, and revoke authorities at database level (C)"
- "dtlob.sqc -- How to use the LOB data type (C)"
- "spclient.sqc -- Call various stored procedures (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "tbreorg.sqc -- How to reorganize a table and update its statistics (C)"
- "dbauth.sqC -- How to grant, display, and revoke authorities at database level (C++)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "spclient.sqC -- Call various stored procedures (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "tbreorg.sqC -- How to reorganize a table and update its statistics (C++)"

SET CONNECTION

La sentencia SET CONNECTION cambia el estado de una conexión de inactivo a actual, convirtiendo la ubicación especificada en el servidor actual. No está bajo el control de la transacción.

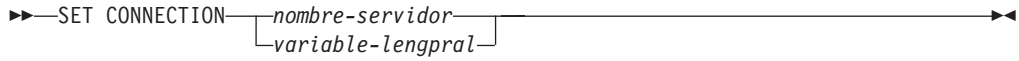
Invocación:

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna.

Sintaxis:



Descripción:

nombre-servidor o *variable-lengpral*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lengpral* que contenga el *nombre-servidor*.

Si se especifica una *variable-lengpral*, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 8 y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lengpral* debe estar justificado por la izquierda y no estar delimitado por comillas.

Observe que el *nombre-servidor* es un alias de base de datos que identifica al servidor de aplicaciones. Debe aparecer en la lista del directorio local del peticionario de aplicaciones.

El *nombre-servidor* o la *variable-lengpral* debe identificar una conexión existente del proceso de aplicación. Si no identifican ninguna conexión existente, se genera un error (SQLSTATE 08003).

Si SET CONNECTION se aplica a la conexión actual, no se cambian los estados de ninguna de las conexiones del proceso de aplicación.

Conexión satisfactoria

Si la sentencia SET CONNECTION se ejecuta satisfactoriamente:

- No se realiza ninguna conexión. El registro especial CURRENT SERVER se actualiza con el *nombre-servidor* especificado.
- La conexión actual previa, si la hay, se coloca en el estado inactivo (suponiendo que se haya especificado un *nombre-servidor* distinto).
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma forma que con “CONNECT (Tipo 1)”, donde encontrará información adicional.

Conexión no satisfactoria

Si la sentencia SET CONNECTION falla:

- No importa cuál haya sido la razón de la anomalía, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.
- Al igual que para un CONNECT de Tipo 1 no satisfactorio, el campo SQLERRP de la SQLCA se establece en el nombre del módulo que detectó el error.

Notas:

- La utilización de sentencias CONNECT de tipo 1 no excluye la utilización de SET CONNECTION, pero la sentencia fallará siempre (SQLSTATE 08003), a menos que la sentencia SET CONNECTION especifique la conexión actual, ya que no pueden existir conexiones inactivas.
- La opción de conexión SQLRULES(DB2) (consulte el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida") no imposibilita la utilización de SET CONNECTION, pero la sentencia no es necesaria porque, en su lugar, pueden utilizarse sentencias CONNECT de tipo 2.
- Cuando se utiliza una conexión, se inactiva y después se restaura al estado actual en la misma unidad de trabajo, dicha conexión refleja su última utilización por el proceso de aplicación en relación con el estado de bloqueos, cursores y sentencias preparadas.

Ejemplos:

Ejecute las sentencias de SQL de IBMSTHDB, ejecute las sentencias de SQL de IBMTOKDB y después ejecute más sentencias de SQL de IBMSTHDB.

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
EXEC SQL CONNECT TO IBMTOKDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMTOKDB */
EXEC SQL SET CONNECTION IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
```

Observe que la primera sentencia CONNECT crea la conexión IBMSTHDB, la segunda sentencia CONNECT lo coloca en el estado inactivo y la sentencia SET CONNECTION la devuelve al estado actual.

Conceptos relacionados:

- "Bases de datos relacionales distribuidas" en la publicación *Consulta de SQL, Volumen 1*

Información relacionada:

- "CONNECT (Tipo 1)" en la página 148

Ejemplos relacionados:

- "dbmcon.sqc -- How to use multiple databases (C)"
- "dbmcon.sqC -- How to use multiple databases (C++)"

SET CURRENT DEFAULT TRANSFORM GROUP

La sentencia SET CURRENT DEFAULT TRANSFORM GROUP cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP. La sentencia no está bajo el control de la transacción.

Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:

Descripción:

nombre-grupo
Especifica un nombre, formado por un solo elemento, que identifica un grupo de transformación definido para todos los tipos estructurados. Este nombre se puede utilizar en sentencias subsiguientes (o hasta que el valor del registro especial se vuelve a cambiar utilizando otra sentencia SET CURRENT DEFAULT TRANSFORM GROUP).

El nombre debe ser un identificador SQL, de hasta 18 caracteres de longitud (SQLSTATE 42815). Cuando se establece el registro especial, no se realiza ninguna validación de que el *nombre-grupo* esté definido para cualquier tipo estructurado. Sólo se comprueba la validez de la definición del grupo de transformación mencionado cuando se referencia explícitamente un tipo estructurado.

Normas:

- Si el valor especificado no se adapta a las normas para un *nombre-grupo*, se emite un error (SQLSTATE 42815)
- Las funciones TO SQL y FROM SQL definidas en el grupo de transformación *nombre-grupo* se utilizan para intercambiar datos de tipo estructurado, definidos por el usuario, con un programa del lenguaje principal.

Notas:

- El valor inicial del registro especial CURRENT DEFAULT TRANSFORM GROUP es la serie de caracteres vacía.

Ejemplos:

Ejemplo 1: Establecimiento del grupo de transformación por omisión en MYSTRUCT1. Este ejemplo utiliza las funciones TO SQL y FROM SQL definidas en el grupo de transformación MYSTRUCT1 para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa actual del lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

Información relacionada:

- "Registro especial CURRENT DEFAULT TRANSFORM GROUP" en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT DEGREE

La sentencia SET CURRENT DEGREE asigna un valor al registro especial CURRENT DEGREE. La sentencia no está bajo el control de la transacción.

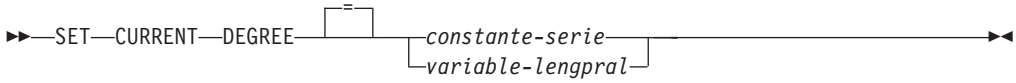
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:
El valor de CURRENT DEGREE se sustituye por el valor de la constante de serie o de la variable del lenguaje principal. El valor debe ser una serie de caracteres que no tenga una longitud superior a 5 bytes. El valor debe ser la representación de serie de caracteres de un entero entre 1 y 32.767 inclusive o 'ANY'.

Si el valor de CURRENT DEGREE representado como un entero es 1 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esta sentencia no utilizará el paralelismo intrapartición.

Si el valor de CURRENT DEGREE es un número cuando se prepara una sentencia de SQL dinámicamente, la ejecución de dicha sentencia puede implicar un paralelismo intrapartición con el grado especificado.

Si el valor de CURRENT DEGREE es 'ANY' cuando una sentencia de SQL se prepara dinámicamente, la ejecución de dicha sentencia puede implicar el paralelismo intrapartición que utiliza un grado determinado por el gestor de bases de datos.

variable-lengpral
El tipo de datos de la *variable-lengpral* debe ser CHAR o VARCHAR y su longitud no debe ser mayor que 5. Si el campo es más largo, se emite un error (SQLSTATE 42815). Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

constante-serie
La longitud de la *constante-serie* no debe exceder de 5.

Notas:

El grado de paralelismo intrapartición para las sentencias de SQL puede controlarse utilizando la opción DEGREE del mandato PREP o BIND.

El grado de ejecución real del paralelismo intrapartición será inferior a:
- Parámetro de configuración de grado máximo de consulta (max_querydegree)
- El grado de ejecución de la aplicación
- El grado de compilación de la sentencia de SQL

La configuración del gestor de bases de datos intraparalelo debe estar activada para poder utilizar el paralelismo intrapartición. Si está desactivada, se pasará por alto el valor de este registro y la sentencia no utilizará el paralelismo intrapartición con el fin de optimización (SQLSTATE 01623).

Algunas sentencias de SQL no pueden utilizar el paralelismo intrapartición.

Ejemplo:

Ejemplo 1: La siguiente sentencia establece CURRENT DEGREE para que inhiba el paralelismo intrapartición.

```
SET CURRENT DEGREE = '1'
```

Ejemplo 2: La siguiente sentencia establece CURRENT DEGREE para que permita el paralelismo intrapartición.

```
SET CURRENT DEGREE = 'ANY'
```

SET CURRENT EXPLAIN MODE

La sentencia SET CURRENT EXPLAIN MODE cambia el valor del registro especial CURRENT EXPLAIN MODE. No está bajo el control de la transacción.

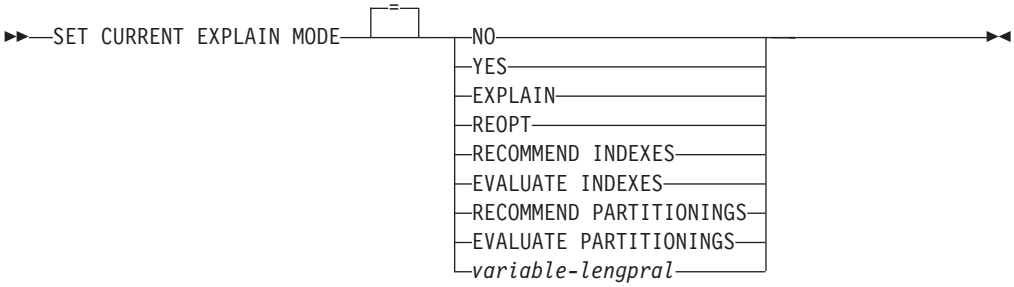
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización especial para ejecutar esta sentencia.

Sintaxis:



Descripción:

NO
Inhabilita el recurso Explain. No se captura la información de Explain. NO es el valor inicial del registro especial.

YES
Habilita el recurso Explain y provoca que la información de Explain se inserte en las tablas de Explain para las sentencias de SQL dinámico elegibles. Todas las sentencias de SQL dinámico se compilan y ejecutan normalmente.

EXPLAIN
Habilita el recurso Explain y provoca que se capte la información de Explain para cualquier sentencia de SQL dinámica elegible que esté preparada. Sin embargo, no se ejecutan las sentencias dinámicas.

REOPT
Habilita el recurso Explain y hace que la información de Explain se capture para una sentencia de SQL estático o dinámico durante la reoptimización de la sentencia en tiempo de ejecución; es decir, cuando los valores reales de las variables del lenguaje principal, los registros especiales o los marcadores de parámetros están disponibles.

RECOMMEND INDEXES
Habilite el compilador SQL para los índices recomendados. Todas las consultas que se ejecutan en esta modalidad de Explain llenarán la tabla ADVISE_INDEX con los índices recomendados. Así mismo, información de Explain será capturada en tablas de Explain para demostrar cómo se utilizan los índices recomendados, pero las sentencias no se compilan ni se ejecutan.

EVALUATE INDEXES
Habilita el compilador SQL para evaluar los índices. Los índices que deben evaluarse se leen de la tabla ADVISE_INDEX y deben marcarse con EVALUATE = Y. El optimizador genera índices virtuales basados en los valores de los catálogos. Todas las peticiones que se ejecutan en esta modalidad Explain serán compilados y optimizados utilizando estadísticas estimadas basadas en los índices virtuales. No se ejecutan las sentencias.

RECOMMEND PARTITIONINGS
Especifica que el compilador debe recomendar la mejor partición para cada tabla a la que accede una consulta específica. Las mejores particiones se graban en una tabla ADVISE_PARTITION. La consulta no se ejecuta.

EVALUATE PARTITIONINGS
Especifica que el compilador debe obtener el rendimiento estimado de una consulta utilizando las particiones virtuales especificadas en la tabla ADVISE_PARTITION.

variable-lengpral
El tipo de datos de la *variable-lengpral* debe ser CHAR o VARCHAR y su longitud no debe ser mayor que 254. Si se proporciona un campo más largo, se devolverá un error (SQLSTATE 42815). El valor especificado debe ser NO, YES, EXPLAIN, RECOMMEND INDEXES o EVALUATE INDEXES. Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

Notas:

- La información de Explain para las sentencias de SQL estático se puede capturar utilizando la opción EXPLAIN del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLAIN y el valor de registro CURRENT EXPLAIN MODE es NO, se capturará la información de Explain para las sentencias de SQL dinámico en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN MODE no es NO, el valor de la opción de enlace EXPLAIN se pasará por alto.
- RECOMMEND INDEXES y EVALUATE INDEXES son modalidades especiales que sólo pueden establecerse con la sentencia SET CURRENT EXPLAIN MODE. Estas modalidades no pueden establecerse utilizando opciones PREP o BIND y no funcionan con la sentencia SET CURRENT EXPLAIN SNAPSHOT.
- Si se activa el recurso Explain, el ID de autorización actual deberá disponer de privilegio INSERT para las tablas de Explain o se generará un error (SQLSTATE 42501).
- Cuando las sentencias de SQL se explican a partir de una rutina, la rutina debe definirse con un indicador de acceso a datos de SQL de MODIFIES SQL DATA (SQLSTATE 42985).
- Si el registro especial se establece en REOPT, y la sentencia de SQL no está cualificada para la reoptimización en tiempo de ejecución (es decir, si la sentencia no tiene variables de entrada o si la opción de vínculo REOPT se ha establecido en NONE), no se capturará información de Explain. Si la opción de vinculación REOPT se establece en ONCE, la información de Explain sólo se capturará una vez, al reoptimizar inicialmente la sentencia. Después de almacenar la sentencia en antememoria, no se adquirirá más información de Explain para esta sentencia en ejecuciones subsiguientes.

- Si se habilita el recurso Explain, se establece la opción de vínculo REOPT en ONCE y se intenta ejecutar una sentencia de SQL que ya está en antememoria, la sentencia se compilará y reoptimizará con los valores actuales de las variables de entradas y las tablas Explain se rellenarán de acuerdo a ello. El plan de acceso que se acaba de generar para esta sentencia no se almacenará en antememoria ni se ejecutará. Las demás aplicaciones que ejecutan simultáneamente esta sentencia almacenada en antememoria continuarán con su ejecución y las nuevas peticiones de ejecución de esta sentencia tomarán el plan de acceso que ya está en la antememoria.
- El valor de REOPT para los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT alterará temporalmente el valor de las opciones de vinculación EXPLAIN y EXPLSNAP en tiempo de vinculación si una sentencia de SQL dinámico o estático tiene variables de entrada y la opción de vinculación REOPT se establece en ONCE o ALWAYS.

Ejemplo:

El ejemplo siguiente establece el registro especial CURRENT EXPLAIN MODE, de modo que se capturará información de Explain para cualquier sentencia de SQL dinámica que pueda elegirse posteriormente y la sentencia no se ejecutará.

```
SET CURRENT EXPLAIN MODE = EXPLAIN
```

Información relacionada:

- "Valores de registro de EXPLAIN" en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT EXPLAIN SNAPSHOT

La sentencia SET CURRENT EXPLAIN SNAPSHOT cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT. No está bajo el control de la transacción.

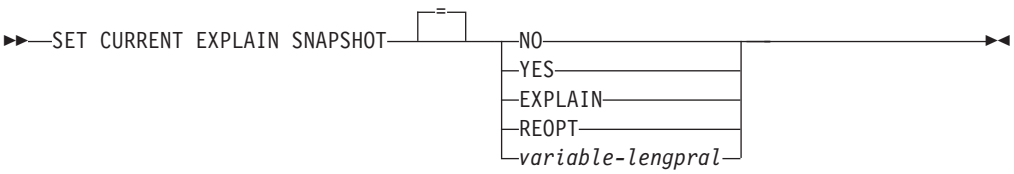
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

NO
Inhabilita el servicio de instantánea de Explain. No se toma ninguna instantánea. NO es el valor inicial del registro especial.

YES
Habilita el recurso de instantánea de Explain, creando una instantánea de la representación interna de cada sentencia de SQL dinámica elegible. Esta información se inserta en la columna SNAPSHOT de la tabla EXPLAIN_STATEMENT.

El recurso EXPLAIN SNAPSHOT se ha diseñado para utilizarse con Visual Explain.

EXPLAIN
Habilita el recurso de instantánea de Explain, creando una instantánea de la representación interna de cada sentencia de SQL dinámica elegible que se prepara. Sin embargo, no se ejecutan las sentencias dinámicas.

REOPT
Habilita el recurso Explain y hace que la información de Explain se capture para una sentencia de SQL estático o dinámico durante la reoptimización de la sentencia en tiempo de ejecución; es decir, cuando los valores reales de las variables del lenguaje principal, los registros especiales o los marcadores de parámetros están disponibles.

variable-lengpral
El tipo de datos de la *variable-lengpral* debe ser CHAR o VARCHAR y la longitud de su contenido no debe ser mayor que 8. Si el campo es más largo, se emite un error (SQLSTATE 42815). El valor contenido en el registro debe ser NO, YES ni EXPLAIN. Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los

valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Notas:

- Las instantáneas de Explain para las sentencias de SQL estático se pueden capturar utilizando la opción EXPLSNAP del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLSNAP y el valor de registro CURRENT EXPLAIN SNAPSHOT es NO, se capturarán instantáneas de Explain para las sentencias de SQL dinámico en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN SNAPSHOT no es NO, la opción EXPLSNAP se pasará por alto.
- Si se activa el recurso de instantánea de Explain, el ID de autorización actual debe tener el privilegio INSERT para las tablas de Explain o se genera un error (SQLSTATE 42501).
- Cuando las sentencias de SQL se explican a partir de una rutina, la rutina debe definirse con un indicador de acceso a datos de SQL de MODIFIES SQL DATA (SQLSTATE 42985).
- Si el registro especial se establece en REOPT, y la sentencia de SQL no está cualificada para la reoptimización en tiempo de ejecución (es decir, si la sentencia no tiene variables de entrada o si la opción de vínculo REOPT se ha establecido en NONE), no se capturará información de Explain. Si la opción de vinculación REOPT se establece en ONCE, la información de instantánea de Explain sólo se capturará una vez, al reoptimizar inicialmente la sentencia. Después de almacenar la sentencia en antememoria, no se adquirirá más información de Explain para esta sentencia en ejecuciones subsiguientes.
- Si se habilita el recurso Explain, se establece la opción de vinculación REOPT en ONCE y se intenta ejecutar una sentencia de SQL reoptimizable que ya está almacenada en antememoria, la sentencia se compilará y reoptimizará con los valores actuales de las variables de entrada y la instantánea de Explain se capturará de acuerdo a ello. El plan de acceso que se acaba de generar para esta sentencia no se almacenará en antememoria ni se ejecutará. Las demás aplicaciones que ejecutan simultáneamente esta sentencia almacenada en antememoria continuarán con su ejecución y las nuevas peticiones de ejecución de esta sentencia tomarán el plan de acceso que ya está en la antememoria.
- El valor de REOPT para los registros especiales CURRENT EXPLAIN MODE y CURRENT EXPLAIN SNAPSHOT alterará temporalmente el valor de las opciones de vinculación EXPLAIN y EXPLSNAP en tiempo de vinculación si una sentencia de SQL estático o dinámico tiene variables de entrada y la opción de vinculación REOPT se establece en ONCE o ALWAYS.

Ejemplos:

Ejemplo 1: La siguiente sentencia establece el registro especial CURRENT EXPLAIN SNAPSHOT de modo que se tomará una instantánea de Explain para cualquier sentencia de SQL dinámica elegible posterior y se ejecutará la sentencia.

```
SET CURRENT EXPLAIN SNAPSHOT = YES
```

Ejemplo 2: El ejemplo siguiente recupera el valor actual del registro especial CURRENT EXPLAIN SNAPSHOT en la variable del lenguaje principal llamada SNAP.

```
EXEC SQL VALUES (CURRENT EXPLAIN SNAPSHOT) INTO :SNAP;
```

Información relacionada:

- "Valores de registro de EXPLAIN" en la publicación *Consulta de SQL, Volumen 1*
- "Tabla EXPLAIN_STATEMENT" en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT ISOLATION

La sentencia SET CURRENT ISOLATION asigna un valor al registro especial CURRENT ISOLATION. La sentencia no está bajo el control de la transacción.

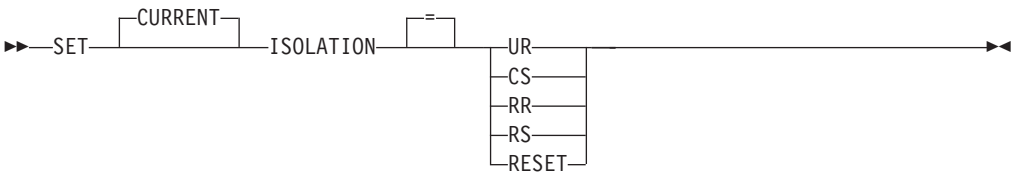
Invocación:

Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:
El valor del registro especial CURRENT ISOLATION se sustituye por el valor especificado o se establece en blancos si se especifica RESET.
Notas:

- *Compatibilidades*
  - También recibe soporte la sintaxis siguiente:
    - TO puede especificarse en lugar del signo de igual (=)
    - DIRTY READ puede especificarse en lugar de UR
    - READ UNCOMMITTED puede especificarse en lugar de UR
    - READ COMMITTED se reconoce y se actualiza a CS
    - CURSOR STABILITY puede especificarse en lugar de CS
    - REPEATABLE READ puede especificarse en lugar de RR
    - SERIALIZABLE puede especificarse en lugar de RR

Conceptos relacionados:

- "Niveles de aislamiento" en la publicación *Consulta de SQL, Volumen 1*

Información relacionada:

- "Registro especial CURRENT ISOLATION" en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT LOCK TIMEOUT

La sentencia SET CURRENT LOCK TIMEOUT cambia el valor del registro especial CURRENT LOCK TIMEOUT. No está bajo el control de la transacción.

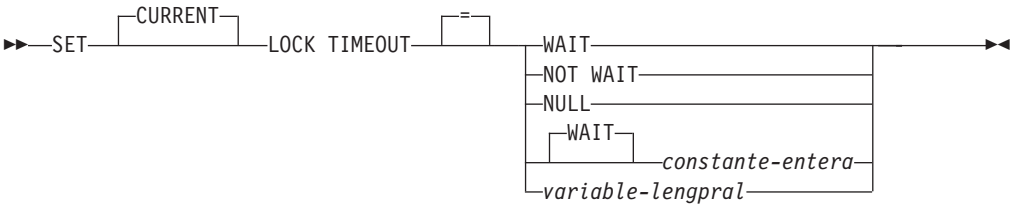
Invocación:

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:
El valor especificado debe ser un entero entre -1 y 32767, inclusive (SQLSTATE 428B7) o el valor nulo.

WAIT
Especifica un valor CURRENT LOCK TIMEOUT de -1, lo que significa que el gestor de bases de datos debe esperar hasta que se libere el bloqueo o se detecte un punto muerto (SQLSTATE 40001 o SQLSTATE 57033).

NOT WAIT
Especifica un valor de CURRENT LOCK TIMEOUT de 0, que significa que el gestor de bases de datos no debe esperar los bloqueos que no se pueden obtener y se devolverá un error (SQLSTATE 40001 o SQLSTATE 57033).

NULL
Especifica que se debe desestablecer el valor CURRENT LOCK TIMEOUT y que el valor del parámetro de configuración de base de datos *locktimeout* se debe utilizar cuando se espera un bloqueo. El valor que se devuelve para el registro especial cambiará cuando cambie el valor de *locktimeout*.

WAIT *constante-entero*
Especifica un valor de entero entre -1 y 32767. Un valor de -1 es equivalente a especificar la palabra clave WAIT sin un valor entero. Un valor de 0 es equivalente a especificar la cláusula NOT WAIT. Si el valor está entre 1 y 32767, el gestor de bases de datos esperará ese número de segundos (si no se puede obtener un bloqueo) antes de devolver un error (SQLSTATE 40001 or SQLSTATE 57033).

variable-lengpral
Una variable de tipo INTEGER. El valor debe estar entre -1 y 32767. Si *variable-lengpral* tiene una variable indicadora asociada y su valor especifica un valor nulo, se desestablece el valor CURRENT LOCK TIMEOUT. Es equivalente a especificar la palabra clave NULL.

Notas:

- *Compatibilidades*
  - Para compatibilidad con Informix:
    - Se puede especificar MODE en lugar de TIMEOUT.
    - Se puede especificar TO en lugar del operador igual (=).
    - Se puede especificar SET LOCK WAIT en lugar de SET CURRENT LOCK TIMEOUT WAIT.
    - Se puede especificar SET LOCK NO WAIT en lugar de SET CURRENT LOCK TIMEOUT NOT WAIT.
- Un valor actualizado del registro especial surte efecto inmediatamente después de la ejecución satisfactoria de esta sentencia. Puesto que el valor de registro especial que se debe utilizar durante la ejecución de la sentencia se fija al principio de la ejecución de la sentencia, sólo devolverán un valor actualizado del registro especial CURRENT LOCK TIMEOUT las sentencias que inician la ejecución después de que la sentencia SET LOCK TIMEOUT se haya completado satisfactoriamente.

Ejemplos:

Ejemplo 1: Establezca el valor de tiempo de espera excedido para que espere 30 segundos antes de devolver un error.

```
SET CURRENT LOCK TIMEOUT 30
```

Ejemplo 2: Desestablezca el valor de tiempo de espera excedido de bloqueo para que se utilice el valor del parámetro de configuración de bases de datos *locktimeout* en su lugar.

```
SET CURRENT LOCK TIMEOUT NULL
```

Información relacionada:

- "Registro especial CURRENT LOCK TIMEOUT" en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

La sentencia SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION cambia el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION. No está bajo el control de la transacción.

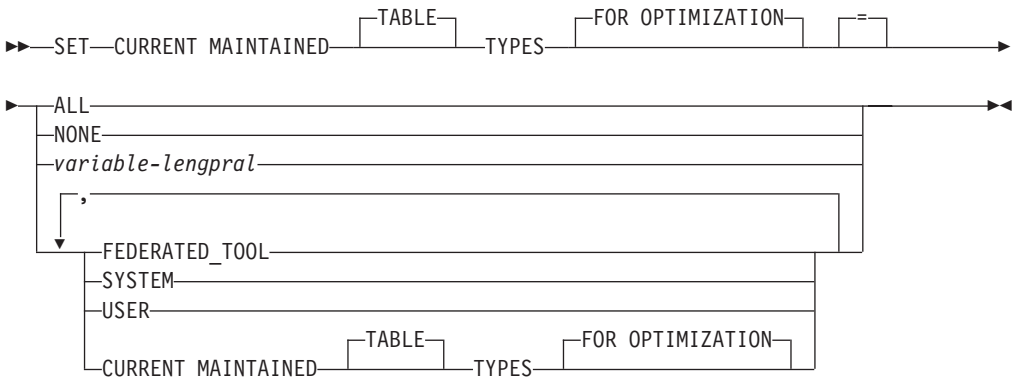
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

ALL
Especifica que todos los tipos posibles de tablas mantenidas que este registro especial controla, ahora y en el futuro, van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

NONE
Especifica que ninguno de los tipos de objetos que este registro especial controla van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

FEDERATED_TOOL
Especifica que las tablas de consultas materializadas de renovación diferida que una herramienta federada mantiene pueden tomarse en consideración para optimizar el proceso de consultas de SQL dinámico, siempre que el valor del registro especial CURRENT QUERY OPTIMIZATION sea 2 o mayor que 5.

SYSTEM
Especifica que las tablas de consultas materializadas con renovación diferida mantenidas por el sistema pueden considerarse para la optimización del proceso de las consultas de SQL dinámico. (Las tablas de consultas materializadas inmediatas siempre están disponibles.)

USER
Especifica que las tablas de consultas materializadas con renovación diferida

mantenidas por el usuario pueden considerarse para la optimización del proceso de las consultas de SQL dinámico.

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
El valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION antes de que se ejecute esta sentencia.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la variable del lenguaje principal no debe exceder de 254 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de *variable-lengpral* deben estar justificados por la izquierda. El contenido de la *variable-lengpral* debe ser una serie que forme una lista de palabras clave separadas por comas y que coincidan con lo que se puede especificar como palabras claves para el registro especial. Estas palabras clave se deben especificar exactamente en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas. El valor deberá rellenarse con blancos por la derecha si su longitud es menor que la de la variable del lenguaje principal.

Notas:

- El valor inicial del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION es SYSTEM.
- El registro especial CURRENT REFRESH AGE debe establecerse en un valor distinto de cero para que los tipos de tablas que se han especificado se consideren cuando se optimice el proceso de las consultas de SQL dinámico.

Ejemplos:

Ejemplo 1: Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION SYSTEM = USER
```

Ejemplo 2: Recupere el valor actual del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION en una variable del lenguaje principal denominada CURMAINTYPES.

```
EXEC SQL VALUES (CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION)
  INTO :CURMAINTYPES
```

Ejemplo 3: Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION de modo que no tenga ningún valor.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = NONE
```

SET CURRENT PACKAGE PATH

La sentencia SET CURRENT PACKAGE PATH asigna un valor al registro especial CURRENT PACKAGE PATH. No está bajo el control de la transacción.

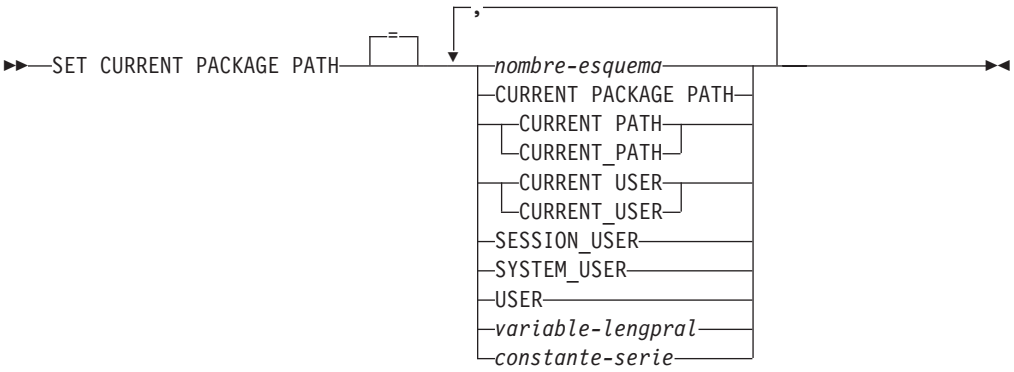
Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

nombre-esquema
Identifica un esquema. El nombre no debe ser un identificador delimitado que esté vacío o que sólo contenga espacios en blanco (SQLSTATE 42815).

CURRENT PACKAGE PATH
El valor del registro especial CURRENT PACKAGE PATH antes de que se ejecute esta sentencia.

CURRENT PATH
El valor del registro especial CURRENT PATH.

CURRENT USER
El valor del registro especial CURRENT USER.

SESSION_USER
El valor del registro especial SESSION_USER.

SYSTEM_USER
El valor del registro especial SYSTEM_USER.

USER
El valor del registro especial USER.

variable-lengpral
Contiene uno o varios nombres de esquema, separados mediante comas. La variable del lenguaje principal debe:

- Ser una variable de serie de caracteres (CHAR o VARCHAR). La longitud real del contenido de la variable del lenguaje principal no debe exceder de la longitud del registro especial CURRENT PACKAGE PATH.
- No ser un valor nulo. Si se proporciona una variable indicadora, su valor no debe indicar un valor nulo.
- Contener una serie vacía o en blanco o uno o varios nombres de esquema separados por comas.
- Rellenarse por la derecha con espacios en blanco si la longitud real de la variable del lenguaje principal es mayor que el contenido.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío ni que sólo contenga espacios en blanco.

constante-serie
Especifica una constante de serie de caracteres que contiene cero, uno o varios nombres de esquema que están separados por comas. La constante de serie debe:

- Tener una longitud que no exceda de la longitud máxima del registro especial CURRENT PACKAGE PATH.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío o que contenga sólo espacios en blanco.

Normas:

- Si el mismo esquema aparece más de una vez en la lista, se utiliza la primera ocurrencia del esquema (SQLSTATE 01625).
- El número de esquemas que se pueden especificar está limitado por la longitud total del registro especial CURRENT PACKAGE PATH. La serie de registro especial se crea tomando el nombre del esquema especificado y eliminando los blancos de cola, delimitando el nombre con comillas dobles y separando los nombres de esquema con comas. La longitud de la lista resultante no puede exceder de la longitud máxima del registro especial (SQLSTATE 0E000).
- Un nombre de esquema que no se ajuste a las normas para un identificador normal (por ejemplo, un nombre de esquema que contenga caracteres en minúsculas o caracteres que no se pueden especificar en un identificador normal), se debe especificar como un nombre de esquema delimitado y no se debe especificar en una variable de lenguaje principal ni constante de serie.
- Para indicar que el valor actual de un registro especial (especificado como una sola palabra clave) se debe utilizar en la vía de acceso del paquete, especifique el nombre del registro especial como palabra clave. Si el nombre del registro especial se especifica como identificador delimitado en su lugar (por ejemplo, "USER"), se interpreta como un nombre de esquema de ese valor ('USER').
- Las normas siguientes se utilizan para determinar si un valor especificado en una sentencia SET CURRENT PACKAGE PATH es una variable o un nombre de esquema:
  - Si el *nombre* es igual que un parámetro o variable de SQL en el procedimiento de SQL, *nombre* se interpreta como parámetro o variable de SQL y el valor de *nombre* se asigna a la vía de acceso del paquete.

– Si *nombre* no es igual que un parámetro o una variable de SQL del procedimiento de SQL, *nombre* se interpreta como un nombre de esquema y el valor de *nombre* se asigna a la vía de acceso del paquete.

Notas:

- ***Consideraciones de la transacción:*** La sentencia SET CURRENT PACKAGE PATH no es una operación que se pueda confirmar. ROLLBACK no tiene ningún efecto en el registro especial CURRENT PACKAGE PATH.
- ***Comprobación de existencia de esquemas:*** No se hace ninguna validación de que existan los esquemas especificados en el momento en que se establece el registro especial CURRENT PACKAGE PATH. Por ejemplo, un esquema que no está bien escrito no se detecta, lo que podría afectar a la manera en que funciona el SQL subsiguiente. En tiempo de ejecución de paquetes, se comprueba la autorización para un paquete coincidente y si la comprobación de autorización falla, se devuelve un error (SQLSTATE 42501).
- ***Contenido de la variable de lenguaje principal o constante de serie:*** El contenido de una variable de lenguaje principal o una constante de serie se interpreta como una lista de nombres de esquema. Si se especifican múltiples nombres de esquema, deben estar separados por comas. Cada nombre de esquema de la lista debe ajustarse a las normas para formar un identificador normal, o especificarse como identificador delimitado. El contenido de la variable del lenguaje principal o constante de serie no se convierte a mayúsculas.
- ***Restricciones específicas para el SQL incorporado para aplicaciones COBOL:*** Pueden aparecer un máximo de diez valores literales (no variables de lenguaje principal) en el lado derecho de una sentencia SET CURRENT PACKAGE PATH. Estos valores tienen una longitud máxima de 130 (no delimitado) o 128 (delimitado).

Ejemplos:

Ejemplo 1: Establezca el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: MYPKGS, 'ABC E', SYSIBM

```
SET CURRENT PACKAGE PATH = MYPKGS, 'ABC E', SYSIBM
```

La sentencia siguiente establece una variable del lenguaje principal en el valor de la lista resultante:

```
SET :hvpklist = CURRENT PACKAGE PATH
```

El valor de la variable del lenguaje principal es: "MYPKGS", "ABC E", "SYSIBM".

Ejemplo 2: Establezca el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: "SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH :hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH4","SCH5".

Ejemplo 3: Establezca el registro especial CURRENT PACKAGE PATH en la lista siguiente de esquemas: "SCH1","SCH#2","SCH3","SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH = SCH1,'SCH#2',"SCH3",:hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH1","SCH#2","SCH3","SCH4","SCH5".

Ejemplo 4: Borre el registro especial CURRENT PACKAGE PATH.

```
SET CURRENT PACKAGE PATH = ''
```

Ejemplo 5: Añada temporalmente el esquema ″SCH_PROD″ (contenido en la variable del lenguaje principal :prodschema) y el esquema ″SCH_PROD2″ (contenido en la variable del lenguaje principal :prod2schema) al final del registro especial CURRENT PACKAGE PATH para la ejecución del procedimiento SUMMARIZE. Después, vuelva a cambiar el registro especial CURRENT PACKAGE PATH a este valor anterior.

```
SET :oldCPP = CURRENT PACKAGE PATH

SET CURRENT PACKAGE PATH = CURRENT PACKAGE PATH,:prodschema,:prod2schema

CALL SUMMARIZE(:V1,:V2)

SET CURRENT PACKAGE PATH = :oldCPP
```

Ejemplo 6: Establezca el registro especial CURRENT PACKAGE PATH en una lista de nombres de esquema delimitados: ″MY.SCHEMA″ (punto intercalado), ″OLD SCHEMA″ (blanco intercalado). Utilice una sola variable del lenguaje principal que contenga ambos identificadores delimitados:

```
hv = '"MY.SCHEMA", "OLD SCHEMA"'

SET CURRENT PACKAGE PATH = :hv
```

o utilice una sola constante de serie que contenga ambos identificadores delimitados:

```
SET CURRENT PACKAGE PATH = '"MY.SCHEMA", "OLD SCHEMA"'
```

o utilice una lista de esquemas delimitados:

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

Información relacionada:

- “Registro especial CURRENT PACKAGE PATH” en la publicación *Consulta de SQL, Volumen 1*

SET CURRENT PACKAGESET

La sentencia SET CURRENT PACKAGESET establece el nombre de esquema (identificador de colección) que se utilizará para seleccionar el paquete que se debe utilizar para las sentencias de SQL posteriores. La sentencia no está bajo el control de la transacción.

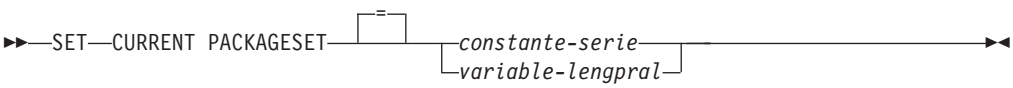
Invocación:

Esta sentencia sólo se puede incluir en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica. Esta sentencia no está soportada en REXX.

Autorización:

No se necesita.

Sintaxis:



Descripción:

constante-serie
Constante de serie de caracteres. Si el valor excede de 30 bytes, sólo se utilizarán los 30 primeros bytes.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. No puede establecerse en nulo. Si el valor excede de 30 bytes, sólo se utilizarán los 30 primeros bytes.

Notas:

- Esta sentencia permite que una aplicación especifique el nombre de esquema utilizado al seleccionar un paquete para una sentencia de SQL ejecutable. La sentencia se procesa en el cliente y no fluye al servidor de aplicaciones.
- Se puede utilizar la opción de enlace lógico COLLECTION para crear un paquete con un nombre de esquema especificado.
- A diferencia de DB2 UDB para OS/390 y z/OS, la sentencia SET CURRENT PACKAGESET se implementa sin soporte para un registro especial denominado CURRENT PACKAGESET.

Ejemplo:

Supongamos que el ID de usuario PRODUSA ha compilado previamente una aplicación denominada TRYIT, donde 'PRODUSA' es el nombre de esquema por omisión en el archivo de enlace. Después, la aplicación se enlaza dos veces con diferentes opciones de enlace lógico. Se utilizan los siguientes mandatos del procesador de línea de mandatos:

```
DB2 CONNECT TO SAMPLE USER PRODUSA
DB2 BIND TRYIT.BND DATETIME USA
DB2 CONNECT TO SAMPLE USER PRODEUR
DB2 BIND TRYIT.BND DATETIME EUR COLLECTION 'PRODEUR'
```

Esto crea dos paquetes llamados TRYIT. El primer mandato de enlace lógico ha creado el paquete en el esquema llamado 'PRODUSA'. El segundo mandato de enlace lógico ha creado el paquete en el esquema llamado 'PRODEUR' basándose en la opción COLLECTION.

Suponga que la aplicación TRYIT contiene las sentencias siguientes:

```
EXEC SQL CONNECT TO SAMPLE;
   .
   .
   EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 1
   .
   .
   EXEC SQL SET CURRENT PACKAGESET 'PRODEUR';                            2
   .
   .
   EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 3
```

1 Esta sentencia se ejecutará utilizando el paquete PRODUSA.TRYIT porque es el paquete por omisión para la aplicación. Por lo tanto, la fecha se devuelve en formato USA.

2 Esta sentencia establece el nombre de esquema en 'PRODEUR' para la selección del paquete.

3 Esta sentencia se ejecutará utilizando el paquete PRODEUR.TRYIT como resultado de la sentencia SET CURRENT PACKAGESET. Por lo tanto, la fecha se devuelve en formato EUR.

SET CURRENT QUERY OPTIMIZATION

La sentencia SET CURRENT QUERY OPTIMIZATION asigna un valor al registro especial CURRENT QUERY OPTIMIZATION. El valor especifica la clase actual de técnicas de optimización habilitadas al preparar las sentencias de SQL dinámico. No está bajo el control de la transacción.

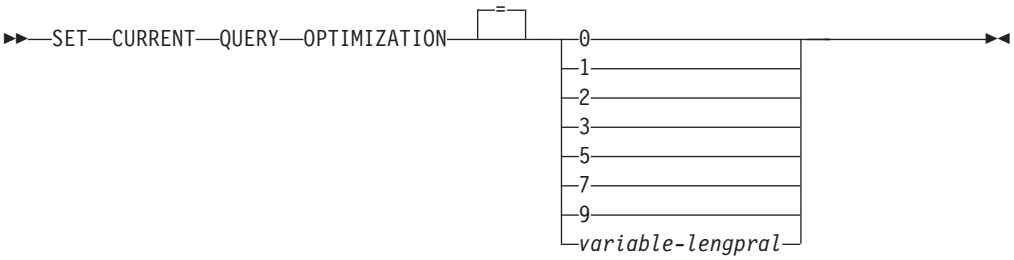
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

clase-optimización

La *clase-optimización* se puede especificar como una constante de enteros o como el nombre de una variable del lenguaje principal que contendrá el valor adecuado en tiempo de ejecución. A continuación se facilita una visión general de las clases.

0 Especifica que se efectúa una optimización mínima para generar un plan de acceso. Esta clase es la más adecuada para el acceso SQL dinámico simple para tablas bien indexadas.

1 Especifica que se efectúa una optimización más o menos comparable a DB2 Versión 1 para generar un plan de acceso.

2 Especifica un nivel de optimización superior al de DB2 Versión 1, pero con un coste de optimización significativamente menor que los niveles 3 y superiores, especialmente para consultas muy complejas.

3 Especifica que se efectúa una optimización moderada para generar un plan de acceso.

5 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Para consultas SQL dinámico complejas, se utilizan las normas heurísticas para limitar el tiempo empleado en seleccionar un plan de acceso. Cuando sea posible, las consultas utilizarán tablas de consultas materializadas en lugar de la tablas base subyacentes.

7 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Similar al 5 pero sin las normas heurísticas.

9 Especifica que se efectúa una optimización máxima para generar un plan de acceso. Puede expandir mucho el número de planes de acceso posibles que se evalúan. Esta clase debe utilizarse para determinar si se puede generar un plan de acceso mejor para las consultas muy complejas y de muy larga ejecución que utilizan tablas grandes. Las medidas de explicación y de rendimiento se pueden utilizar para verificar que se haya generado el plan mejor.

variable-lengpral

El tipo de datos es INTEGER. El valor debe estar en el rango de 0 a 9 (SQLSTATE 42815) pero debe ser 0, 1, 2,3, 5, 7 ó 9 (SQLSTATE 01608). Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Notas:

- Cuando el registro CURRENT QUERY OPTIMIZATION se establece en un valor en particular, se habilita un conjunto de normas para volver a escribir la consulta y ciertas variables de optimización toman valores determinados. Esta clase de técnicas de optimización se utiliza después durante la preparación de sentencias de SQL.
- En general, el cambio de la clase de optimización afecta al tiempo de ejecución de la aplicación, al tiempo de compilación y a los recursos necesarios. La mayoría de sentencias se optimizarán de manera adecuada utilizando la clase de optimización de consulta por omisión. Las clases de optimización de consulta inferiores, especialmente las clases 1 y 2,pueden ser adecuadas para las sentencias de SQL dinámico para las cuales los recursos que consume una operación *PREPARE* dinámica son una parte significativa de los necesarios para ejecutar la consulta. Las clases de optimización superiores sólo deben elegirse después de tener en cuenta los recursos adicionales que pueden consumirse y verificar que se haya generado un plan de acceso mejor.
- Las clases de optimización de consulta deben estar en el rango de 0 a 9. Las clases fuera de este rango devolverán un error (SQLSTATE 42815). Las clases no soportadas dentro de este rango devolverán un aviso (SQLSTATE 01608) y se sustituirán por la siguiente clase de optimización de consulta inferior. Por ejemplo, una clase 6 de optimización de consulta se sustituirá por 5.
- Las sentencias preparadas dinámicamente utilizan la clase de optimización que se ha establecido por la sentencia SET CURRENT QUERY OPTIMIZATION más reciente que se ha ejecutado. En los casos en los que todavía no se ha ejecutado una sentencia SET CURRENT QUERY OPTIMIZATION, la clase de optimización de la consulta la determina el valor del parámetro de configuración de la base de datos, dft_queryopt.
- Las sentencias enlazadas estáticamente no utilizan el registro especial CURRENT QUERY OPTIMIZATION; por lo tanto, esta sentencia no tiene ningún efecto sobre ellas. La opción QUERYOPT se utiliza durante el preproceso o enlace lógico para especificar la clase de optimización deseada para las sentencias enlazadas estáticamente. Si no se especifica QUERYOPT, se utilizará el valor por omisión que especifica el parámetro de configuración de la base de datos, dft_queryopt.

- No se retrotrae el resultado de la ejecución de la sentencia SET CURRENT QUERY OPTIMIZATION si se retrotrae la unidad de trabajo en la que se ejecuta.

Ejemplos:

Ejemplo 1: Este ejemplo muestra cómo se puede seleccionar el grado de optimización más alto.

```
SET CURRENT QUERY OPTIMIZATION 9
```

Ejemplo 2: El ejemplo siguiente muestra cómo se puede utilizar el registro especial CURRENT QUERY OPTIMIZATION en una consulta.

Mediante la utilización de la vista de catálogo SYSCAT.PACKAGES, busque todos los planes que se han enlazado con el mismo valor que el valor actual del registro especial CURRENT QUERY OPTIMIZATION.

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
    WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

SET CURRENT REFRESH AGE

La sentencia SET CURRENT REFRESH AGE cambia el valor del registro especial CURRENT REFRESH AGE. No está bajo el control de la transacción.

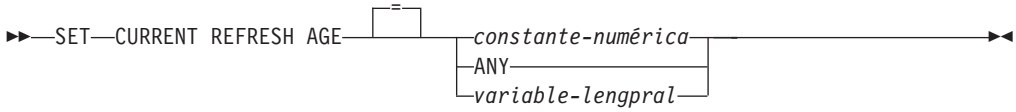
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

constante-numérica
Un valor DECIMAL(20,6) que representa una duración de indicación de la hora. El valor debe ser 0 o 99 999 999 999 999 (la parte de microsegundos del valor se ignora y, por lo tanto, puede ser cualquier valor).

ANY
Es una abreviación de 99 999 999 999 999.

variable-lengpral
Una variable de tipo DECIMAL(20,6) u otro tipo que se pueda asignar a DECIMAL(20,6). No puede establecerse en nulo. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815). El valor de la *variable-lengpral* debe ser 0 ó 99 999 999 999 999.

Notas:

- El valor inicial del registro especial CURRENT REFRESH AGE es cero.
- El valor de CURRENT REFRESH AGE se sustituye por el valor especificado. El valor debe ser 0 ó 99 999 999 999 999. El valor 99 999 999 999 999 representa 9999 años, 99 meses, 99 días, 99 horas, 99 minutos y 99 segundos.

  Si el valor de CURRENT REFRESH AGE es 0, las tablas de consultas materializadas afectadas por este registro especial no se utilizarán para optimizar el proceso de una consulta. Si el valor de CURRENT REFRESH AGE es 99 999 999 999 999, las tablas de consultas materializadas afectadas por este registro especial pueden utilizarse para optimizar el proceso de una consulta, pero sólo si el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION las incluye y el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o un valor mayor o igual a 5. Las tablas de consultas materializadas afectadas por este registro especial son REFRESH DEFERRED MAINTAINED BY USER y REFRESH DEFERRED MAINTAINED BY SYSTEM.

Las tablas de consultas materializadas REFRESH IMMEDIATE MAINTAINED BY SYSTEM siempre pueden utilizarse para optimizar el proceso de una consulta si el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o un valor mayor o igual a 5.

Las tablas de consultas materializadas REFRESH DEFERRED MAINTAINED BY FEDERATED_TOOL se utilizan para la optimización si el registro especial CURRENT QUERY OPTIMIZATION se establece en 2 o en un valor mayor o igual a 5 y el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION se establece en ALL o incluye FEDERATED_TOOL.

- Debe tener cuidado cuando establezca el registro especial CURRENT REFRESH AGE en un valor que no sea cero. Un tipo de tabla especificado por el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION puede no representar los valores de la tabla base subyacente. Si se utiliza un tipo de tabla así para optimizar el proceso de una consulta, el resultado de la consulta podría *no* representar de forma precisa los datos de la tabla subyacente. Puede ser razonable utilizarlo si sabe que los datos subyacentes no han cambiado, o si puede aceptar un grado de error en los resultados, basándose en su conocimiento de los datos almacenados en antememoria.
- El valor CURRENT REFRESH AGE actual de 99 999 999 999 999 no se puede utilizar en operaciones aritméticas de indicación de la hora, porque el resultado estaría fuera del rango válido de fechas (SQLSTATE 22008).

Ejemplos:

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT REFRESH AGE.

```
SET CURRENT REFRESH AGE ANY
```

Ejemplo 2: El siguiente ejemplo recupera el valor del registro especial CURRENT REFRESH AGE de una variable del lenguaje principal denominada CURMAXAGE. El valor, establecido en el ejemplo anterior, es 99999999999999.000000.

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO  :CURMAXAGE;
```

SET ENCRYPTION PASSWORD

La sentencia SET ENCRYPTION PASSWORD establece las contraseñas que las funciones ENCRYPT, DECRYPT_BIN y DECRYPT_CHAR utilizarán. La contraseña no está vinculada a la autentificación de DB2 y se utiliza solamente para el cifrado y descifrado de datos.

La sentencia no está bajo el control de la transacción.

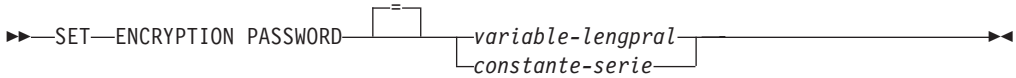
Invocación:

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:
La sentencia ENCRYPTION PASSWORD pueden utilizarla las funciones incorporadas ENCRYPT, DECRYPT_BIN y DECRYPT_CHAR para el cifrado basado en contraseñas. La longitud debe ser de 6 a 127 bytes. Todos los caracteres deben especificarse en las mayúsculas y minúsculas exactas que se desean porque no se realiza ninguna conversión automática a caracteres en mayúsculas.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. La longitud de *variable-lengpral* debe tener una longitud de 6 a 127 bytes (SQLSTATE 428FC). No puede establecerse en nulo. Todos los caracteres se especifican en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas.

constante-serie
Constante de serie de caracteres. La longitud debe ser de 6 a 127 bytes (SQLSTATE 428FC).

Notas:

- El valor inicial de ENCRYPTION PASSWORD es la serie vacía (`''`).
- La *variable-lengpral* o *constante-serie* se transmite al servidor de bases de datos utilizando los mecanismos normales de DB2.

Ejemplos:

Ejemplo 1: La sentencia siguiente establece la contraseña de cifrado (ENCRYPTION PASSWORD).

```
SET ENCRYPTION PASSWORD = 'Gre89Ea'
```

Información relacionada:

- "Funciones escalares DECRYPT_BIN y DECRYPT_CHAR" en la publicación *Consulta de SQL, Volumen 1*
- "Función escalar ENCRYPT" en la publicación *Consulta de SQL, Volumen 1*

SET EVENT MONITOR STATE

La sentencia SET EVENT MONITOR STATE activa o desactiva el supervisor de sucesos. El estado actual de un supervisor de sucesos (activo o no activo) se determina utilizando la función incorporada EVENT_MON_STATE. La sentencia SET EVENT MONITOR STATE no está bajo el control de la transacción.

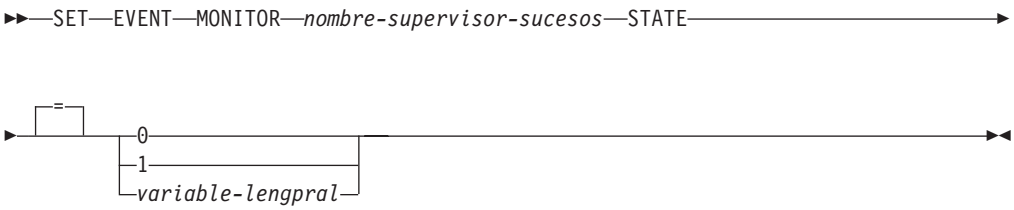
Invocación:

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

Autorización:

El ID de autorización de la sentencia debe tener autorización SYSADM o DBADM (SQLSTATE 42815).

Sintaxis:



Descripción:

nombre-supervisor-sucesos
Identifica el supervisor de sucesos que se debe activar o desactivar. El nombre debe identificar un supervisor de sucesos que exista en el catálogo (SQLSTATE 42704).

estado-nuevo
El *estado-nuevo* puede especificarse como una constante de enteros o como el nombre de una variable del lenguaje principal que contendrá el valor adecuado en tiempo de ejecución. Se puede especificar lo siguiente:

0 Indica que el supervisor de sucesos especificado debe desactivarse.

1 Indica que el supervisor de sucesos especificado debe activarse. El supervisor de sucesos no debe estar activo todavía; de lo contrario se emite un aviso (SQLSTATE 01598).

variable-lengpral
El tipo de datos es INTEGER. El valor especificado debe ser 0 ó 1 (SQLSTATE 42815). Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Normas:

- Aunque puede definirse un número no limitado de supervisores de sucesos, existe un límite de 32 supervisores de sucesos que pueden estar activos simultáneamente (SQLSTATE 54030).

- Para activar un supervisor de sucesos, la transacción en la que se ha creado el supervisor de sucesos debe haberse confirmado (SQLSTATE 55033). Esta norma impide (en una unidad de trabajo) la creación de un supervisor de sucesos, la activación del supervisor y después la retrotracción de la transacción.
- Si el número o tamaño de los archivos del supervisor de sucesos excede de los valores especificados para MAXFILES o MAXFILESIZE en la sentencia CREATE EVENT MONITOR, se genera un error (SQLSTATE 54031).
- Si la vía de acceso de destino del supervisor de sucesos (que se ha especificado en la sentencia CREATE EVENT MONITOR) ya se utiliza en otro supervisor de sucesos, se genera un error (SQLSTATE 51026).

Notas:

- La activación de un supervisor de sucesos realiza una restauración de cualquier contador asociado al mismo.
- Cuando se inicia un supervisor de sucesos WRITE TO TABLE utilizando SET EVENT MONITOR STATE, éste actualiza la columna EVMON_ACTIVATES de la vista de catálogo SYSCAT.EVENTMONITORS. Si la unidad de trabajo en la que se ha realizado la operación de conjuntos se retrotrae por cualquier motivo, la actualización del catálogo se perderá. Cuando se reinicie el supervisor de sucesos, volverá a utilizar el valor de EVMON_ACTIVATES que se haya retrotraído.

Ejemplo:

El ejemplo siguiente activa un supervisor de sucesos llamado SMITHPAY.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

SET INTEGRITY

La sentencia SET INTEGRITY se utiliza para:

- Desactivar la comprobación de la integridad para una o más tablas. Esto incluye la comprobación de las restricciones de comprobación y de referencia, la comprobación de la integridad de DATALINK y la generación de valores para las columnas generadas. Si la tabla es una tabla de consultas materializadas definida con REFRESH IMMEDIATE o una tabla de etapas con el atributo PROPAGATE IMMEDIATE, se desactiva la renovación inmediata de los datos. Esto establece la tabla en *estado pendiente de comprobación*, donde sólo está permitido el acceso limitado por parte de un conjunto restringido de sentencias y mandatos. Se continúan comprobando las restricciones de unicidad y de clave primaria.
- Volver a activar la comprobación de la integridad y realizar todas las comprobaciones diferidas que corresponden a una o más tablas. Si la tabla es una tabla de consultas materializas mantenida por el sistema o una tabla de etapas, los datos se renuevan en función de las necesidades. Si la tabla de consultas materializadas se ha definido con el atributo REFRESH IMMEDIATE o la tabla de etapas se ha definido con el atributo PROPAGATE IMMEDIATE, se activa la renovación inmediata de los datos. Las tablas de claves externas descendentes, las tablas de consultas materializadas inmediatas descendentes y las tablas de etapas inmediatas descendentes correspondientes a las tablas que se han establecido en estado pendiente de comprobación mediante la utilización de la opción CASCADE DEFERRED, se establecen (si es necesario) en estado sin acceso pendiente de comprobación.
- Activar la comprobación de integridad para una o más tablas sin realizar primero ninguna comprobación de integridad diferida. Si la tabla es una tabla de consultas materializadas definida con el atributo REFRESH IMMEDIATE o una tabla de etapas definida con el atributo PROPAGATE IMMEDIATE, se activa la renovación inmediata de los datos. Las tablas de claves externas descendentes, las tablas de consultas materializadas inmediatas descendentes y las tablas de etapas inmediatas descendentes correspondientes a las tablas que se han establecido en estado pendiente de comprobación mediante la utilización de la opción CASCADE DEFERRED, se establecen (si es necesario) en estado sin acceso pendiente de comprobación.
- Establecer la tabla en estado pendiente de comprobación si la tabla ya está en estado pendiente de reconciliación de DataLink (DRP) o en estado de reconciliación de DataLink no posible (DRNP). Si una tabla no se encuentra en ninguno de esos dos estados, establezca la tabla incondicionalmente en el estado DRP y el estado de comprobación pendiente.
- Establecer una o más tablas que están en estado sin movimiento de datos nuevamente a la modalidad de acceso completo.
- Podar el contenido de una o más tablas de etapas.

Cuando la sentencia se utiliza para comprobar la integridad de una tabla tras haberse cargado ésta, el sistema puede realizar un proceso incremental en la tabla y comprobar sólo la parte añadida para verificar si existen violaciones de restricciones. Si la tabla sujeto es una tabla de consultas materializadas o una tabla de etapas y se realizan operaciones de carga en sus tablas subyacentes, el sistema puede realizar una renovación incremental en la tabla de consultas materializadas o bien realizar una propagación incremental en la tabla de etapas utilizando sólo las partes añadidas de sus tablas subyacentes. Sin embargo, existen algunas situaciones en las que el sistema no podrá realizar tales optimizaciones y, en lugar de ello, realizará un proceso completo para garantizar la integridad de los datos. El

proceso completo se realiza comprobando toda la tabla para verificar si existen violaciones de las restricciones, volviendo a calcular la definición de una tabla de consultas materializadas o marcando una tabla de etapas como no coherente. Esta última acción implica la necesidad de realizar una renovación completa de su tabla de consultas materializadas asociada. Existen también una situación en la que puede que el usuario desee solicitar explícitamente un proceso incremental especificando la opción INCREMENTAL.

La sentencia SET INTEGRITY está bajo control de transacciones.

Invocación:

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

Autorización:

Los privilegios necesarios para ejecutar SET INTEGRITY dependen de la utilización de la sentencia, tal como se resalta abajo:

- Desactivar la comprobación de la integridad.

  El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:
  - Privilegio CONTROL para:
    - Las tablas especificadas y
    - Las tablas de claves foráneas descendentes donde la sentencia desactivará la comprobación de la integridad y
    - Las tablas de consultas materializadas inmediatas descendentes donde la sentencia desactivará la comprobación de la integridad y
    - Las tablas de etapas inmediatas descendentes donde la sentencia desactivará la comprobación de la integridad.
  - Autorización SYSADM o DBADM
  - Autorización LOAD
- Activar la comprobación de la integridad y llevar a cabo la comprobación.

  El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:
  - Autorización SYSADM o DBADM
  - Privilegio CONTROL para las tablas que están comprobándose *y*, si están enviándose excepciones a una o más tablas, privilegio INSERT para las tablas de excepciones. Privilegio CONTROL para todas las tablas de claves foráneas descendentes, las tablas de consultas materializadas inmediatas descendentes y las tablas de etapas inmediatas descendentes que la sentencia establecerá implícitamente en estado pendiente de comprobación.
  - Autorización LOAD y, si están enviándose excepciones a una o más tablas:
    - Privilegio SELECT y DELETE para cada tabla que se comprueba; y
    - Privilegio INSERT para las tablas de excepciones.
- Activar las restricciones de integridad sin antes llevar a cabo la comprobación.

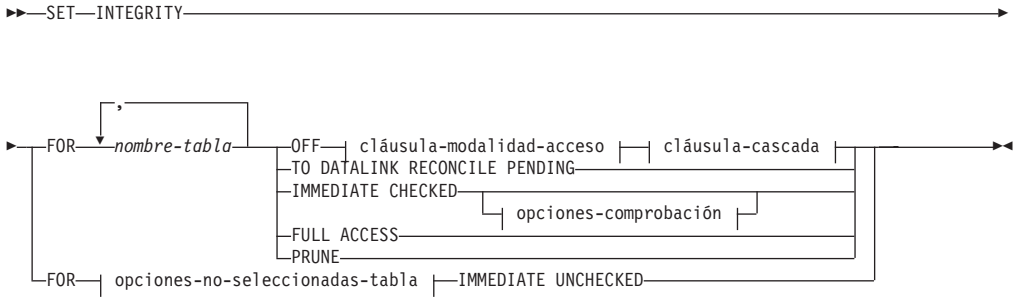
  El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes:
  - Autorización SYSADM o DBADM

  - Privilegio CONTROL para las tablas que están comprobándose. Privilegio CONTROL para cada tabla de claves foráneas descendente, tabla de consultas materializadas inmediata descendente y tabla de etapas inmediata descendente que la sentencia establecerá implícitamente en estado pendiente de comprobación.
  - Autorización LOAD
- Establecer la tabla que está en modalidad sin movimiento de datos en modalidad de acceso completo.

  El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes:
  - Autorización SYSADM o DBADM
  - Privilegio CONTROL para las tablas que cambian de la modalidad sin movimiento de datos a la modalidad de acceso completo.
  - Autorización LOAD
- Podar una tabla de etapas.

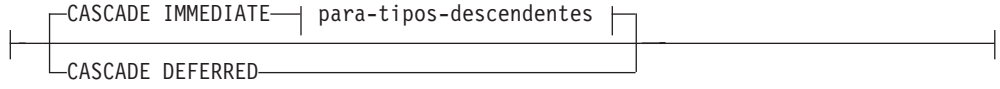
  El ID de autorización de la sentencia debe tener como mínimo uno de los siguientes:
  - Autorización SYSADM o DBADM
  - Privilegio CONTROL para la tabla que está podándose.

Sintaxis:

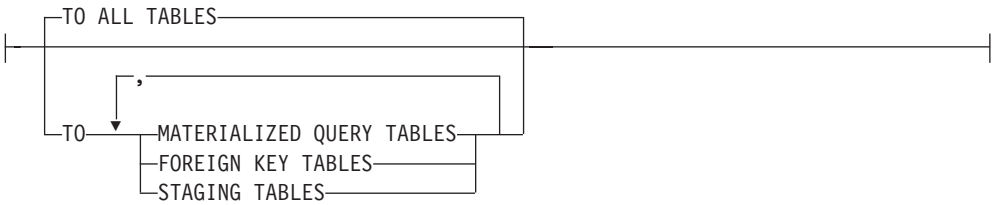


cláusula-modalidad-acceso:

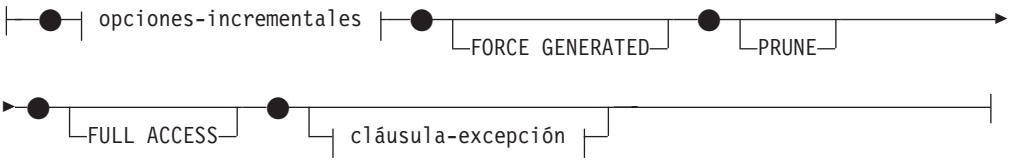
cláusula-cascada:



para-tipos-descendentes:

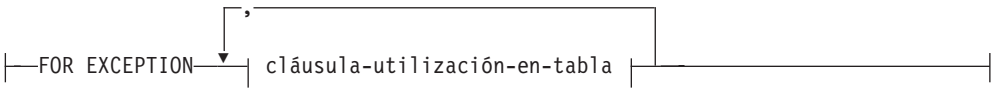


opciones-comprobación:



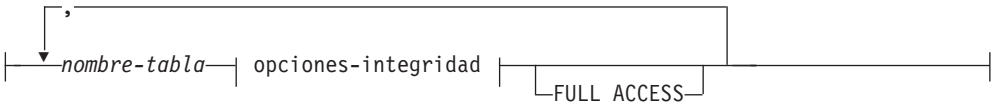
opciones-incrementales:

cláusula-excepción:

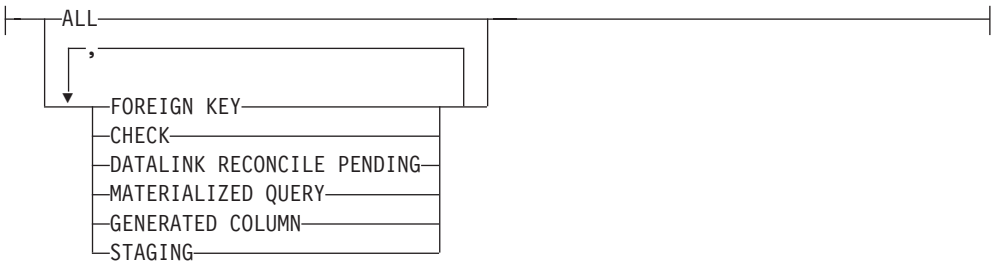


cláusula-utilización-en-tabla:

opciones-no-seleccionadas-tabla:



opciones-integridad:



Descripción:

FOR *nombre-tabla*
Identifica una o más tablas para el proceso de la integridad. Debe ser una tabla descrita en el catálogo y no debe ser una vista, una tabla del catálogo ni una tabla con tipo.

OFF
Especifica que las tablas van a tener desactivadas las restricciones de clave foránea, las restricciones de comprobación y la generación de columnas y que, por lo tanto, se establecerán en estado pendiente de comprobación. Si es una tabla de consultas materializadas o una tabla de etapas, la renovación inmediata se desactiva (si se aplica) y la tabla de consultas materializadas o la tabla de etapas se establece en estado pendiente de comprobación.

Tenga en cuenta que es posible que una tabla ya se encuentre en estado pendiente de comprobación con sólo un tipo de comprobación de la integridad desactivado; en una situación así, el otro tipo de comprobación de la integridad también se desactivará.

En una tabla que está en estado pendiente de comprobación sólo está permitida una actividad muy limitada.

cláusula-modalidad-acceso
Especifica hasta qué punto está preparada la tabla mientras se encuentra en estado pendiente de comprobación.

NO ACCESS
Especifica que la tabla va a establecerse en estado sin acceso pendiente de comprobación, donde no está permitido el acceso de lectura o de grabación a la tabla.

READ ACCESS
Especifica que la tabla va a establecerse en estado de lectura pendiente de comprobación, donde está permitido el acceso de lectura a la parte no añadida de la tabla. Esta opción no está permitida en una tabla que está en estado sin acceso pendiente de comprobación (SQLSTATE 428FH).

Si no se especifica la *cláusula-modalidad-acceso*, la tabla se establece en estado sin acceso pendiente de comprobación.

cláusula-cascada
Especifica si el estado pendiente de comprobación de la tabla a la que se hace referencia en la sentencia SET INTEGRITY va a aplicarse en cascada inmediatamente a todas las tablas de claves foráneas descendentes, a todas las tablas de consultas materializadas inmediatas descendentes y a todas las tablas de etapas inmediatas descendentes.

CASCADE IMMEDIATE
Especifica que el estado pendiente de comprobación correspondiente a las restricciones de clave foránea va a extenderse inmediatamente a todas las tablas de claves foráneas descendentes. Si la tabla tiene tablas de consultas materializadas inmediatas descendentes o tablas de etapas inmediatas descendentes, el estado pendiente de comprobación se extiende inmediatamente a las tablas de consultas materializadas y a las tablas de etapas.

Cuando posteriormente se comprueba la tabla para verificar si existen violaciones de integridad y se elimina su estado pendiente de comprobación, todas las tablas descendentes que estén en estado de lectura pendiente de comprobación se establecerán en estado sin acceso pendiente de comprobación.

para-tipos-descendentes

TO ALL TABLES
Especifica que el estado pendiente de comprobación va a aplicarse en cascada inmediatamente a todas las tablas descendentes de las tablas de la lista de invocación. Las tablas descendentes incluyen todas las tablas de claves foráneas descendentes, las tablas de etapas inmediatas y las tablas de consultas materializadas inmediatas que son descendentes de las tablas de la lista de invocación o que son descendentes de las tablas de claves foráneas descendentes.

La especificación de TO ALL TABLES equivale a especificar TO FOREIGN KEY TABLES, TO MATERIALIZED QUERY TABLES *y* TO STAGING TABLES, todas ellas en la misma sentencia.

TO MATERIALIZED QUERY TABLES
Si sólo se especifica TO MATERIALIZED QUERY TABLES, el estado pendiente de comprobación se aplicarán en cascada inmediatamente sólo a las tablas de consultas materializadas inmediatas descendentes. Posteriormente, las demás tablas descendentes podrían establecerse en estado pendiente de comprobación, si es necesario, al eliminarse el estado pendiente de comprobación de la tabla. Si se especifica TO FOREIGN KEY TABLES y también TO MATERIALIZED QUERY TABLES, el estado pendiente de comprobación se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendentes, a todas las tablas de consultas materializadas inmediatas descendentes de las tablas de la lista de invocación y a todas las tablas de consultas materializadas inmediatas que son descendentes de las tablas de claves foráneas descendentes.

TO FOREIGN KEY TABLES
Especifica que el estado pendiente de comprobación se aplicará en cascada inmediatamente a las tablas de claves foráneas descendentes. Posteriormente, las demás tablas descendentes podrían establecerse en estado pendiente de comprobación, si es necesario, al eliminarse el estado pendiente de comprobación de la tabla.

TO STAGING TABLES
Especifica que el estado pendiente de comprobación se aplicará en cascada inmediatamente a las tablas de etapas descendentes. Posteriormente, las demás tablas descendentes podrían establecerse en estado pendiente de comprobación, si es necesario, al eliminarse el estado pendiente de comprobación de la tabla. Si se especifica TO FOREIGN KEY TABLES y también TO STAGING TABLES, el estado pendiente de comprobación se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendentes, a todas las tablas de etapas inmediatas descendentes de las tablas de la lista de invocación y a todas las tablas de etapas inmediatas que son descendentes de las tablas de claves foráneas descendentes.

CASCADE DEFERRED
Especifica que sólo las tablas de la lista de invocación van a establecerse en estado pendiente de comprobación. Los estados de las tablas de claves foráneas descendentes, tablas de consultas materializas inmediatas descendentes y tablas de etapas inmediatas descendentes no cambiarán. Posteriormente, las tablas de claves foráneas descendentes pueden establecerse implícitamente en estado sin acceso pendiente de comprobación cuando se comprueben sus tablas padre para verificar si existen violaciones de las restricciones (utilizando la opción IMMEDIATE

CHECKED de la sentencia SET INTEGRITY). Las tablas de consultas materializadas inmediatas descendentes y las tablas de etapas inmediatas descendentes pueden establecerse implícitamente en estado sin acceso pendiente de comprobación al comprobarse una de sus tablas subyacentes para verificar si existen violaciones de integridad. Se emitirá un mensaje de aviso (SQLSTATE 01586) para indicar que las tablas descendentes se han establecido en estado pendiente de comprobación.

Si no se especifica la *cláusula-cascada*, el estado pendiente de comprobación se aplicará en cascada inmediatamente a las tablas dependientes y descendentes.

TO DATALINK RECONCILE PENDING
Especifica que las tablas van a tener desactivada la comprobación de integridad de DATALINK y que las tablas van a establecerse en estado sin acceso pendiente de comprobación. Si la tabla ya se encuentra en el estado de reconciliación de DataLink no posible (DRNP), permanecerá en este estado y en el de pendiente de comprobación. De lo contrario, la tabla se establecerá en un estado pendiente de reconciliación de DataLink (DRP).

La tabla dependiente y la descendiente no quedan afectadas cuando se especifica esta opción.

IMMEDIATE CHECKED
Especifica que la tabla debe tener activadas sus restricciones y que se debe llevar a cabo la comprobación de integridad que se ha diferido. Esto se realiza de acuerdo con la información que se ha establecido en las columnas STATUS y CONST_CHECKED del catálogo SYSCAT.TABLES. Es decir:

- El valor de STATUS debe ser C (la tabla está en estado pendiente de comprobación) o se devolverá un error (SQLSTATE 51027), a menos que la tabla sea una tabla de claves foráneas descendente, una tabla de consultas materializadas descendente o una tabla de etapas descendente de una tabla que se ha especificado en la lista, que esté en estado pendiente de comprobación y cuyos antecesores inmediatos también estén en la lista.
- Si la tabla que está comprobándose está en estado pendiente de comprobación, el valor de CONST_CHECKED indica qué opciones de integridad van a comprobarse.

Si la tabla se ha establecido en estado pendiente de comprobación mediante la utilización de la opción CASCADE DEFERRED, sus tablas de claves foráneas descendentes, tablas de consultas materializadas inmediatas descendentes y tablas de etapas inmediatas descendentes se establecen, si es necesario, en estado sin acceso pendiente de comprobación. Se emitirá un mensaje de aviso (SQLSTATE 01586) para indicar que las tablas descendentes se han establecido en estado pendiente de comprobación.

Si es una tabla de consultas materializadas mantenida por el sistema, los datos se comprueban con la consulta y se renuevan en función de las necesidades. (Esta sentencia no puede utilizarse para las tablas de consultas materializadas mantenidas por el usuario.) Si es una tabla de etapas, los datos se comprueban con su definición de consulta y se propagan en función de las necesidades.

Cuando se comprueba la integridad de una tabla hija:

- Ninguno de sus padres puede estar en estado pendiente de comprobación o
- Cada uno de sus padres debe comprobarse para verificar si existen violaciones de las restricciones en la misma sentencia SET INTEGRITY.

Cuando se renueva una tabla de consultas materializadas inmediata o cuando se propagan deltas a una tabla de etapas:

- Ninguna de sus tablas subyacentes puede estar en estado pendiente de comprobación o
- Cada una de sus tablas subyacentes debe comprobarse en la misma sentencia SET INTEGRITY.

De lo contrario, se devolverá un error (SQLSTATE 428A8).

Los valores de DATALINK no se comprueban, aunque la tabla esté en estado DRP o DRNP. Debe utilizarse la API o el mandato RECONCILE para llevar a cabo la reconciliación de valores DATALINK. Se elimina el estado pendiente de comprobación de la tabla, pero ésta sigue teniendo establecido el distintivo DRP o DRNP. La tabla puede utilizarse, pues la reconciliación de los valores de DATALINK se difiere.

opciones-comprobación

opciones-incrementales

INCREMENTAL
Especifica la aplicación de comprobaciones de integridad a la parte añadida (si existe) de la tabla. Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que toda la tabla debe comprobarse para verificar la integridad de los datos), se devuelve un error (SQLSTATE 55019).

NOT INCREMENTAL
Especifica la aplicación de comprobaciones de integridad a toda la tabla. Si la tabla es una tabla de consultas materializadas, se vuelve a calcular la definición de la tabla de consultas materializadas. Si la tabla incluye la definición de, como mínimo, una restricción, esta opción hace que tenga lugar el proceso completo de las tablas de claves foráneas descendentes y de las tablas de consultas materializadas inmediatas descendentes. Si la tabla es una tabla de etapas, se establece en estado no coherente.

Si no se especifica la cláusula *opciones-incrementales*, el sistema determinará si es posible el proceso incremental; si no es posible, se comprobará toda la tabla.

FORCE GENERATED
Si la tabla incluye columnas generadas, los valores se calculan basándose en la expresión y se almacenan en la columna. Si no se especifica esta cláusula, los valores actuales se comparan con el valor calculado de la expresión, como si existiera una restricción de comprobación de igualdad. Si se comprueba la integridad de la tabla de manera incremental, las columnas generadas sólo se calcularán para la parte añadida.

PRUNE
Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH). Si también se especifica la opción de comprobación INCREMENTAL, se devuelve un error (SQLSTATE 428FH).

FULL ACCESS
Especifica que las tablas van a ser tablas de acceso completo tras

ejecutarse la sentencia SET INTEGRITY. Esta opción puede especificarse en las cláusulas IMMEDIATE CHECKED e IMMEDIATE UNCHECKED.

Cuando se realice el proceso incremental de una tabla subyacente de la lista de invocación y ésta tenga tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establecerá, en función de las necesidades, en la modalidad sin movimiento de datos tras emitirse la sentencia SET INTEGRITY. Cuando se elimine el estado pendiente de comprobación de todas las tablas de etapas y tablas de consultas materializadas inmediatas dependientes que pueden renovarse de manera incremental, la modalidad sin movimiento de datos de la tabla subyacente cambiará por la modalidad de acceso completo. Si se especifica la opción FULL ACCESS en la cláusula IMMEDIATE CHECKED o IMMEDIATE UNCHECKED, la tabla subyacente eludirá la modalidad sin movimiento de datos y se establecerá directamente en modalidad de acceso completo. Las tablas de consultas materializadas inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden señalarse como incoherentes.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en modalidad de acceso completo tras emitirse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

La especificación de la opción FULL ACCESS en la cláusula IMMEDIATE UNCHECKED generará un error (SQLSTATE 428FH) si la sentencia no elimina el estado pendiente de comprobación de la tabla.

cláusula-excepción

FOR EXCEPTION
Indica que cualquier fila que viole una restricción de clave foránea o una restricción de comprobación se copiará en una tabla de excepciones y se suprimirá de la tabla original. Aunque se detecten errores, las restricciones vuelven a activarse y se elimina el estado pendiente de comprobación de la tabla. Se emite un aviso (SQLSTATE 01603) para indicar que se ha movido una o varias filas a las tablas de excepciones.

Si no se especifica la cláusula FOR EXCEPTION y se produce una violación de alguna restricción, al usuario sólo se le devolverá la primera violación que se ha detectado (SQLSTATE 23514). Si se produce una violación en alguna de las tablas, todas las tablas quedarán en estado pendiente de comprobación, tal como estaban antes de ejecutarse la sentencia.

IN *nombre-tabla*
Especifica la tabla de la que se deben copiar las filas que violan restricciones. Debe haber una tabla de excepciones especificada para cada tabla que se comprueba. Esta cláusula no puede especificarse si la tabla es una tabla de consultas materializadas o una tabla de etapas (SQLSTATE 428A7).

USE *nombre-tabla*
Especifica la tabla de excepciones en la que se deben copiar las filas erróneas.

FULL ACCESS
Si se especifica la opción FULL ACCESS como única operación de la sentencia, la modalidad sin movimiento de datos de la tabla cambiará por la modalidad de acceso completo. La tabla no se comprobará para verificar si existen violaciones de integridad. Sin embargo, las tablas de consultas materializadas inmediatas dependientes que no se han renovado podrían necesitar un nuevo cálculo completo en las sentencias REFRESH posteriores, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla podrían cambiar para establecerse en estado incompleto. Esta opción sólo puede especificarse para una tabla que está en modalidad sin movimiento de datos (SQLSTATE 428FH).

PRUNE
Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH).

opciones-no-seleccionadas-tabla

nombre-tabla
Identifica una o más tablas para el proceso de la integridad. Debe ser una tabla descrita en el catálogo y no debe ser una vista, una tabla del catálogo ni una tabla con tipo.

opciones-integridad
Se utiliza para definir las opciones de integridad que se establecen en IMMEDIATE UNCHECKED.

ALL
Se activarán todas las opciones de integridad y se eliminará el estado pendiente de comprobación de la tabla.

FOREIGN KEY
Las restricciones de clave foránea se activarán cuando se elimine el estado pendiente de comprobación de la tabla.

CHECK
Las restricciones de comprobación se activarán cuando se elimine el estado pendiente de comprobación de la tabla.

DATALINK RECONCILE PENDING
Las restricciones de integridad de DATALINK se activarán cuando se elimine el estado pendiente de comprobación de la tabla.

MATERIALIZED QUERY
Se activará la renovación inmediata para una tabla de consultas materializadas con el atributo REFRESH IMMEDIATE.

GENERATED COLUMN
Las columnas generadas se activarán cuando se elimine el estado pendiente de comprobación de la tabla.

STAGING
Se activará la propagación inmediata para la tabla de etapas.

FULL ACCESS
Especifica que las tablas van a ser tablas de acceso completo tras ejecutarse

la sentencia SET INTEGRITY. Esta opción puede especificarse en las cláusulas IMMEDIATE CHECKED e IMMEDIATE UNCHECKED.

Cuando se realice el proceso incremental de una tabla subyacente de la lista de invocación y ésta tenga tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establecerá, en función de las necesidades, en la modalidad sin movimiento de datos tras emitirse la sentencia SET INTEGRITY. Cuando se elimine el estado pendiente de comprobación de todas las tablas de etapas y tablas de consultas materializadas inmediatas dependientes que pueden renovarse de manera incremental, la modalidad sin movimiento de datos de la tabla subyacente cambiará por la modalidad de acceso completo. Si se especifica la opción FULL ACCESS en la cláusula IMMEDIATE CHECKED o IMMEDIATE UNCHECKED, la tabla subyacente eludirá la modalidad sin movimiento de datos y se establecerá directamente en modalidad de acceso completo. Las tablas de consultas materializadas inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden señalarse como incoherentes.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en modalidad de acceso completo tras emitirse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

La especificación de la opción FULL ACCESS en la cláusula IMMEDIATE UNCHECKED generará un error (SQLSTATE 428FH) si la sentencia no elimina el estado pendiente de comprobación de la tabla.

IMMEDIATE UNCHECKED

Especifica uno de los siguientes:

- La tabla va a tener activada su comprobación de integridad (y, por lo tanto, se eliminará el estado pendiente de comprobación de la tabla) y no se comprobará si existen violaciones de integridad.

  Esto se indica especificando ALL o especificando una o más de las otras *opciones-integridad* para cada opción de integridad que está desactivada para esa tabla.

- La tabla va a tener activado un tipo de comprobación de integridad, sin comprobarse si existen violaciones de integridad para este tipo, pero quedará en estado pendiente de comprobación.

  Esto se indica especificando una o más de las opciones CHECK, FOREIGN KEY, MATERIALIZED QUERY, STAGING, GENERATED COLUMN o DATALINK RECONCILE PENDING de forma que, como mínimo, una de las opciones de integridad siga desactivada para esa tabla.

Deben tenerse en cuenta las implicaciones con respecto a la integridad de los datos antes de utilizar esta opción. Consulte el apartado “Notas” de esta sentencia.

Notas:

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - SET CONSTRAINTS puede especificarse en lugar de SET INTEGRITY

- SUMMARY puede especificarse en lugar de MATERIALIZED QUERY

- Efectos en las tablas que están en uno de los estados pendientes de comprobación:
  - La utilización de INSERT, UPDATE o DELETE no está permitida en una tabla que está en estado de lectura o sin acceso pendiente de comprobación. Además, se rechazará cualquier sentencia que necesite una modificación de este tipo para una tabla que está en estado pendiente de comprobación.

    Por ejemplo, no está permitido suprimir una fila de una tabla padre que se aplica en cascada a una tabla dependiente que está en estado pendiente de comprobación.
  - La utilización de SELECT no está permitida en una tabla que está en estado sin acceso pendiente de comprobación. Además, se rechazará cualquier sentencia que necesite acceso de lectura a una tabla que está en estado sin acceso pendiente de comprobación.
  - Normalmente las nuevas restricciones añadidas a una tabla se imponen inmediatamente. Sin embargo, si la tabla está en estado pendiente de comprobación, la comprobación de cualquier restricción nueva se diferirá hasta que se elimine el estado pendiente de comprobación de la tabla. Si la tabla está en estado de lectura pendiente de comprobación, la adición de una nueva restricción hará que la tabla entre en estado si acceso pendiente de comprobación, pues no se ha comprobado la validez de los datos.
  - La sentencia CREATE INDEX no puede hacer referencia a ninguna tabla que esté en estado pendiente de comprobación. De forma similar, una tabla ALTER TABLE para añadir una clave primaria o una restricción de unicidad no puede hacer referencia a ninguna tabla que esté en estado pendiente de comprobación.
  - El programa de utilidad IMPORT no puede utilizarse para realizar operaciones en una tabla que está en estado pendiente de comprobación. (El programa de utilidad IMPORT se diferencia del programa de utilidad LOAD en que siempre comprueba las restricciones de forma inmediata.)
  - El programa de utilidad EXPORT no puede utilizarse para realizar operaciones en una tabla que está en estado pendiente de comprobación, pero sí puede utilizarse para realizar operaciones en una tabla que está en estado de lectura pendiente de comprobación. Si una tabla está en estado de lectura pendiente de comprobación, el programa de utilidad EXPORT sólo exportará los datos que no están en la parte añadida.
  - Las operaciones (como, por ejemplo, REORG, REDISTRIBUTE, la actualización de la clave de particionamiento, la actualización de la clave de clúster, etc.) que pueden implicar un movimiento de datos dentro de una tabla no pueden utilizarse para realizar acciones en una tabla que está en alguno de los estados pendientes de comprobación o que está en la modalidad sin movimiento de datos.
  - Los programas de utilidad LOAD, BACKUP, RESTORE, UPDATE STATISTICS, RUNSTATS, REORGCHK, LIST HISTORY y ROLLFORWARD pueden utilizarse en una tabla que está en cualquiera de los estados pendientes de comprobación.
  - Las sentencias ALTER TABLE, COMMENT, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE y SET INTEGRITY pueden hacer referencia a una tabla que está en cualquiera de los estados pendientes de comprobación. Sin embargo, pueden dar lugar a que la tabla se establezca en la modalidad sin acceso.
  - Los paquetes, las vistas y cualquier otro objeto que dependa de una tabla que está en estado sin acceso pendiente de comprobación devolverán un error al

accederse a la tabla en tiempo de ejecución. Los paquetes que dependan de una tabla que está en estado de lectura pendiente de comprobación devolverán un error cuando se intente realizar una operación de inserción, actualización o supresión en la tabla en tiempo de ejecución.

La eliminación de las filas con las violaciones mediante la sentencia SET INTEGRITY no es un suceso de supresión. Por lo tanto, una sentencia SET INTEGRITY nunca activa los activadores. Similarmente, la actualización de columnas mediante la opción FORCE GENERATED no produce la activación de activadores.

- El proceso incremental se utilizará siempre que la situación lo permita, pues es más eficaz. En la mayoría de los casos, no se necesita la opción INCREMENTAL. Sin embargo, sí se necesita para garantizar que las comprobaciones de integridad realmente se procesan de forma incremental. Si el sistema detecta que es necesario realizar un proceso completo para garantizar la integridad de los datos, se devuelve un error (SQLSTATE 55019).
- Aviso acerca de la utilización de la cláusula IMMEDIATE UNCHECKED:
  – Esta cláusula se ha diseñado para que la utilicen los programas de utilidad y no se recomienda que la utilicen los programas de aplicación. Si en la tabla existen datos que no satisfacen las especificaciones de integridad que se han definido para la tabla y se utiliza la cláusula IMMEDIATE CHECKED, puede que se devuelvan resultados de consulta incorrectos.

    El hecho de haberse activado la comprobación de la integridad sin realizarse la comprobación diferida quedará registrado en el catálogo (el valor de la columna CONST_CHECKED de la vista SYSCAT.TABLES se establecerá en 'U'). Esto indica que el usuario ha asumido la tarea de asegurar la integridad de los datos con respecto a las restricciones específicas. Este valor no cambia hasta que:

    - La tabla vuelve a establecerse en estado pendiente de comprobación (haciéndose referencia a la tabla en una sentencia SET INTEGRITY con la cláusula OFF), momento en el que los valores 'U' de la columna CONST_CHECKED cambiarán por valores 'W', para indicar que el usuario había asumido anteriormente la responsabilidad de comprobar la integridad de los datos y que el sistema debe verificar los datos.
    - Se eliminan todas las restricciones no comprobadas para la tabla.

    El estado 'W' se diferencia del estado 'N' en que registra el hecho de que el usuario comprobaba anteriormente la integridad, pero que el sistema todavía no lo hace. Si el usuario emite la sentencia SET INTEGRITY ... IMMEDIATE CHECKED con la opción NOT INCREMENTAL, el sistema vuelve a comprobar toda la tabla para verificar la integridad de los datos (o realiza una renovación completa en una tabla de consultas materializadas) y, a continuación, cambia el estado 'W' por el estado 'Y'. Si se especifica IMMEDIATE UNCHECKED, o si no se especifica NOT INCREMENTAL, el estado 'W' vuelve a cambiar por el estado 'U' para registrar el hecho de que el sistema todavía no ha verificado algunos de los datos. En este último caso (cuando no se especifica NOT INCREMENTAL), se devuelve un aviso (SQLSTATE 01636).

    Si se ha comprobado la integridad de una tabla subyacente utilizando la cláusula IMMEDIATE UNCHECKED, los valores 'U' de la columna CONST_CHECKED de la tabla subyacente se propagarán a la columna CONST_CHECKED correspondiente de:

    - Las tablas de consultas materializadas inmediatas dependientes
    - Las tablas de consultas materializadas diferidas dependientes
    - Las tablas de etapas dependientes

Para una tabla de consultas materializadas inmediata dependiente, esta propagación tiene lugar siempre que se elimina el estado pendiente de comprobación de la tabla subyacente y siempre que se renueva la tabla de consultas materializadas. Para una tabla de consultas materializadas diferida dependiente, esta propagación tiene lugar siempre que se renueva la tabla de consultas materializadas. Para las tablas de etapas dependientes, esta propagación tiene lugar siempre que se elimina el estado pendiente de comprobación de la tabla subyacente. Estos valores 'U' propagados de las columnas CONST_CHECKED de las tablas de consultas materializadas dependientes y tablas de etapas registran el hecho de que estas tablas de consultas materializadas y tablas de etapas dependen de alguna tabla subyacente cuya integridad se ha comprobado utilizando la cláusula IMMEDIATE UNCHECKED.

Para una tabla de consultas materializadas, el valor 'U' de la columna CONST_CHECKED que la tabla subyacente ha propagado se conservará hasta que la tabla de consultas materializada se haya renovado por completo y ninguna de sus tablas subyacentes tenga un valor 'U' en su correspondiente columna CONST_CHECKED. Tras realizarse esta renovación, el valor 'U' de la columna CONST_CHECKED de la tabla de consultas materializadas cambiará por 'Y'.

Para una tabla de etapas, el valor 'U' de la columna CONST_CHECKED que la tabla subyacente ha propagado se conservará hasta que se haya renovado la correspondiente tabla de consultas materializadas diferida de la tabla de etapas. Tras realizarse esta renovación, el valor 'U' de la columna CONST_CHECKED de la tabla de etapas cambiará por 'Y'.

– Si una tabla hija y su tabla padre se comprueban en la misma sentencia SET INTEGRITY ... IMMEDIATE CHECKED y la tabla padre necesita que se realice una renovación completa de sus restricciones, se comprobarán las restricciones de clave foránea de la tabla hija, con independencia de si la tabla hija tiene o no un valor 'U' en la columna CONST_CHECKED para las restricciones de clave foránea.

- Después de haber añadido datos utilizando LOAD INSERT, la sentencia SET INTEGRITY ... IMMEDIATE CHECKED comprueba la tabla para verificar si existen violaciones de restricciones. El sistema determina si es posible o no el proceso incremental en la tabla. Si es posible, sólo se comprueba la parte añadida para las violaciones de integridad. Si no es posible, el sistema comprueba toda la tabla para verificar si existen violaciones de integridad.
- Veamos la sentencia siguiente:

```
SET INTEGRITY FOR T IMMEDIATE CHECKED
```

A continuación se indican las situaciones en las que el sistema deberá realizar una renovación completa o en las que comprobará la integridad de toda la tabla (no puede especificarse la opción INCREMENTAL):

– Cuando se han añadido nuevas restricciones a la propia T.

– Cuando ha tenido lugar una operación LOAD REPLACE para T, su tabla padre o sus tablas subyacentes.

– Cuando se ha activado la opción NOT LOGGED INITIALLY WITH EMPTY TABLE tras la última comprobación de integridad realizada en T, en sus tablas padre o en sus tablas subyacentes.

– El efecto de aplicación en cascada del proceso completo, cuando se ha comprobado la integridad de T (o una tabla subyacente, si T es una tabla de consultas materializadas o una tabla de etapas) de forma no incremental.

  - Si la tabla se encontraba en estado pendiente de comprobación antes de la migración, debe realizarse el proceso completo la primera vez que se compruebe la integridad de la tabla tras la migración.
  - Si el espacio de tablas que contiene la tabla o su tabla padre (o tabla subyacente de una tabla de consultas materializadas o una tabla de etapas) se ha retrotraído hasta un momento dado y la tabla y su tabla padre (o tabla subyacente si la tabla es una tabla de consultas materializadas o una tabla de etapas) residen en distintos espacios de tabla.
  - Cuando T es una tabla de consultas materializadas y ha tenido lugar una operación LOAD REPLACE o LOAD INSERT directamente en T tras la última renovación.
- Si las condiciones del proceso completo que se describen en el punto anterior no se satisfacen, el sistema intentará comprobar sólo la integridad de la parte añadida o realizar una renovación incremental (si es una tabla de consultas materializadas) cuando el usuario no especifique la opción NOT INCREMENTAL para la sentencia `SET INTEGRITY FOR T IMMEDIATE CHECKED`.
- Una tabla que se encuentre en estado de reconciliación de DataLink no posible (DRNP) necesita que se lleve a cabo una acción correctiva (posiblemente fuera de la base de datos). Una vez completada la acción correctiva, la tabla deja el estado DRNP mediante la opción IMMEDIATE UNCHECKED. Deberá utilizarse entonces la API o el mandato RECONCILE para comprobar las restricciones de integridad de DATALINK.
- Mientras está comprobándose la integridad, se retiene un bloqueo exclusivo en cada tabla que se ha especificado durante la invocación de SET INTEGRITY. Se adquiere un bloqueo compartido en cada tabla que no se ha especificado durante la invocación de SET INTEGRITY pero que es una tabla padre o subyacente de una de las tablas dependientes que está comprobándose.
- Si se produce un error durante la comprobación de la integridad, se retrotraerán todos los efectos de la comprobación (incluidas la supresión realizada en la tabla original y la inserción en las tablas de excepciones).
- Si la emisión de una sentencia SET INTEGRITY con la opción FORCE GENERATED no se ejecuta satisfactoriamente porque el espacio de anotaciones cronológicas es insuficiente, incremente el espacio de anotaciones cronológicas activo disponible y vuelva a emitir la sentencia SET INTEGRITY. Alternativamente, utilice la sentencia SET INTEGRITY GENERATED COLUMN IMMEDIATE UNCHECKED para eludir la comprobación de las columnas generadas para la tabla. A continuación, emita una sentencia SET INTEGRITY IMMEDIATE CHECKED sin la opción FORCE GENERATED para comprobar si en la tabla existen otras violaciones de la integridad (si se aplica) y para eliminar el estado pendiente de comprobación de la tabla. Cuando se haya eliminado el estado pendiente de comprobación de la tabla, las columnas generadas podrán actualizarse para que tomen sus valores por omisión (generados) asignando éstos a la palabra clave DEFAULT en una sentencia UPDATE. Esta acción se realiza utilizando varias sentencias de actualización buscada basadas en rangos (cada una de ellas seguida de una confirmación) o bien haciendo uso de un enfoque basado en cursores utilizando confirmaciones intermitentes. Deberá utilizarse un cursor “con retención” si van a retenerse bloqueos tras las confirmaciones intermitentes del enfoque basado en cursores.
- Cuando una tabla se ha establecido en estado pendiente de comprobación mediante la utilización de la opción CASCADE DEFERRED (de la sentencia SET INTEGRITY o del mandato LOAD) y se comprueba para verificar si existen violaciones de integridad mediante la utilización de la opción IMMEDIATE CHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendentes, tablas de consultas materializadas inmediatas descendentes y

tablas de etapas inmediatas descendentes se establecerán en estado sin acceso pendiente de comprobación, según proceda:

- Si se comprueba toda la tabla para verificar si existen violaciones de integridad, sus tablas de claves foráneas descendentes, tablas de consultas materializadas inmediatas descendentes y tablas de etapas inmediatas descendentes se establecerán en estado sin acceso pendiente de comprobación.
- Si la tabla se comprueba para verificar si existen violaciones de integridad de manera incremental, sus tablas de consultas materializadas inmediatas descendentes y tablas de etapas se establecerán en estado sin acceso pendiente de comprobación y sus tablas de claves foráneas descendentes seguirán en sus estados originales.
- Si la tabla no necesita ninguna comprobación, todas sus tablas de consultas materializadas inmediatas descendentes, tablas de etapas descendentes y tablas de claves foráneas descendentes seguirán en sus estados originales.

- Cuando una tabla se ha establecido en estado pendiente de comprobación mediante la utilización de la opción CASCADE DEFERRED (de la sentencia SET INTEGRITY o del mandato LOAD) y se ha eliminado su estado pendiente de comprobación mediante la utilización de la opción IMMEDIATE UNCHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendentes, tablas de consultas materializadas inmediatas descendentes y tablas de etapas inmediatas descendentes se establecerán en estado sin acceso pendiente de comprobación, según proceda:
  - Si la tabla se ha cargado utilizando la modalidad REPLACE, sus tablas de claves foráneas descendentes, tablas de consultas materializadas inmediatas descendentes y tablas de etapas inmediatas descendentes se establecerán en estado sin acceso pendiente de comprobación.
  - Si la tabla se ha cargado utilizando la modalidad INSERT, sus tablas de consultas materializadas inmediatas descendentes y tablas de etapas se establecerán en estado sin acceso pendiente de comprobación y sus tablas de claves foráneas descendentes seguirán en sus estados originales.
  - Si la tabla no se ha cargado, sus tablas de consultas materializadas inmediatas descendentes, tablas de etapas descendentes y tablas de claves foráneas descendentes seguirán en sus estados originales.

Ejemplos:

Ejemplo 1: El siguiente es un ejemplo de una consulta que proporciona información acerca del estado pendiente de comprobación de las tablas. SUBSTR se utiliza para extraer los dos primeros bytes de la columna CONST_CHECKED de SYSCAT.TABLES. El primer byte representa las restricciones de clave foránea y el segundo byte representa las restricciones de comprobación. STATUS proporciona el estado pendiente de comprobación y ACCESS_MODE proporciona la modalidad de acceso.

```
SELECT TABNAME, STATUS, ACCESS_MODE,
  SUBSTR( CONST_CHECKED, 1, 1 ) AS FK_CHECKED,
  SUBSTR( CONST_CHECKED, 2, 1 ) AS CC_CHECKED
   FROM SYSCAT.TABLES
```

Ejemplo 2: Establezca las tablas T1 y T2 en estado sin acceso pendiente de comprobación y aplique inmediatamente en cascada el estado pendiente de comprobación a sus descendentes.

```
SET INTEGRITY FOR T1, T2 OFF NO ACCESS CASCADE IMMEDIATE
```

Ejemplo 3: Establezca la tabla padre T1 en estado de lectura pendiente de comprobación sin aplicar inmediatamente en cascada el estado pendiente de comprobación a su tabla hija T2.

```
SET INTEGRITY FOR T1 OFF READ ACCESS CASCADE DEFERRED
```

Ejemplo 4: Compruebe la integridad de T1 y obtenga sólo la primera violación.

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
```

Ejemplo 5: Compruebe la integridad de T1 y T2 y coloque las filas con violaciones en las tablas de excepciones E1 y E2.

```
SET INTEGRITY FOR T1, T2 IMMEDIATE CHECKED
   FOR EXCEPTION IN T1 USE E1,
                 IN T2 USE E2
```

Ejemplo 6: Habilite la comprobación de restricciones FOREIGN KEY en T1 y la comprobación de restricciones CHECK en T2 de modo que se eludan con la opción IMMEDIATE UNCHECKED.

```
SET INTEGRITY FOR T1 FOREIGN KEY,
                                       T2 CHECK IMMEDIATE UNCHECKED
```

Ejemplo 7: Añada una restricción de comprobación y una clave foránea a la tabla EMP_ACT utilizando dos sentencias ALTER TABLE. Para realizar la comprobación de restricciones en un único pase de la tabla, la comprobación de la integridad se desactiva antes de invocarse las sentencias ALTER y se activa después de haberse ejecutado las sentencias.

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
```

Ejemplo 8: Establezca la integridad para las columnas generadas.

```
SET INTEGRITY FOR T1 IMMEDIATE CHECKED
  FORCE GENERATED
```

Ejemplo 9: Realice adiciones (utilizando LOAD INSERT) desde distintas fuentes a una tabla subyacente UT1 de una tabla de consultas materializadas REFRESH IMMEDIATE, AST1, compruebe la integridad de los datos de UT1 (de manera incremental) y renueve AST1 (de manera incremental). En esta situación de ejemplo, la comprobación de la integridad que se realiza para UT1 y la renovación de AST1 son incrementales, pues el sistema elige el proceso incremental.

```
LOAD FROM IMTFILE1.IXF OF IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF OF IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;
REFRESH TABLE AST1;
```

Información relacionada:

- "Tablas de excepciones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "TbGenCol.java -- How to use generated columns (JDBC)"

SET PASSTHRU

La sentencia SET PASSTHRU abre y cierra una sesión para someter directamente el SQL nativo de una fuente de datos a esa fuente de datos. La sentencia no está bajo el control de la transacción.

Invocación:

Esta sentencia puede emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Los privilegios que tiene el ID de autorización de la sentencia deben proporcionar autorización para:

- Realizar un paso a través para la fuente de datos.
- Satisfacer las medidas de seguridad en la fuente de datos.

Sintaxis:

Descripción:

nombre-servidor
Indica la fuente de datos para la que se debe abrir una sesión de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

RESET
Cierra una sesión de paso a través.

Notas:

- A las fuentes de datos de Microsoft SQL Server, Sybase y Oracle se aplican las restricciones siguientes:
  - Las transacciones definidas por el usuario no pueden utilizarse para fuentes de datos de Microsoft SQL Server y Sybase en modalidad de paso a través, pues Microsoft SQL Server y Sybase restringen qué sentencias de SQL pueden especificarse dentro de una transacción definida por el usuario. Puesto que DataJoiner no analiza las sentencias de SQL que se procesan en modalidad de paso a través, no es posible detectar si el usuario ha especificado una sentencia de SQL que está permitida dentro de una transacción definida por el usuario.
  - La cláusula COMPUTE no recibe soporte en fuentes de datos de Microsoft SQL Server y Sybase.
  - Las sentencias de DDL no están sujetas a la semántica de transacción en fuentes de datos de Microsoft SQL Server, Oracle y Sybase. Microsoft SQL Server, Oracle o Sybase confirman la operación automáticamente cuando ésta se ha completado. Si se produce una retrotracción, el DDL no se retrotrae.

Ejemplos:

Ejemplo 1: Inicie una sesión de paso a través para la fuente de datos BACKEND.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
```

Ejemplo 2: Inicie una sesión de paso a través con una sentencia PREPARE.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;
```

Ejemplo 3: Finalice una sesión de paso a través.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

Ejemplo 4: Utilice las sentencias PREPARE y EXECUTE para finalizar una sesión de paso a través.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

Ejemplo 5: Abra una sesión para que tenga lugar el paso a través para una fuente de datos, cree un índice de clúster para una tabla en esta fuente de datos y cierre la sesión de paso a través.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT                modalidad de paso a través
FROM "CREATE UNIQUE
        CLUSTERED INDEX TABLE_INDEX
        ON USER2.TABLE              la tabla no es un
        WITH IGNORE DUP KEY";        alias
EXEC SQL EXECUTE STMT;
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

SET PATH

La sentencia SET PATH cambia el valor del registro especial CURRENT PATH. No está bajo el control de la transacción.

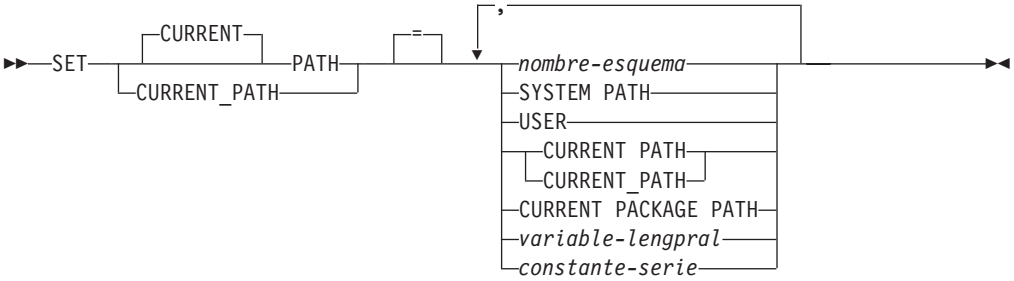
Invocación:

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

nombre-esquema
Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. No se efectúa ninguna validación de que existe el esquema en el momento en que se establece la vía de acceso. Si, por ejemplo, un *nombre-esquema* está mal escrito, el error no se capturará y podría afectar a las operaciones posteriores de SQL.

SYSTEM PATH
Este valor equivale a especificar los nombres de esquema *"SYSIBM","SYSFUN","SYSPROC"*.

USER
El valor del registro especial USER.

CURRENT PATH
El valor del registro especial CURRENT PATH antes que se ejecute esta sentencia.

CURRENT PACKAGE PATH
El valor del registro especial CURRENT PACKAGE PATH.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lengpral* no debe ser mayor que 30 (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lengpral* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lengpral*, deben especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea ya que no se efectúa la conversión a mayúsculas.

constante-serie
Una constante de serie de caracteres con una longitud máxima de 30 bytes.

Normas:

- Un nombre de esquema no puede aparecer más de una vez en la vía de acceso de función (SQLSTATE 42732).
- El número de esquemas que se pueden especificar está limitado por la longitud total del registro especial CURRENT PATH. La serie del registro especial se crea tomando cada nombre de esquema especificado y eliminando los blancos de cola, delimitando con comillas dobles, doblando las comillas dentro del nombre de esquema cuando sea necesario y, después, separando cada nombre de esquema por una coma. La longitud de la serie resultante no puede exceder de 254 bytes (SQLSTATE 42907).

Notas:

- Compatibilidades
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - CURRENT FUNCTION PATH puede utilizarse en lugar de CURRENT PATH
- El valor inicial del registro especial CURRENT PATH es ″SYSIBM″,″SYSFUN″,″SYSPROC″,″X″ donde la X es el valor del registro especial USER.
- No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema (en este caso, no se incluye en el registro especial CURRENT PATH).
- El registro especial CURRENT PATH especifica la vía de acceso de SQL utilizada para resolver las funciones, los procedimientos y los tipos de datos definidos por el usuario de las sentencias de SQL dinámico. La opción de enlace lógico FUNCPATH especifica la vía de acceso de SQL que se debe utilizar para resolver las funciones y los tipos de datos definidos por el usuario de las sentencias de SQL estático.

Ejemplos:

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT PATH.

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

Ejemplo 2: El ejemplo siguiente recupera el valor actual del registro especial CURRENT PATH en la variable del lenguaje principal denominada CURPATH.

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

El valor sería ″FERMAT″,″McDrw #8″,″SYSIBM″ si se estableciese por el ejemplo anterior.

SET SCHEMA

La sentencia SET SCHEMA cambia el valor del registro especial CURRENT SCHEMA. No está bajo el control de la transacción. Si el paquete está enlazado con la opción DYNAMICRULES BIND, esta sentencia no afecta al calificador utilizado para referencias de objetos de base de datos no calificadas.

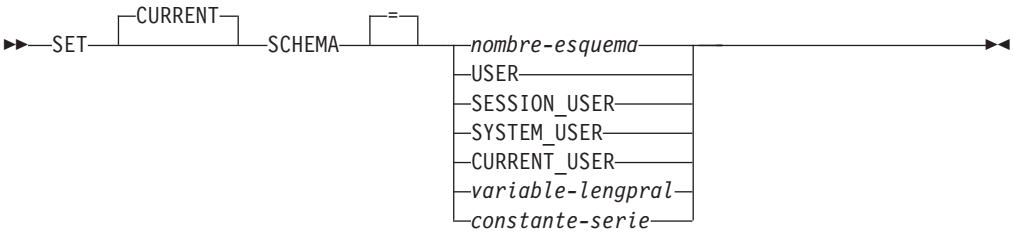
Invocación:

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

No se necesita ninguna autorización para ejecutar esta sentencia.

Sintaxis:



Descripción:

nombre-esquema
Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. La longitud no debe ser mayor que 30 bytes (SQLSTATE 42815). No se efectúa ninguna validación de que existe el esquema en el momento en que se establece el esquema. Si un *nombre-esquema* está escrito incorrectamente, no se capturará y ello puede afectar a las operaciones subsiguientes de SQL.

USER
El valor del registro especial USER.

SESSION_USER
El valor del registro especial SESSION_USER.

SYSTEM_USER
El valor del registro especial SYSTEM_USER.

CURRENT_USER
El valor del registro especial CURRENT_USER.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lengpral* no debe ser mayor que 30 (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lengpral* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lengpral*, deben

especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea que estén ya que no se realiza ninguna conversión a caracteres en mayúsculas.

constante-serie
Es una constante de tipo serie de caracteres con una longitud máxima de 30.

Normas:

- Si el valor especificado no se adapta a las normas para un *nombre-esquema*, se produce un error (SQLSTATE 3F000).
- El valor del registro especial CURRENT SCHEMA se utiliza como nombre de esquema en todas las sentencias de SQL dinámico, con la excepción de la sentencia CREATE SCHEMA, donde existe una referencia no calificada a un objeto de base de datos.
- La opción de enlace QUALIFIER especifica el nombre de esquema que debe utilizarse como calificador para los nombres de objetos de base de datos no calificados en las sentencias de SQL estático.

Notas:

- El valor inicial del registro especial CURRENT SCHEMA equivale a USER.
- El establecimiento del registro especial CURRENT SCHEMA no afecta al registro especial CURRENT PATH. En consecuencia, el registro especial CURRENT SCHEMA no se incluirá en la vía de acceso de SQL y es posible que la resolución de funciones, procedimientos y tipos definidos por el usuario no encuentre estos objetos. Para incluir el valor del esquema actual en la vía de acceso de SQL, siempre que se emita la sentencia SET SCHEMA, emita también la sentencia SET PATH incluyendo el nombre de esquema de la sentencia SET SCHEMA.
- CURRENT SQLID se acepta como sinónimo de CURRENT SCHEMA y el efecto de una sentencia SET CURRENT SQLID será idéntico al de una sentencia SET CURRENT SCHEMA. No tendrán lugar otros efectos, como, por ejemplo, cambios de autorización de la sentencia.

Ejemplos:

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT SCHEMA.

```
SET SCHEMA RICK
```

Ejemplo 2: El ejemplo siguiente recupera el valor actual del registro especial CURRENT SCHEMA en la variable del lenguaje principal denominada CURSCHEMA.

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

El valor sería RICK, establecido en el ejemplo anterior.

SET SERVER OPTION

La sentencia SET SERVER OPTION especifica un valor de opción de servidor que debe permanecer en vigor mientras un usuario o una aplicación esté conectado a la base de datos federada. Cuando finaliza la conexión, se reinstaura el valor anterior de esta opción de servidor. La sentencia no está bajo el control de la transacción.

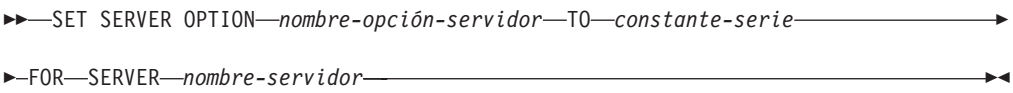
Invocación:

Esta sentencia puede emitirse interactivamente. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

El ID de autorización de la sentencia debe tener autorización SYSADM o DBADM en la base de datos federada.

Sintaxis:



Descripción:

nombre-opción-servidor
Nombra la opción del servidor que se debe establecer.

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

SERVER *nombre-servidor*
Nombra la fuente de datos a la que se aplica *nombre-opción-servidor*. Debe ser un servidor descrito en el catálogo.

Notas:

- Los nombres de opción del servidor se pueden entrar en mayúsculas o minúsculas.
- Se pueden someter una o varias sentencias SET SERVER OPTION cuando un usuario o una aplicación se conecta a la base de datos federada. La sentencia (o sentencias) debe especificarse en el inicio de la primera unidad de trabajo que se procese después de establecer la conexión.
- SYSCAT.SERVEROPTIONS no se actualizará basándose en una sentencia SET SERVER OPTION, pues este cambio sólo afecta a la conexión actual.

Ejemplos:

Ejemplo 1: Se ha definido una fuente de datos de Oracle denominado ORASERV para una base de datos federada denominada DJDB. ORASERV se ha configurado para inhabilitar la indicación de planes. Sin embargo, el DBA desearía que las indicaciones de planes estuviesen habilitadas para una ejecución de prueba de una aplicación nueva. Cuando la ejecución finalice, las indicaciones de planes se inhabilitarán de nuevo.

```
CONNECT TO DJDB;
strcpy(stmt,"establecer la opción de servidor plan_hints en 'Y' para el
servidor oraserv");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"seleccionar c1 en ora_t1 donde c1 > 100"); /*Generar
indicaciones de planes*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

Ejemplo 2: Ha establecido la opción de servidor PASSWORD en 'Y' (validando las contraseñas en la fuente de datos) para todas las fuentes de datos de Oracle 8. Sin embargo, para una sesión en particular en la que una aplicación se conecta con la base de datos federada para acceder a una fuente de datos de Oracle 8 específico —uno que se ha definido para la base de datos federada DJDB como ORA8A— no será necesario validar las contraseñas.

```
CONNECT TO DJDB;
strcpy(stmt,"establecer la opción de servidor password en 'N' para
      el servidor ora8a");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL EXECUTE STMT_NAME FROM :stmt;
strcpy(stmt,"seleccionar max(c1) en ora8a_t1");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR STMT_NAME;
EXEC SQL OPEN c1; /*No valida la contraseña en ora8a*/
EXEC SQL FETCH c1 INTO :hv;
```

Información relacionada:

- "Server options for federated systems" en la publicación *Federated Systems Guide*

SET SESSION AUTHORIZATION

La sentencia SET SESSION AUTHORIZATION cambia el valor del registro especial SESSION_USER. No está bajo el control de la transacción. Esta sentencia tiene como objetivo proporcionar soporte para un solo usuario, suponiendo que los ID de autorización son distintos en la misma conexión y no se debe utilizar en escenarios en los que distintos usuarios reutilizan la misma conexión, lo que normalmente se denomina agrupación de conexiones.

Invocación:

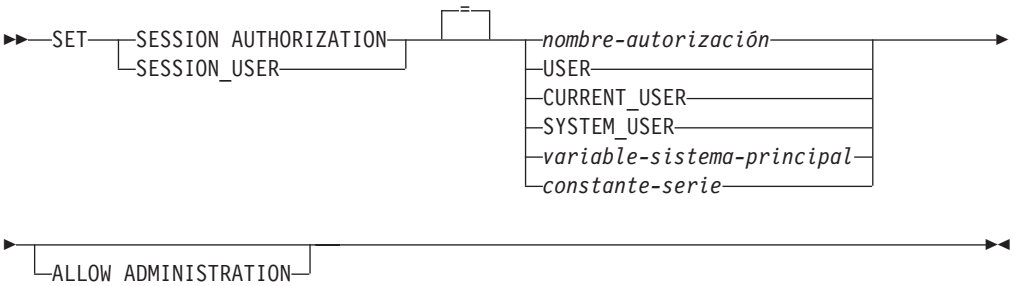
La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

Para cambiar el ID de autorización por un valor diferente del utilizado para establecer la conexión (el valor reflejado en el registro especial SYSTEM_USER), los privilegios que mantiene el ID de autorización de la sentencia deben incluir al menos uno de los siguientes (SQLSTATE 28000):

- Autorización SYSADM o DBADM

Sintaxis:



Descripción:

nombre-autorización
Especifica el ID de autorización que se debe utilizar como el nuevo valor para el registro especial SESSION_USER.

USER
El valor del registro especial USER.

CURRENT_USER
El valor del registro especial CURRENT USER.

SYSTEM_USER
El valor del registro especial SYSTEM_USER.

variable-lengpral
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lengpral* no debe ser mayor que 30 (SQLSTATE 28000). No puede establecerse en nulo. Si una *variable-lengpral* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 28000).

Los caracteres de *variable-lengpral* deben estar justificados por la izquierda. Cuando se especifica un *nombre-autorización* con una variable del sistema principal, todos los caracteres se deben especificar con las mayúsculas y minúsculas exactas, porque no hay conversión a caracteres en mayúsculas.

constante-serie
Es una constante de tipo serie de caracteres con una longitud máxima de 30.

ALLOW ADMINISTRATION
Indica que se pueden especificar sentencias de esquema SQL antes de esta sentencia en la misma unidad de trabajo.

Normas:

- El valor especificado para el registro especial SESSION_USER que cumple con las reglas de un ID de autorización de tipo USER (SQLSTATE 42602).
- La opción de vinculación OWNER especifica el ID de autorización que se debe utilizar para sentencias de SQL estático.
- Esta sentencia sólo se puede emitir como la primera sentencia (que no sea una sentencia de registro especial SET) en una nueva unidad de trabajo sin ningún cursor WITH HOLD abierto (SQLSTATE 25001). Esta restricción incluye cualquier petición PREPARE correspondiente a una sentencia que no sea una sentencia de registro especial SET.
- El valor del registro especial SESSION_USER se utiliza como el ID de autorización para todas las sentencias de SQL dinámico de un paquete vinculado con la opción de vinculación DYNAMICRULES(RUN). (Esto incluye INVOKERUN y DEFINERUN cuando el paquete no lo utiliza ninguna rutina). Si un paquete utiliza autorización de propietario, invocador o definidor basado en la opción DYNAMICRULES, esta sentencia no tiene ningún efecto sobre las sentencias de SQL dinámico emitidas desde dentro del paquete.

Notas:

- El valor inicial del registro especial SESSION_USER para una nueva conexión es el mismo que el valor del registro especial SYSTEM_USER.
- La información de grupo correspondiente al ID de autorización de sesión especificado en esta sentencia se adquiere en el momento de ejecutar la sentencia.
- El hecho de establecer el registro especial SESSION_USER no afecta a los registros especiales CURRENT SCHEMA ni CURRENT PATH.
- Si se produce algún error durante el establecimiento del registro especial SESSION_USER, el registro vuelve a tomar su valor anterior.
- Esta sentencia no se debe utilizar para permitir que varios usuarios diferentes reutilicen la misma conexión, puesto que cada usuario heredará la capacidad de cambiar el valor del registro especial SESSION_USER. Esta sentencia depende del valor de SYSTEM_USER correspondiente a la comprobación de privilegios y la sentencia SET SESSION AUTHORIZATION no modifica el ID de autorización de conexión inicial. Además, esta sentencia no está relacionada con los siguientes comportamientos que afectan a la reutilización de la conexión:
  - El privilegio CONNECT no se comprueba para el nuevo ID de autorización
  - El contenido de cualquier registro especial actualizable no se restablece; en concreto, el contenido del registro especial ENCRYPTION PASSWORD no se modifica y está disponible para el nuevo ID de autorización para las funciones de cifrado y descifrado.
  - El contenido de cualquier tabla temporal global declarada no se ve afectado y resulta accesible para el nuevo ID de autorización

  - Cualquier enlace existente con servidores remotos no se restablecer
- Si se especifica la cláusula ALLOW ADMINISTRATION, los siguientes tipos de sentencias u operaciones pueden preceder la sentencia SET SESSION AUTHORIZATION:
  - Lenguaje de definición de datos (DDL), incluida la definición de los puntos de salvaguarda y la declaración de tablas temporales globales, pero no incluyendo SET INTEGRITY
  - Sentencias GRANT y REVOKE
  - LOCK TABLE, sentencia
  - Sentencias COMMIT y ROLLBACK
  - SET de registros especiales

Ejemplos:

Ejemplo 1: La siguiente sentencia establecer el registro especial SESSION_USER.

```
SET SESSION_USER = RAJIV
```

Ejemplo 2: Establezca el ID de autorización de sesión (el registro especial SESSION_USER) de modo que sea un valor del ID de autorización del sistema, que es el ID que ha establecido la conexión en la que se ha emitido la sentencia.

```
SET SESSION AUTHORIZATION SYSTEM_USER
```

Información relacionada:

- "Mandato BIND" en la publicación *Consulta de mandatos*

SET Variable

La sentencia SET Variable asigna valores a las variables locales, los parámetros de salida o las nuevas variables de transición. Está bajo el control de la transacción.

Invocación:

Esta sentencia sólo puede utilizarse como sentencia de SQL en una sentencia compuesta dinámica, un activador, una función de SQL, un método de SQL o un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización:

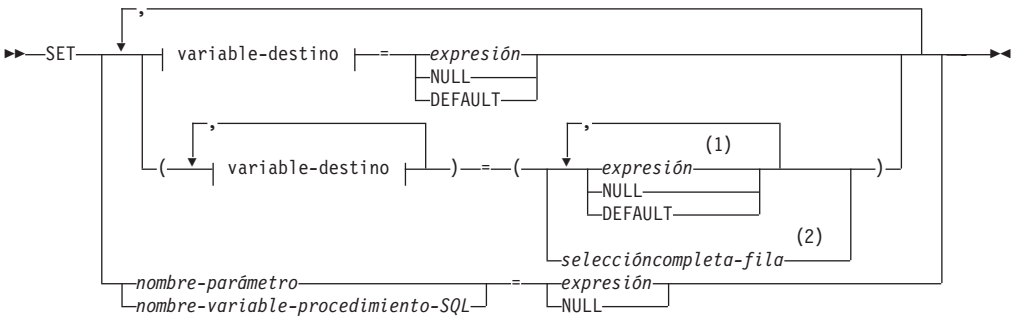
Para hacer referencia a una variable de transición, el ID de autorización del creador de activadores debe tener al menos uno de los privilegios siguientes:

- El privilegio UPDATE para las columnas a las que se hace referencia en la parte izquierda de la asignación y el privilegio SELECT para cualquiera de las columnas a las que se hace referencia en la parte derecha
- Privilegio CONTROL para la tabla (tabla sujeto del activador)
- Autorización SYSADM o DBADM.

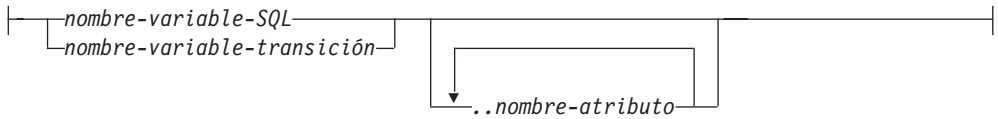
Para ejecutar esta sentencia con una *selecciónfila-completa* como parte derecha de la asignación, los privilegios contenidos en el ID de autorización del definidor del activador o del propietario de la sentencia compuesta dinámica también debe incluir como mínimo uno de los elementos siguientes, para cada tabla o vista a la que se hace referencia:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Sintaxis:



variable-destino:



Notas:

1 El número de expresiones, NULL y DEFAULT debe coincidir con el número de especificaciones de la *variable-destino*.

2 El número de columnas de la lista de selección debe coincidir con el número de especificaciones de *variable-destino*.

Descripción:

variable-destino
Identifica la variable de destino de la asignación. Una *variable-destino* que represente la misma variable no debe especificarse más de una vez (SQLSTATE 42701).

nombre-variable-SQL
Identifica la variable SQL que es el sujeto de la asignación. Las variables SQL se deben declarar antes de utilizarlas. Las variables SQL se pueden definir en sentencias compuestas dinámicas o de procedimiento.

nombre-variable-transición
Identifica la columna que se debe actualizar en la fila de transición. Un *nombre-variable-transición* debe identificar una columna en la tabla sujeto de un activador, calificado opcionalmente por un nombre de correlación que identifique el nuevo valor (SQLSTATE 42703).

..nombre de atributo
Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*).El *nombre-variable-SQL* o *nombre-variable-transición* especificado debe definirse con una tipo estructurado definido por el usuario (SQLSTATE 428DP).El *nombre-atributo* debe ser un atributo del tipo estructurado (SQLSTATE 42703).Una asignación que no incluye la cláusula *..nombre de atributo* se denomina asignación convencional.

nombre-parámetro
Identifica el parámetro que es el sujeto de la asignación. El parámetro se debe especificar en la *declaración-parámetro* de la sentencia CREATE PROCEDURE y debe estar definido como parámetro OUT o INOUT.

nombre-variable-procedimiento-SQL
Identifica la variable SQL que es el destino de la asignación dentro de un procedimiento SQL. Las variables SQL se deben declarar antes de utilizarlas. Las variables SQL se pueden definir en una sentencia compuesta de procedimiento.

expresión
Indica el nuevo valor del destino de la asignación. La expresión es cualquier expresión del tipo que se describe en el apartado “Expresiones”. La expresión no puede incluir una función de columna excepto cuando se produce dentro de una selección completa escalar (SQLSTATE 42903). En el contexto de una sentencia CREATE TRIGGER, una *expresión* puede contener referencias a las variables de transición OLD y NEW. Las variables de transición debe calificarlas el *nombre-correlación* (SQLSTATE 42702).

NULL
Especifica el valor nulo. NULL no puede ser el valor en una asignación de atributos (SQLSTATE 429B9), a menos que se convirtiera específicamente al tipo de datos del atributo.

DEFAULT
Especifica que se debe utilizar el valor por omisión.

Si *variable-destino* es una columna, el valor insertado depende de cómo se haya definido la columna en la tabla.

- Si la columna se ha definido utilizando la cláusula WITH DEFAULT, el valor se establece en el valor por omisión que se ha definido para la columna (consulte *cláusula-valor-por-omisión* en "ALTER TABLE").
- Si se utiliza la cláusula IDENTITY para definir la columna, el gestor de bases de datos genera el valor.
- Si la columna se ha definido sin especificar la cláusula WITH DEFAULT, la cláusula IDENTITY ni la cláusula NOT NULL, el valor es NULL.
- Si la columna se ha definido utilizando la cláusula NOT NULL y:
  - No se utiliza la cláusula IDENTITY o
  - No se ha utilizado la cláusula WITH DEFAULT o
  - se ha utilizado la cláusula DEFAULT NULL

  la clave DEFAULT no se puede especificar para dicha columna (SQLSTATE 23502).

Si la *variable-destino* es una variable SQL, el valor que se ha insertado es el valor por omisión, tal como se ha especificado o está implícito en la declaración de variable.

selecciónpleta-fila
Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de variables de destino especificadas para la asignación. Los valores se asignan a cada variable de destino correspondiente. Si la selección completa da como resultado ninguna fila, se asignan valores nulos. En el contexto de una sentencia CREATE TRIGGER, una *selección-completa-fila* puede contener referencias a las variables de transición OLD y NEW, que se deben calificar por su *nombre-correlación* para especificar qué variable de transición se debe utilizar (SQLSTATE 42702). Se devuelve un error si hay más de una fila en el resultado (SQLSTATE 21000).

Normas:

- El número de valores que se debe asignar a partir de las expresiones, NULL, DEFAULT o la *selección-completa-fila* debe coincidir con el número de *variables-destino* especificado para la asignación (SQLSTATE 42802).
- Una sentencia SET variable no puede asignar una variable SQL y una variable de transición en una sentencia (SQLSTATE 42997).
- Se asignan valores a las variables de destino en función de normas de asignación específicas.
- Las sentencias de asignación de los procedimientos SQL deben ajustarse a las normas de asignación del SQL. Los argumentos de serie de caracteres utilizan normas de asignación de almacenamiento.
- Si una variable se ha declarado con un identificador que coincide con el nombre de un registro especial (por ejemplo, PATH), se debe delimitar la variable para impedir una asignación no intencionada al registro especial (por ejemplo, `SET "PATH" = 1;` para una variable llamada PATH que se ha declarado como entero).

Notas:

- Si se incluye más de una asignación, se evalúa cada *expresión* y *selección-completa-fila* antes de que se realicen las asignaciones. Por lo tanto, las referencias a variables de destino en una expresión o una selección completa de fila son siempre el valor de la variable de destino antes de cualquier asignación en una única sentencia SET.

- Cuando se actualiza una columna de identidad que se ha definido como un tipo diferenciado, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior al tipo de fuente antes de realizarse el cálculo.)
- Para que DB2 genere un valor en una sentencia SET para una columna de identidad, utilice la palabra clave por omisión DEFAULT:

  ```
  SET NEW.EMPNO = DEFAULT
  ```

  En este ejemplo, NEW.EMPNO está definido como columna de identidad, y el valor utilizado para actualizar la columna es generado por DB2.
- Para obtener más información acerca de la utilización máxima de los valores de una secuencia generada para una columna de identidad y para obtener información acerca de cuándo se excede el valor máximo de una columna de identidad, consulte "INSERT".

Ejemplos:

Ejemplo 1: Establezca la columna del salario de la fila para la que se ejecuta actualmente la acción del activador en 50000.

```
SET NEW_VAR.SALARY = 50000;
```

O bien:

```
SET (NEW_VAR.SALARY) = (50000);
```

Ejemplo 2: Establezca las columnas del salario y la comisión de la fila para la que se está ejecutando actualmente la acción de activador en 50000 y 8000, respectivamente.

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
```

O bien:

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

Ejemplo 3: Establezca la columna del salario y la comisión de la fila para la que se está ejecutando actualmente la acción del activador en el salario y la comisión promedio de los empleados del departamento que está asociado a la fila actualizada.

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM)
   = (SELECT AVG(SALARY), AVG(COMM)
        FROM EMPLOYEE E
        WHERE E.WORKDEPT = NEW_VAR.WORKDEPT);
```

Ejemplo 4: Establezca la columna de salario y de comisión de la fila para la que se ejecuta actualmente la acción del activador en 10000 y el valor original del salario (es decir, antes de ejecutar la sentencia SET), respectivamente.

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
```

O bien:

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

Ejemplo 5: Incremente la variable SQL `p_salary` un 10 por ciento.

```
SET p_salary = p_salary + (p_salary * .10)
```

Ejemplo 6: Establezca la variable SQL `p_salary` en el valor nulo.

```
SET p_salary = NULL
```

Información relacionada:

- "Expresiones" en la publicación *Consulta de SQL, Volumen 1*
- "ALTER TABLE" en la página 46
- "INSERT" en la página 603
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

SIGNAL

La sentencia SIGNAL se utiliza para transmitir una condición de error o de aviso. Da lugar a que se devuelva un error o un aviso especificándose el SQLSTATE, junto con el texto de mensaje opcional.

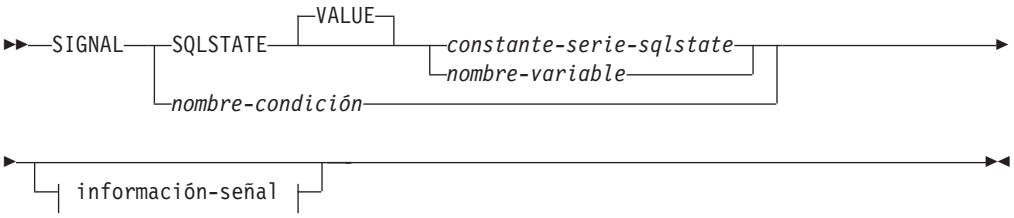
Invocación:

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

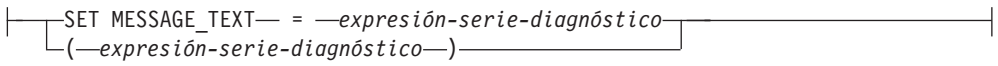
Autorización:

No se necesita.

Sintaxis:



información-señal:



Descripción:

SQLSTATE VALUE *constante-serie-sqlstate*
La constante especificada de tipo serie representa un estado SQL (SQLSTATE). Debe ser una constante de tipo serie de caracteres con exactamente 5 caracteres que siguen las normas para QLSTATEs:

- Cada carácter debe pertenecer al conjunto de dígitos ('0' a '9') o de letras mayúsculas no acentuadas ('A' a 'Z').
- La clase SQLSTATE (primeros dos caracteres) no puede ser '00', pues esto representa una finalización satisfactoria.

En el contexto de una sentencia dinámica compuesta, un activador, una función SQL o un método SQL, se deben aplicar también las normas siguientes:

- La clase SQLSTATE (dos primeros caracteres) no puede ser '01' o '02', dado que éstos no son clases de error.
- Si la clase SQLSTATE empieza con los números '0' a '6' o las letras 'A' a 'H', la subclase (los tres últimos caracteres) debe empezar con una letra en el rango de 'I' a 'Z'.
- Si la clase SQLSTATE empieza con los números '7', '8', '9' o las letras 'I' a 'Z', la subclase puede ser cualquier carácter de '0' a '9' o de 'A' a 'Z'.

Si SQLSTATE no se ajusta a estas normas, se devuelve un error (SQLSTATE 428B3).

SQLSTATE VALUE *nombre-variable*
El nombre de variable especificado debe ser de tipo CHAR(5). Su valor en tiempo de ejecución de la sentencia debe cumplir las mismas normas que se describen para *constante-serie-sqlstate*. Si SQLSTATE no se ajusta a estas normas, se devuelve un error (SQLSTATE 428B3).

nombre-condición
Especifica el nombre de la condición. El nombre de la condición debe ser exclusivo dentro del procedimiento y sólo puede estar referenciado dentro de la sentencia compuesta donde está declarado.

SET MESSAGE_TEXT =
Especifica una serie de caracteres que describe el error o aviso. La serie de caracteres se coloca en el campo SQLERRMC de la SQLCA. Si la serie tiene más de 70 bytes de longitud, se truncará sin avisar de ello.

expresión-serie-diagnóstico
Una expresión con un tipo de CHAR o VARCHAR que devuelve una serie de caracteres de hasta 70 bytes para describir la condición de error. Si la serie tiene más de 70 bytes, se truncará.

(*expresión-serie-diagnóstico***)**
Una expresión con un tipo de CHAR o VARCHAR que devuelve una serie de caracteres de hasta 70 bytes para describir la condición de error. Si la serie tiene más de 70 bytes, se truncará. Esta opción sólo se proporciona en el ámbito de una sentencia CREATE TRIGGER para compatibilidad con versiones anteriores de DB2. Se recomienda no utilizarla de forma continua.

Notas:

- Si se emite una sentencia SIGNAL, el SQLCODE que se devuelve se basa en el SQLSTATE, tal como se indica a continuación:
  - Si la clase de SQLSTATE especificada es '01' ó '02', se devuelve un aviso o una condición de no encontrado y el SQLCODE se establece en +438.
  - De otro modo, se devuelve una condición de excepción y el SQLCODE se establece en -438.

  Los demás campos de la SQLCA se establecen de la forma siguiente:
  - Los campos sqlerrd se establecen en cero.
  - Los campos sqlwarn se establecen en blancos.
  - El campo sqlerrmc se establece en los 70 primeros bytes de MESSAGE_TEXT.
  - El campo sqlerrml se establece en la longitud de sqlerrmc o bien en cero si no se ha especificado ninguna cláusula SET MESSAGE_TEXT.
  - El campo sqlerrp se establece en ROUTINE.
- Los valores de SQLSTATE constan de un código de clase formado por dos caracteres, seguido de un código de subclase formado por tres caracteres. Los códigos de clase representan condiciones de ejecución satisfactoria o errónea.

  Se puede utilizar un valor válido cualquiera de SQLSTATE en la sentencia SIGNAL. Sin embargo, es recomendable que el programador defina nuevos SQLSTATE basados en los rangos de valores reservados para las aplicaciones. Esto evita el uso involuntario de un valor de SQLSTATE que el gestor de bases de datos podría definir en un futuro release.

- Se pueden definir las clases de SQLSTATE que comienzan con los caracteres del '7' al '9', o de la 'I' a la 'Z'. Dentro de estas clases, se puede definir cualquier subclase.
- Las clases de SQLSTATE que comienzan con los caracteres del '0' al '6', o de la 'A' a la 'H' están reservadas para el gestor de bases de datos. Dentro de estas clases, las subclases que comienzan con los caracteres del '0' a la 'H' están reservadas para el gestor de bases de datos. Se pueden definir subclases que empiecen con los caracteres 'I' a 'Z'.

Ejemplos:

En el ejemplo siguiente, se utiliza un procedimiento SQL para un sistema de gestión de pedidos que señala un error de aplicación cuando la aplicación no reconoce un número de cliente. La tabla ORDERS incluye una clave foránea para la tabla CUSTOMER, lo cual hace necesario que exista el número de cliente (CUSTNO) para poder insertar un pedido.

```
CREATE PROCEDURE SUBMIT_ORDER
  (IN ONUM INTEGER, IN CNUM INTEGER,
   IN PNUM INTEGER, IN QNUM INTEGER)
   SPECIFIC SUBMIT_ORDER
   MODIFIES SQL DATA
  LANGUAGE SQL
  BEGIN
    DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
      SIGNAL SQLSTATE '75002'
        SET MESSAGE_TEXT = 'No se conoce el número del cliente';
    INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
      VALUES (ONUM, CNUM, PNUM, QNUM);
  END
```

UPDATE

La sentencia UPDATE actualiza los valores de las columnas especificadas en filas de una tabla, vista o apodo, o en las tablas, apodos o vistas subyacentes de la selección completa especificada. La actualización de una fila de una vista actualiza una fila de su tabla base, si no se ha definido ningún activador INSTEAD OF para la operación de actualización en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador. La actualización de una fila utilizando un apodo actualiza una fila del objeto de fuente de datos al que hace referencia el apodo.

Las formas de esta sentencia son:

- La forma UPDATE *Con búsqueda* se utiliza para actualizar una o varias filas (determinadas opcionalmente por la condición de búsqueda).
- La forma de UPDATE *Con posición* se utiliza para actualizar exactamente una fila (tal como determina la posición actual de un cursor).

Invocación:

Una sentencia UPDATE puede incorporarse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización:

El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- El privilegio UPDATE para la tabla, vista o apodo donde van a actualizarse filas.
- Privilegio UPDATE para cada una de las columnas que se deben actualizar.
- Privilegio CONTROL para la tabla, vista o apodo donde van a actualizarse filas.
- Autorización SYSADM o DBADM.
- Si se incluye una *selección-completa-fila* en la asignación, como mínimo, uno de los siguientes para cada tabla, vista o apodo al que se hace referencia:
  - Privilegio SELECT
  - Privilegio CONTROL
  - Autorización SYSADM o DBADM.

Para cada tabla, vista o apodo al que haga referencia una subconsulta, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM.

Si el paquete utilizado para procesar la sentencia se ha compilado previamente con normas SQL92 (opción LANGLEVEL con un valor de SQL92E o MIA) y la forma buscada de una sentencia UPDATE incluye una referencia a una columna de la tabla, vista o apodo a la derecha de la *cláusula-asignación* o en cualquier punto de la *condición-búsqueda*, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

- Privilegio SELECT
- Privilegio CONTROL

- Autorización SYSADM o DBADM.

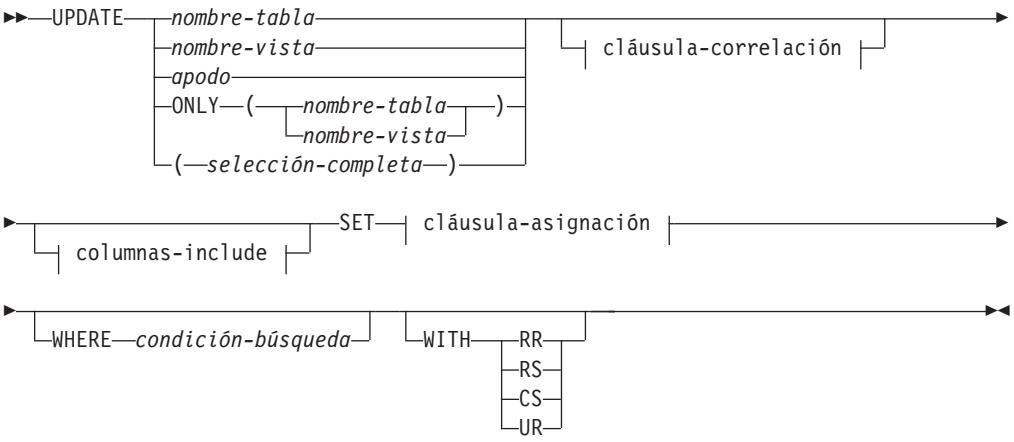
Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

No se comprueban los privilegios GROUP para las sentencias UPDATE estáticas.

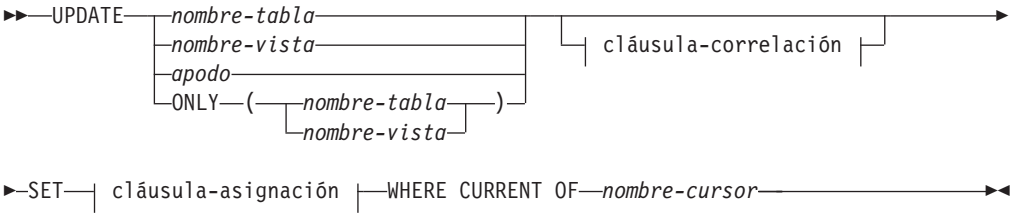
Si el destino de la operación de actualización es un apodo, los privilegios para el objeto en la fuente de datos no se consideran hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización que se ha utilizado para conectarse con la fuente de datos debe disponer de los privilegios necesarios para realizar la operación en el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis:

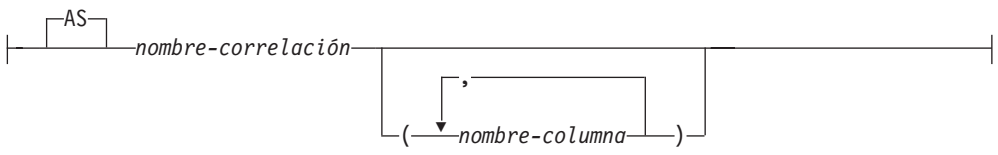
actualización-búsqueda:



actualización-posición:

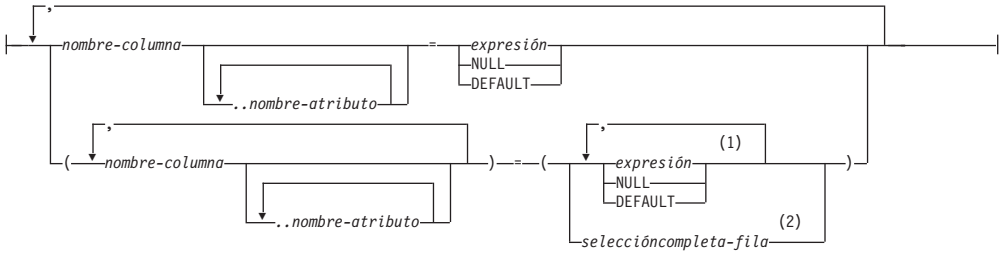


cláusula-correlación:



columnas-include:

cláusula-asignación:



Notas:

1 El número de expresiones, de NULL y de DEFAULT debe coincidir con el número de nombres de columna.

2 El número de columnas en la lista de selección debe coincidir con el número de nombres de columna.

Descripción:

nombre-tabla, *nombre-vista*, *apodo* **o** *(selección-completa)*
Identifica el objeto de la operación de actualización. El nombre debe identificar una tabla, vista o apodo que se haya descrito en el catálogo, pero no una tabla de catálogo, una vista de una tabla de catálogo (a menos que sea una de las vistas SYSSTAT que pueden actualizarse), una tabla de consultas materializadas mantenida por el sistema o una vista de sólo lectura que no tenga definido ningún activador INSTEAD OF para sus operaciones de actualización.

Si *nombre-tabla* es una tabla con tipo, la sentencia puede actualizar filas de la tabla o cualquiera de sus subtablas correspondientes. Sólo pueden establecerse o referirse las columnas de la tabla especificada en la cláusula WHERE. Para una sentencia UPDATE con posición, el cursor asociado también debe especificar la misma tabla, vista o apodo en la cláusula FROM sin utilizar ONLY.

Si el objeto de la operación de actualización es una selección completa, esta debe ser actualizable, según lo definido en el elemento de Notas "Vistas actualizables" de la descripción de la sentencia CREATE VIEW.

ONLY (*nombre-tabla***)**
Aplicable a las tablas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la tabla especificada y no puede actualizar las filas de las subtablas correspondientes. Para una sentencia UPDATE con posición, el cursor asociado también debe haber especificado la tabla en la cláusula FROM utilizando ONLY. Si *nombre-tabla* no es una tabla con tipo, la palabra clave ONLY no tiene efecto en la sentencia.

ONLY (*nombre-vista***)**
Aplicable a las vistas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la vista especificada y no puede actualizar las filas de las subvistas correspondientes. Para una sentencia

UPDATE con posición, el cursor asociado también debe haber especificado la vista en la cláusula FROM utilizando ONLY. Si *nombre-vista* no es una vista con tipo, la palabra clave ONLY no tiene efecto en la sentencia.

cláusula-correlación
Se puede utilizar dentro de la *condición-búsqueda* o *cláusula-asignación* para designar una tabla, vista, apodo o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte "referencia-tabla" en la descripción de "Subselección".

columnas-include
Especifica un conjunto de columnas que se incluyen, junto con las columnas de *nombre-tabla* o *nombre-vista*, en la tabla de resultados intermedia de la sentencia UPDATE cuando está anidada en la cláusula FROM de una selección completa. Las *columnas-include* se añaden al final de la lista de columnas especificadas para *nombre-tabla* o *nombre-vista*.

INCLUDE
Especifica una lista de columnas que se van a incluir en la tabla de resultados intermedia de la sentencia UPDATE.

nombre-columna
Especifica una columna de la tabla de resultados intermedia de la sentencia UPDATE. El nombre no puede coincidir con el nombre de otra columna include ni de una columna en *nombre-tabla* o *nombre-vista* (SQLSTATE 42711).

tipo-datos
Especifica el tipo de datos de la columna include. El tipo de datos debe ser uno que reciba soporte de la sentencia CREATE TABLE.

SET
Introduce la asignación de valores a nombres de columna.

cláusula-asignación

nombre-columna
Identifica una columna que se debe actualizar. El *nombre-columna* debe identificar una columna actualizable de la tabla, vista o apodo especificado o debe identificar una columna INCLUDE. La columna de ID de objeto de una tabla con tipo no es actualizable (SQLSTATE 428DZ). Una columna no debe especificarse más de una vez, a no ser que vaya seguida de *..nombre-atributo* (SQLSTATE 42701).

Si especifica una columna INCLUDE, el nombre de columna no se puede cualificar.

Para una sentencia UPDATE con posición:

- Si se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor, cada nombre de columna de la *cláusula-asignación* también debe aparecer en la *cláusula-actualización*.
- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL MIA o SQL92E al compilarse previamente la aplicación, puede especificarse el nombre de cualquier columna que pueda actualizarse.
- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL SAA1 explícitamente o por omisión al compilarse previamente la aplicación, no puede actualizarse ninguna columna.

.. nombre-atributo
Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*). El *nombre-columna* especificado se debe definir con un tipo estructurado definido por el usuario (SQLSTATE 428DP). El nombre de atributo debe ser un atributo del tipo estructurado del *nombre-columna* (SQLSTATE 42703). Las asignaciones en las que no interviene la cláusula *..nombre-atributo* se denominan *asignaciones convencionales*.

expresión
Indica el nuevo valor de la columna. La expresión es cualquier expresión del tipo que se describe en el apartado "Expresiones". La expresión no puede incluir una función de columna excepto cuando se produce dentro de una selección completa escalar (SQLSTATE 42903).

Una *expresión* puede contener referencias a las columnas de la tabla de destino de la sentencia UPDATE. Para cada fila que se actualiza, el valor de dicha columna en una expresión es el valor de la columna en la fila antes de que se actualice la fila.

Una expresión no puede contener referencias a una columna INCLUDE.

NULL
Especifica el valor nulo y sólo se puede especificar para las columnas que pueden contener nulos (SQLSTATE 23502). NULL no puede ser el valor en una asignación de atributos (SQLSTATE 429B9), a menos que se convirtiera específicamente al tipo de datos del atributo.

DEFAULT
Especifica que debe utilizarse el valor por omisión basándose en cómo se define la columna correspondiente en la tabla. El valor que se inserta depende de cómo se ha definido la columna.

- Si la columna se definió como columna generada basándose en una expresión, el sistema genera el valor de la columna de acuerdo con esa expresión.
- Si se utiliza la cláusula IDENTITY para definir la columna, el gestor de bases de datos genera el valor.
- Si la columna se ha definido utilizando la cláusula WITH DEFAULT, el valor se establece en el valor por omisión que se ha definido para la columna (consulte *cláusula-valor-por-omisión* en "ALTER TABLE").
- Si para definir la columna se utiliza la cláusula NOT NULL y no la cláusula GENERATED, o no se utiliza WITH DEFAULT o se utiliza DEFAULT NULL, no se puede especificar la palabra clave DEFAULT para esa columna (SQLSTATE 23502).

El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).

La palabra clave DEFAULT no se puede utilizar como valor en una asignación de atributos (SQLSTATE 429B9).

La palabra clave DEFAULT no puede utilizarse como valor en una asignación para realizar una actualización en un apodo donde la fuente de datos no da soporte a la sintaxis DEFAULT.

selecciónfila-completa
Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de *nombres-columna* especificados para

la asignación. Los valores se asignan a cada *nombre-columna* correspondiente. Si el resultado de la *selecciónconpleta-fila* es ninguna fila, se asignan los valores nulos.

Una *selecciónconpleta-fila* puede contener referencias a las columnas de la tabla de destino de la sentencia UPDATE. Para cada fila que se actualiza, el valor de dicha columna en una expresión es el valor de la columna en la fila antes de que se actualice la fila. Se devuelve un error si hay más de una fila en el resultado (SQLSTATE 21000).

WHERE
Introduce una condición que indica qué filas se actualizan. Puede omitir la cláusula, proporcionar una condición de búsqueda o nombrar un cursor. Si se omite la cláusula, se actualizan todas las filas de la tabla, vista o apodo.

condición-búsqueda
Cada *nombre-columna* de la condición de búsqueda, a excepción de en una subconsulta, debe especificar el nombre de una columna de la tabla, vista o apodo. Cuando la condición de búsqueda incluye una subconsulta en la que la misma tabla es el objeto base de UPDATE y de la subconsulta, la subconsulta se evalúa completamente antes de que se actualice cualquier fila.

La condición-búsqueda se aplica a cada fila de la tabla, vista o apodo y las filas actualizadas son las filas para las que el resultado de la condición-búsqueda es verdadero.

Si la condición de búsqueda contiene una subconsulta, la subconsulta puede considerarse como ejecutada cada vez que la condición de búsqueda se aplica a una fila y el resultado se utiliza en la aplicación de la condición de búsqueda. En realidad, una subconsulta sin referencias correlacionadas se ejecuta una sola vez, mientras que una subconsulta con una referencia correlacionada puede tener que ejecutarse una vez para cada fila.

CURRENT OF *nombre-cursor*
Identifica el cursor que se debe utilizar en la operación de actualización. El *nombre-cursor* debe identificar un cursor declarado, que se explica en "DECLARE CURSOR". La sentencia DECLARE CURSOR debe preceder a la sentencia UPDATE en el programa.

La tabla, vista o apodo que se ha especificado también debe indicarse en la cláusula FROM de la sentencia SELECT del cursor, y la tabla de resultados del cursor no debe ser de sólo lectura. (Para obtener información acerca de las tablas de resultados de sólo lectura, consulte "DECLARE CURSOR".)

Cuando se ejecuta la sentencia UPDATE, el cursor debe posicionarse en una fila; dicha fila se actualiza.

Esta forma de UPDATE no se puede utilizar (SQLSTATE 42828) si el cursor hace referencia a:

- Una vista en la que se ha definido un activador INSTEAD OF UPDATE
- Una vista que incluye una función OLAP en la lista de la selección completa que define la vista
- Una vista que se ha definido, directa o indirectamente, utilizando la cláusula WITH ROW MOVEMENT

WITH
Especifica el nivel de aislamiento en el que se ejecuta la sentencia UPDATE.

RR
Lectura repetible

RS
Estabilidad de lectura

CS
Estabilidad del cursor

UR
Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia.

Normas:

- ***Activadores:*** Las sentencias UPDATE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores de la actualización. Si una operación de actualización en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se realizan en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.
- ***Asignación***: Los valores de actualización se asignan a las columnas según las normas de asignación específicas.
- ***Validez***: La fila actualizada se debe ajustar a cualquier restricción impuesta en la tabla (o en la tabla base de la vista) por cualquier índice exclusivo de una columna actualizada.

  Si se utiliza una vista que no se ha definido utilizando WITH CHECK OPTION, se pueden cambiar las filas para que no se ajusten más a la definición de la vista. Dichas filas se actualizan en la tabla base de la vista y ya no aparecen más en la vista.

  Si se utiliza una vista que se ha definido utilizando WITH CHECK OPTION, una fila actualizada debe ajustarse a la definición de la vista. Para obtener información acerca de las normas que rigen esta situación, consulte "CREATE VIEW".
- ***Restricción de comprobación***: El valor de actualización debe satisfacer las condiciones de comprobación de las restricciones de comprobación definidas en la tabla.

  En UPDATE para una tabla con restricciones de comprobación definidas se evalúan las condiciones de restricción para cada columna una vez para cada fila que se actualiza. Cuando se procesa una sentencia UPDATE, sólo se comprueban las restricciones de comprobación que hacen referencia a las columnas actualizadas.
- ***Integridad de referencia***: El valor de las claves exclusivas padre no se pueden cambiar si la norma de actualización es RESTRICT y hay una o varias filas dependientes. Sin embargo, si la norma de actualización es NO ACTION, las claves de unicidad padres pueden actualizarse siempre que cada hijo tenga una clave padre en el momento en que se completa la sentencia de actualización. Un valor de actualización no nulo para una clave foránea debe ser igual a un valor de la clave primaria de la tabla padre de la relación.

Notas:

- Si un valor de actualización viola cualquier restricción o si se produce cualquier otro error durante la ejecución de la sentencia UPDATE, no se actualiza ninguna fila. El orden en que se actualizan múltiples filas no está definido.

- Una actualización de una vista definida utilizando la cláusula WITH ROW MOVEMENT puede ocasionar una operación de supresión y una operación de inserción contra las tablas subyacentes de la vista. Para ver detalles, consulte la descripción de la sentencia CREATE VIEW.
- Cuando una sentencia UPDATE completa su ejecución, el valor de SQLERRD(3) de la SQLCA es el número de filas que se han calificado para la operación de actualización. En el contexto de una sentencia de procedimiento de SQL, el valor puede recuperarse utilizando la variable ROW_COUNT de la sentencia GET DIAGNOSTICS. El campo SQLERRD(5) contiene el número de filas insertadas, suprimidas o actualizadas por todos los activadores activados.
- A menos que ya existan los bloqueos adecuados, se adquieren uno o varios bloqueos por la ejecución de una sentencia UPDATE satisfactoria. Hasta que se liberan los bloqueos, la fila actualizada sólo puede accederse por el proceso de aplicación que ha realizado la actualización (excepto las aplicaciones que utilizan el nivel de aislamiento de Lectura no confirmada). Para obtener más información sobre el bloqueo, consulte las descripciones de las sentencias COMMIT, ROLLBACK y LOCK TABLE.
- Si se actualiza el valor del URL de una columna DATALINK, es lo mismo que suprimir el valor DATALINK antiguo e insertar el nuevo. En primer lugar, si el valor antiguo estaba enlazado con un archivo, este archivo se desenlaza. A continuación, a no ser que los atributos de enlace del valor DATALINK estén vacíos, el archivo especificado se enlaza con esta columna. La única excepción a esto es que si el URL del valor de DATALINK antiguo es *idéntico* al URL del valor de DATALINK existente, no es necesario comunicarse con el Data Links Manager asociado para eliminar el enlace y volver a enlazar el mismo archivo. En esta situación, la actividad general se elimina por completo.

  El valor del comentario de una columna DATALINK puede actualizarse sin volver a enlazar el archivo si se especifica una serie vacía como vía de acceso de URL (por ejemplo, como argumento *ubicación-datos* de la función escalar DLVALUE o si se especifica el valor nuevo de manera que sea el mismo que el valor antiguo).

  Si una columna DATALINK se actualiza con un valor nulo, es lo mismo que suprimir el valor DATALINK existente.

  Puede producirse un error cuando se intente actualizar un valor DATALINK si el servidor de archivos del valor existente o del valor nuevo ya no está registrado con el servidor de bases de datos (SQLSTATE 55022).
- Cuando se actualicen las estadísticas de distribución de columna para una tabla con tipo, debe especificarse la subtabla que haya introducido primero la columna.
- Puede haber varias asignaciones de atributos en la misma columna de tipo estructurado, en el orden especificado por la cláusula SET.
- La asignación de atributos invoca el método mutador para el atributo del tipo estructurado definido por el usuario. Por ejemplo, la asignación `st..a1=x` tiene el mismo efecto que utilizar el método mutador en la asignación `st = st..a1(x)`.
- Aunque una columna determinada sólo puede ser objeto de una sola asignación convencional, puede intervenir en varias asignaciones de atributos (pero únicamente si no es objeto de una asignación convencional).
- Cuando se actualiza una columna de identidad que se ha definido como un tipo diferenciado, todo el cálculo tiene lugar en el tipo de fuente y el resultado se convierte al tipo diferenciado antes de que el valor se asigne realmente a la columna. (No se produce ninguna conversión del valor anterior al tipo de fuente antes de realizarse el cálculo.)

- Para que DB2 genere un valor en una sentencia SET para una columna de identidad, utilice la palabra clave DEFAULT:

  ```
  SET NEW.EMPNO = DEFAULT
  ```

  En este ejemplo, NEW.EMPNO está definido como columna de identidad, y el valor utilizado para actualizar la columna es generado por DB2.
- Para obtener más información acerca de la utilización máxima de los valores de una secuencia generada para una columna de identidad o acerca de cuándo se excede el valor máximo de una columna de identidad, consulte "INSERT".

Ejemplos:

- *Ejemplo 1:* Cambie el trabajo (JOB) del empleado número (EMPNO) '000290' en la tabla EMPLOYEE por 'LABORER'.

  ```
  UPDATE EMPLOYEE
  SET JOB = 'LABORER'
     WHERE EMPNO = '000290'
  ```

- *Ejemplo 2:* Aumente las personas que trabajan en el proyecto (PRSTAFF) en 1,5 para todos los proyectos de los cuales sea responsable el departamento (DEPTNO) 'D21' en la tabla PROJECT.

  ```
  UPDATE PROJECT
  SET PRSTAFF = PRSTAFF + 1,5
     WHERE DEPTNO = 'D21'
  ```

- *Ejemplo 3:* Todos los empleados excepto el director del departamento (WORKDEPT) 'E21' se han vuelto a asignar temporalmente. Indique esto cambiando su trabajo (JOB) por NULL y los valores de su paga (SALARY, BONUS, COMM) a cero en la tabla EMPLOYEE.

  ```
  UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
  ```

  Esta sentencia también se podría escribir de la siguiente manera.

  ```
  UPDATE EMPLOYEE
    SET (JOB, SALARY, BONUS, COMM) = (NULL, 0, 0, 0)
    WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
  ```

- *Ejemplo 4:* Actualice la columna del salario y comisión del empleado con el número de empleado 000120 por el promedio del salario y la comisión de los empleados del departamento de la fila actualizada respectivamente.

  ```
  UPDATE (SELECT SALARY,
      COMM,
      AVG(SALARY) OVER (PARTITION BY WORKDEPT),
      AVG(COMM) OVER (PARTITION BY WORKDEPT)
    FROM EMPLOYEE)
    AS E(SALARY, COMM, AVGSAL, AVGCOMM)
    SET (SALARY, COMM)
      = (AVGSAL, AVGCOMM)
    WHERE EU.EMPNO = '000120'
  ```

  La sentencia anterior es semánticamente equivalente a la sentencia siguiente, pero sólo necesita un acceso a la tabla EMPLOYEE, mientras que la sentencia siguiente especifica la tabla EMPLOYEE dos veces.

  ```
  UPDATE EMPLOYEE EU
    SET (EU.SALARY, EU.COMM)
    =
  (SELECT AVG(ES.SALARY), AVG(ES.COMM)
    FROM EMPLOYEE ES
    WHERE ES.WORKDEPT = EU.WORKDEPT)
    WHERE EU.EMPNO = '000120'
  ```

- *Ejemplo 5:* En un programa C visualice las filas de la tabla EMPLOYEE y, después, si se le pide hacerlo, cambie el trabajo (JOB) de algunos empleados por el nuevo trabajo escrito.

```
EXEC SQL  DECLARE C1 CURSOR FOR
              SELECT *
                FROM EMPLOYEE
                FOR UPDATE OF JOB;

EXEC SQL  OPEN C1;

EXEC SQL  FETCH C1 INTO ...     ;
if ( strcmp (change, "YES") == 0 )
  EXEC SQL  UPDATE EMPLOYEE
              SET JOB = :newjob
              WHERE CURRENT OF C1;

EXEC SQL  CLOSE C1;
```

- *Ejemplo 6:* Mutación de atributos de objetos de columnas.

  Sean los siguientes tipos y tablas:

```
CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE
```

  El ejemplo siguiente actualiza la tabla CIRCLES cambiando la columna OWNER y el atributo RADIUS de la columna CIRCLE cuyo ID es 999:

```
UPDATE CIRCLES
  SET OWNER = 'Bruce'
    C..RADIUS = 5
  WHERE ID = 999
```

  El ejemplo siguiente traslada las coordenadas X e Y del centro del círculo identificado por 999:

```
UPDATE CIRCLES
  SET C..CENTER..X = C..CENTER..Y,
    C..CENTER..Y = C..CENTER..X
  WHERE ID = 999
```

  El ejemplo siguiente constituye otro modo de escribir las dos sentencias anteriores. Este ejemplo combina los efectos de ambos ejemplos anteriores:

```
UPDATE CIRCLES
  SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
    ('Bruce',5,C..CENTER..Y,C..CENTER..X)
  WHERE ID = 999
```

Información relacionada:

- "Expresiones" en la publicación *Consulta de SQL, Volumen 1*
- "Condiciones de búsqueda" en la publicación *Consulta de SQL, Volumen 1*
- "Subselección" en la publicación *Consulta de SQL, Volumen 1*
- "ALTER TABLE" en la página 46
- "CREATE VIEW" en la página 466
- "DECLARE CURSOR" en la página 483
- "INSERT" en la página 603
- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "tbmod.sqc -- How to modify table data (C)"
- "tut_mod.sqc -- How to modify table data (C)"
- "dtstruct.sqC -- Create, use, drop a hierarchy of structured types and typed tables (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "tbmod.sqC -- How to modify table data (C++)"
- "tut_mod.sqC -- How to modify table data (C++)"
- "SpServer.java -- Provide a variety of types of stored procedures to be called from (JDBC)"
- "TbMod.java -- How to modify table data (JDBC)"
- "TutMod.java -- Modify data in a table (JDBC)"
- "SpServer.sqlj -- Provide a variety of types of stored procedures to be called from (SQLj)"
- "TbMod.sqlj -- How to modify table data (SQLj)"
- "TutMod.sqlj -- Modify data in a table (SQLj)"
- "tbmod.c -- How to modify table data"
- "tut_mod.c -- How to modify table data"
- "updat.sqb -- How to update, delete and insert table data (MF COBOL)"
- "varinp.sqb -- How to update table data using parameter markers (MF COBOL)"

VALUES

La sentencia VALUES es una forma de consulta. Puede incorporarse en un programa de aplicación o emitirse interactivamente.

Información relacionada:

- "Selección completa" en la publicación *Consulta de SQL, Volumen 1*

Ejemplos relacionados:

- "dtlob.c -- How to read and write LOB data"
- "dtlob.sqc -- How to use the LOB data type (C)"
- "fnuse.sqc -- How to use built-in SQL functions (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "dtlob.sqC -- How to use the LOB data type (C++)"
- "fnuse.sqC -- How to use built-in SQL functions (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "lobloc.sqb -- Demonstrates the use of LOB locators (MF COBOL)"

VALUES INTO

La sentencia VALUES INTO produce una tabla de resultados que consta de como máximo una fila y asigna los valores de dicha fila a las variables del lenguaje principal.

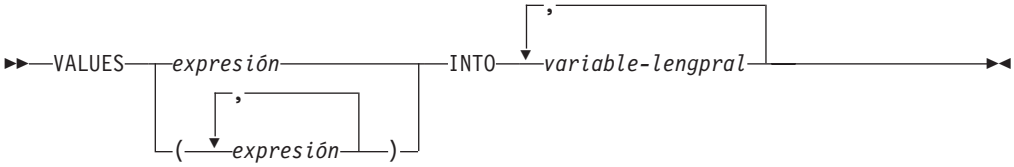
Invocación:

Esta sentencia sólo se puede incluir en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización:

No se necesita.

Sintaxis:



Descripción:

VALUES
Introduce una sola fila que consta de una o varias columnas.

expresión
Una expresión que define un solo valor de una tabla de resultados de una columna.

(expresión,...)
Una o más expresiones que definen los valores para una o varias columnas de la tabla de resultados.

INTO
Introduce una lista de variables del lenguaje principal.

variable-lengpral
Identifica una variable que está descrita en el programa bajo las normas para la declaración de variables del lenguaje principal.

El primer valor de la fila del resultado se asigna a la primera variable de la lista, el segundo valor a la segunda variable, etcétera. Si el número de variables del lenguaje principal es menor que el número de valores de columna, se asigna el valor 'W' al campo SQLWARN3 de la SQLCA.

Cada asignación que se realiza para una variable se realiza secuencialmente y siguiendo la lista. Si se produce un error, no se asigna ningún valor a la variable del lenguaje principal.

Ejemplos:

Ejemplo 1: Este ejemplo C recupera el valor del registro especial CURRENT PATH en una variable del lenguaje principal.

```
EXEC SQL VALUES(CURRENT PATH)
         INTO :hvl;
```

Ejemplo 2: Este ejemplo C recupera una parte de un campo LOB en una variable del lenguaje principal, utilizando el localizador de LOB para la recuperación diferida.

```
EXEC SQL VALUES (substr(:locator1,35))
         INTO :details;
```

Información relacionada:

- "SQLCA (área de comunicaciones SQL)" en la publicación *Consulta de SQL, Volumen 1*
- "Asignaciones y comparaciones" en la publicación *Consulta de SQL, Volumen 1*

WHENEVER

La sentencia WHENEVER especifica la acción que se debe tomar cuando se produce una condición de excepción especificada.

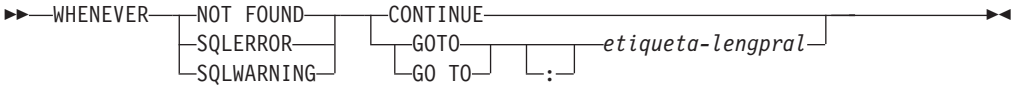
Invocación:

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. La sentencia no está soportada en REXX.

Autorización:

No se necesita.

Sintaxis:



Descripción:
Las cláusulas NOT FOUND, SQLERROR o SQLWARNING se utilizan para identificar el tipo de condición de excepción.

NOT FOUND
Identifica cualquier condición que dé como resultado un SQLCODE de +100 o un SQLSTATE de '02000'.

SQLERROR
Identifica cualquier condición que dé como resultado un SQLCODE negativo.

SQLWARNING
Identifica cualquier condición que dé como resultado una condición de aviso (SQLWARN0 es 'W'), o que dé como resultado un código de retorno SQL positivo que no sea +100.

Las cláusulas CONTINUE o GO TO se utilizan para especificar lo que ocurre cuando existe el tipo de condición de excepción identificado.

CONTINUE
Provoca que se ejecute la siguiente instrucción secuencial del programa fuente.

GOTO o **GO TO** *etiqueta-lengpral*
Provoca que el control pase a la sentencia identificada por la etiqueta-lengpral. Para etiqueta-lengpral, utilice un símbolo individual, precedido opcionalmente por dos puntos. El formato del símbolo depende del lenguaje principal.

Notas:

Hay tres tipos de sentencias WHENEVER:

- WHENEVER NOT FOUND
- WHENEVER SQLERROR
- WHENEVER SQLWARNING

Cada sentencia de SQL ejecutable en un programa está dentro del ámbito de una sentencia WHENEVER implícita o explícita de cada tipo. El ámbito de una sentencia WHENEVER está relacionado con la secuencia de listado de las sentencias del programa, no con su secuencia de ejecución.

Una sentencia de SQL está dentro del ámbito de la última sentencia WHENEVER de cada tipo que se especifica antes de la sentencia de SQL en el programa fuente. Si una sentencia WHENEVER de cualquier tipo no se especifica antes de una sentencia de SQL, dicha sentencia de SQL está dentro del ámbito de una sentencia WHENEVER de dicho tipo en la que se especifica CONTINUE.

Ejemplo:

En el ejemplo C siguiente, si se produce un error, vaya a HANDLERR. Si se produce un código de aviso, continúe con el flujo normal del programa. Si no se devuelven datos, vaya a ENDDATA.

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLERR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA;
```

Ejemplos relacionados:

- "outsrv.sqb -- Demonstrates stored procedures using the SQLDA structure (MF COBOL)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"

WHILE

La sentencia WHILE repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

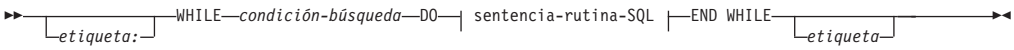
Invocación:

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

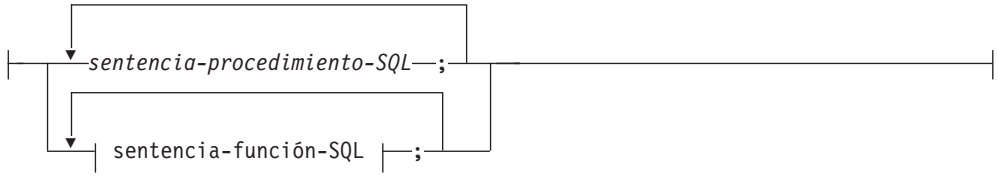
Autorización:

No se requieren privilegios para invocar una sentencia WHILE. Sin embargo, el ID de autorización de la sentencia debe tener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia WHILE.

Sintaxis:



sentencia-rutina-SQL:



Descripción:

etiqueta
Especifica la etiqueta de la sentencia WHILE. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

condición-búsqueda
Especifica una condición que se evalúa antes de cada ejecución del bucle. Si la condición es verdadera, se procesan las sentencias de procedimiento SQL incluidas en el bucle.

sentencia-procedimiento-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte la sentencia-procedimiento-SQL en la descripción de la sentencia de SQL compuesto (procedimiento).

sentencia-función-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-función-SQL* sólo se puede aplicar en el contexto de una función de SQL o de un método de SQL. Consulte sentencia-función-SQL en la descripción de la sentencia FOR.

Ejemplos:

Este ejemplo utiliza una sentencia WHILE para ejecutar un proceso iterativo mediante sentencias FETCH y SET. Mientras el valor de la variable de SQL *v_counter* sea menor que la mitad del número de empleados del departamento que identifica el parámetro IN *deptNumber*, la sentencia WHILE seguirá realizando las sentencias FETCH y SET. Cuando la condición deja de ser verdadera, el flujo de control sale de la sentencia WHILE y cierra el cursor.

```
CREATE PROCEDURE DEPT_MEDIAN
  (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
  LANGUAGE SQL
  BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
      SELECT CAST(salary AS DOUBLE)
        FROM staff
        WHERE DEPT = deptNumber
        ORDER BY salary;
    DECLARE EXIT HANDLER FOR NOT FOUND
      SET medianSalary = 6666;
    SET medianSalary = 0;
    SELECT COUNT(*) INTO v_numRecords
      FROM staff
      WHERE DEPT = deptNumber;
    OPEN c1;
    WHILE v_counter < (v_numRecords / 2 + 1) DO
      FETCH c1 INTO medianSalary;
      SET v_counter = v_counter + 1;
    END WHILE;
    CLOSE c1;
  END
```

Información relacionada:

- "SQL compuesto (procedimiento)" en la página 139

Ejemplos relacionados:

- "dbinline.sqc -- How to use inline SQL Procedure Language (C)"

Apéndice A. Información técnica sobre DB2 Universal Database

Documentación y ayuda de DB2

Está disponible información técnica de DB2® a través de las herramientas y los métodos siguientes:

- Centro de información de DB2
  - Temas
  - Herramientas de ayuda para DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Archivos PDF descargables y en CD y manuales impresos
  - Guías
  - Manuales de consulta
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes
  - Ayuda para estados de SQL
- Código fuente instalado
  - Programas de ejemplo

Puede acceder a información técnica adicional de DB2 Universal Database™ como, por ejemplo, notas técnicas, white papers y Redbooks™ en línea en ibm.com®. Acceda al sitio de la biblioteca de software de gestión de información de DB2 en www.ibm.com/software/data/pubs/.

Actualizaciones de la documentación de DB2

De forma periódica, IBM® puede realizar FixPaks de la documentación y otras actualizaciones de la misma en el Centro de información de DB2 disponible. Si accede al Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2help/, siempre visualizará la información más actualizada. Si ha instalado el Centro de información de DB2 localmente, tendrá que instalar cualquier actualización de forma manual para poder visualizarla. Las actualizaciones de la documentación le permiten actualizar la información que ha instalado desde el *CD del Centro de información de DB2* cuando está disponible nueva información.

El Centro de información se actualiza con mayor frecuencia que los manuales PDF o en copia impresa. Para conseguir la información técnica de DB2 más actualizada, instale las actualizaciones de la documentación a medida que estén disponibles o diríjase al Centro de información de DB2 en el sitio www.ibm.com.

Conceptos relacionados:

- "CLI sample programs" en la publicación *CLI Guide and Reference, Volume 1*
- "Programas de ejemplo Java" en la publicación *Guía de desarrollo de aplicaciones: Creación y ejecución de aplicaciones*
- "Centro de información de DB2" en la página 780

Tareas relacionadas:

- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 798
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 800
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 800
- "Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos" en la página 801

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 792

Centro de información de DB2

El Centro de información de DB2® le proporciona acceso a toda la información que necesita para obtener el máximo provecho de los productos de la familia de DB2, incluidos DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator y DB2 Query Patroller™. El Centro de información de DB2 también contiene información relativa a las características y los componentes principales de DB2, como la duplicación, el depósito de datos y DB2 Extenders.

El Centro de información de DB2 presenta las características siguientes si se visualiza en Mozilla 1.0 o posterior o bien en Microsoft® Internet Explorer 5.5 o posterior. Algunas características requieren que se habilite el soporte de JavaScript™:

Opciones flexibles de instalación
Puede elegir visualizar la documentación de DB2 utilizando la opción que mejor se ajuste a sus necesidades:

- Para asegurarse fácilmente de que la documentación siempre esté actualizada, puede acceder a toda la documentación directamente desde el Centro de información de DB2 incluido en el sitio Web de IBM® de http://publib.boulder.ibm.com/infocenter/db2help/
- Para minimizar el esfuerzo de actualización y mantener el tráfico de red en su intranet, puede instalar la documentación de DB2 en un solo servidor de la intranet
- Para maximizar la flexibilidad y reducir la dependencia de las conexiones de red, puede instalar la documentación de DB2 en su propio sistema

Búsqueda
Es posible buscar en todos los temas del Centro de información de DB2 entrando un término de búsqueda en el campo de texto **Buscar**. Puede recuperar coincidencias exactas encerrando los términos entre comillas y puede afinar la búsqueda mediante operadores de comodín (*, ?) y operadores booleanos (AND, NOT, OR).

Tabla de contenido orientada a tareas
Puede localizar los temas en la documentación de DB2 a partir de una sola tabla de contenido. La tabla de contenido está organizada principalmente

según la clase de tareas que puede desear realizar, pero también incluye entradas para visiones generales de productos, objetivos, información de consulta, un índice y un glosario.

- Las visiones generales de los productos describen la relación entre los productos disponibles en la familia de DB2, las características que ofrece cada uno de estos productos y proporcionan información actualizada del release de cada uno de estos productos.
- Las categorías de objetivos, como la instalación, la administración y el desarrollo, incluyen temas que permiten realizar rápidamente tareas y desarrollar un conocimiento más profundo de la información de fondo para realizar dichas tareas.
- Los temas de consulta proporcionan información detallada sobre un tema, incluida la sintaxis de sentencias y mandatos, la ayuda de mensajes y los parámetros de configuración.

Mostrar el tema actual en la tabla de contenido
Puede mostrar dónde encaja el tema actual en la tabla de contenido pulsando el botón **Renovar / Mostrar tema actual** en el marco de la tabla de contenido o pulsando el botón **Mostrar en tabla de contenido** en el marco del contenido. Esta característica es útil si ha seguido varios enlaces con temas relacionados en varios archivos o ha llegado a un tema a partir de resultados de una búsqueda.

Índice Es posible acceder a toda la documentación desde el índice. El índice está organizado en orden alfabético por términos del índice.

Glosario
Puede utilizar el glosario a fin de buscar definiciones de términos utilizados en la documentación de DB2. El glosario está organizado en orden alfabético por términos del glosario.

Información adaptada integrada
El Centro de información de DB2 visualiza la información en el idioma preferido que se ha establecido en las preferencias de navegador. Si un tema no está disponible en el idioma preferido del usuario, el Centro de información de DB2 visualiza la versión inglesa de ese tema.

Si desea información técnica sobre iSeries™, consulte el centro de información de IBM eServer™ iSeries en www.ibm.com/eserver/iseries/infocenter/.

Conceptos relacionados:

- "Escenarios de instalación del Centro de información de DB2" en la página 782

Tareas relacionadas:

- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 791
- "Invocación del Centro de información de DB2" en la página 789
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 784
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 787

Escenarios de instalación del Centro de información de DB2

Los entornos de trabajo distintos pueden plantear requisitos distintos para el modo de acceder a la información de DB2®. Se puede acceder al Centro de información de DB2 en el sitio Web de IBM®, en un servidor de la red de la organización o en una versión instalada en el sistema. En los tres casos, la documentación está incluida en el Centro de información de DB2, el cual consiste en una Web estructurada de información que se organiza en temas y que se visualiza mediante un navegador. Por omisión, los productos de DB2 acceden al Centro de información de DB2 en el sitio Web de IBM. No obstante, si desea acceder al Centro de información de DB2 en un servidor de intranet o en su propio sistema, es necesario que instale el Centro de información de DB2 utilizando el CD del Centro de información de DB2 que encontrará en el Paquete de soportes del producto. Consulte el siguiente resumen de opciones para acceder a la documentación de DB2, junto con los tres escenarios de instalación, como ayuda para determinar qué método de acceso al Centro de información de DB2 le funciona mejor en su entorno de trabajo y qué cuestiones relacionadas con la instalación se pueden tener en cuenta.

Resumen de opciones para acceder a la documentación de DB2:

La siguiente tabla proporciona recomendaciones sobre las opciones que son posibles en su entorno de trabajo a la hora de acceder a la documentación de productos de DB2 del Centro de información de DB2.

Acceso a Internet	Acceso a Intranet	Recomendación
Sí	Sí	Acceda al Centro de información de DB2 en el sitio Web de IBM o acceda al Centro de información de DB2 instalado en un servidor de intranet.
Sí	No	Acceda al Centro de información de DB2 en el sitio Web de IBM.
No	Sí	Acceda al Centro de información de DB2 instalado en un servidor de intranet.
No	No	Acceda al Centro de información de DB2 en un sistema local.

Escenario: Acceso al Centro de información de DB2 en su sistema:

Tsu-Chen es propietario de una fábrica en una pequeña ciudad que no dispone de ISP local para proporcionarle acceso a Internet. Ha adquirido DB2 Universal Database™ para la gestión de su inventario, pedidos de productos, información de cuentas bancarias y gastos empresariales. Puesto que nunca había utilizado un producto de DB2 anteriormente, Tsu-Chen tendrá que aprender a partir de la documentación de productos de DB2.

Después de instalar DB2 Universal Database en el sistema utilizando la opción de instalación típica, Tsu-Chen intenta acceder a la documentación de DB2. Sin embargo, el navegador emite un mensaje de error que indica que la página que ha intentado abrir no se encuentra. Tsu-Chen comprueba el manual de instalación de su producto de DB2 y descubre que tiene que instalar el Centro de información de DB2 si desea acceder a la documentación de DB2 en su sistema. Encuentra el *CD del Centro de información de DB2* en el paquete de soportes y lo instala.

Desde el programa ejecutor de aplicaciones del sistema operativo, Tsu-Chen dispone ahora de acceso al Centro de información de DB2 y puede aprender a utilizar el producto de DB2 para incrementar el éxito de su empresa.

Escenario: Acceso al Centro de información de DB2 en el sitio Web de IBM:

Colin es un consultor de tecnologías de la información con una empresa de formación. Está especializado en tecnología de bases de datos y SQL y ofrece clases sobre estos temas a empresas por toda Norteamérica utilizando DB2 Universal Database. Parte de las clases de Colin incluye el uso de la documentación de DB2 como una herramienta didáctica. Por ejemplo, mientras imparte los cursos sobre SQL, Colin utiliza la documentación de DB2 relativa a SQL como un modo de enseñar sintaxis básica y avanzada para las consultas de base de datos.

La mayoría de las empresas en las que Colin imparte cursos tienen acceso a Internet. Esta situación ha influido en la decisión de Colin de configurar su sistema portátil para que acceda al Centro de información de DB2 en el sitio Web de IBM cuando ha instalado la versión más reciente de DB2 Universal Database. Dicha configuración permite a Colin disponer de acceso en línea a la documentación más reciente de DB2 durante sus clases.

Sin embargo, a veces, mientras viaja, Colin no tiene acceso a Internet. Esto le planteaba un problema, especialmente cuando necesitaba acceder a la documentación de DB2 para preparar las clases. A fin de evitar tales situaciones, Colin ha instalado una copia del Centro de información de DB2 en el sistema portátil.

Colin disfruta de la flexibilidad que supone tener siempre una copia de la documentación de DB2 a su disposición. Mediante el mandato **db2set**, puede configurar fácilmente las variables de registro en el sistema portátil para acceder al Centro de información de DB2 en el sitio Web de IBM o en el sistema portátil, según su situación.

Escenario: Acceso al Centro de información de DB2 en un servidor de intranet:

El trabajo de Eva es el de administrador sénior de bases de datos en una compañía de seguros de vida. Sus responsabilidades administrativas incluyen la instalación y configuración de la versión más reciente de DB2 Universal Database en los servidores de bases de datos UNIX® de la compañía. Recientemente, la compañía ha informado a sus empleados de que, por razones de seguridad, no se les proporcionará acceso a Internet en el trabajo. Dado que la compañía tiene un entorno de red, Eva decide instalar una copia del Centro de información de DB2 en un servidor de intranet a fin de que todos los empleados de la compañía que utilicen el depósito de datos de la misma de forma regular (representantes de ventas, gestores de ventas y analistas de empresa) tengan acceso a la documentación de DB2.

Eva indica a su equipo encargado de las bases de datos que instalen la versión más reciente de DB2 Universal Database en los sistemas de todos los empleados a través de un archivo de respuestas, para asegurarse de que cada sistema esté configurado de manera que acceda al Centro de información de DB2 utilizando el nombre de sistema principal y el número de puerto del servidor de intranet.

No obstante, debido a un malentendido, Miguel, un administrador de bases de datos auxiliar del equipo de Eva, instala una copia del Centro de información de DB2 en varios sistemas de los empleados en lugar de configurar DB2 Universal

Database para que acceda al Centro de información de DB2 en el servidor de intranet. Con el fin de corregir esta situación, Eva indica a Miguel que utilice el mandato **db2set** para cambiar las variables de registro del Centro de información de DB2 (DB2_DOCHOST para el nombre de sistema principal y DB2_DOCPORT para el número de puerto) en cada uno de esos sistemas. Ahora todos los sistemas correspondientes de la red tienen acceso al Centro de información de DB2, y los empleados pueden hallar las respuestas a sus preguntas sobre DB2 en la documentación de DB2.

Conceptos relacionados:

- "Centro de información de DB2" en la página 780

Tareas relacionadas:

- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 784
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 787
- "Establecimiento de la ubicación para acceder al Centro de información de DB2"

Información relacionada:

- "db2set - Mandato Registro de perfiles de DB2" en la publicación *Consulta de mandatos*

Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)

Se puede acceder a la documentación de los productos de DB2 de tres maneras: en el sitio Web de IBM, en un servidor de intranet o en una versión instalada en el sistema. Por omisión, el acceso de los productos de DB2 dentro de la documentación de DB2 se efectúa en el sitio Web de IBM. Si desea acceder a la documentación de DB2 en un servidor de intranet o en su propio sistema, deberá instalar la documentación desde el *CD del Centro de información de DB2*. Mediante el asistente de instalación de DB2, puede definir sus preferencias de instalación e instalar el Centro de información de DB2 en un sistema que utilice un sistema operativo UNIX.

Prerrequisitos:

Este apartado lista los requisitos de hardware, sistema operativo, software y comunicaciones para instalar el Centro de información de DB2 en los sistemas UNIX.

- **Requisitos de hardware**

  Necesita uno de los procesadores siguientes:
  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel de 32 bits (Linux)
  - Sistemas Solaris UltraSPARC (Entorno operativo Solaris)
- **Requisitos de sistema operativo**

  Necesita uno de los sistemas operativos siguientes:

- IBM AIX 5.1 (en PowerPC)
- HP-UX 11i (en HP 9000)
- Red Hat Linux 8.0 (en Intel de 32 bits)
- SuSE Linux 8.1 (en Intel de 32 bits)
- Sun Solaris Versión 8 (en sistemas UltraSPARC del Entorno operativo Solaris)

Nota: El Centro de información de DB2 se ejecuta en un subconjunto de los sistemas operativos UNIX en los que están soportados los clientes DB2. Por consiguiente, es recomendable que acceda al Centro de información de DB2 desde el sitio Web de IBM o que instale el Centro de información de DB2 y acceda al mismo en un servidor de intranet.

- **Requisitos de software**
  - Está soportado el navegador siguiente:
    - Mozilla Versión 1.0 o superior
- El asistente de instalación de DB2 es un instalador gráfico. Debe disponer de una implementación del software X Window System capaz de representar una interfaz gráfica de usuario para que el asistente de instalación de DB2 se ejecute en el sistema. A fin de ejecutar el asistente de instalación de DB2, debe asegurarse de que ha exportado debidamente la visualización. Por ejemplo, entre el mandato siguiente en el indicador de mandatos:

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **Requisitos de comunicaciones**
  - TCP/IP

Procedimiento:

Para instalar el Centro de información de DB2 utilizando el asistente de instalación de DB2:

1. Inicie una sesión en el sistema.
2. Inserte y monte el CD del producto Centro de información de DB2 en el sistema.
3. Vaya al directorio en el que está montado el CD entrando el mandato siguiente:

   ```
   cd /cd
   ```

   donde */cd* representa el punto de montaje del CD.
4. Entre el mandato **./db2setup** para iniciar el asistente de instalación de DB2.
5. Se abrirá el Área de ejecución para la instalación de IBM DB2. Para continuar directamente con la instalación del Centro de información de DB2, pulse en **Instalar producto**. Existe ayuda en línea disponible para guiarle durante los pasos restantes. Para invocar la ayuda en línea, pulse en **Ayuda**. Puede pulsar en **Cancelar** en cualquier momento para interrumpir la instalación.
6. En la página **Seleccione el producto que desee instalar**, pulse en **Siguiente**.
7. Pulse en **Siguiente** en la página **Bienvenido al asistente de instalación de DB2**. El asistente de instalación de DB2 le guiará durante el proceso de instalación del programa.
8. Para continuar con la instalación, debe aceptar el contrato de licencia. En la página **Contrato de licencia**, seleccione **Acepto los términos del contrato de licencia** y pulse en **Siguiente**.
9. Seleccione **Instalar el Centro de información de DB2 en este sistema** en la página **Seleccionar la acción de instalación**. Si desea utilizar un archivo de

respuestas para instalar el Centro de información de DB2 en éste o en otros sistemas más adelante, seleccione **Guardar los valores en un archivo de respuestas**. Pulse en **Siguiente**.

10. Seleccione los idiomas en los que se instalará el Centro de información de DB2 en la página **Seleccionar los idiomas a instalar**. Pulse en **Siguiente**.
11. Configure el Centro de información de DB2 para las comunicaciones entrantes en la página **Especificar el puerto del Centro de información de DB2**. Pulse en **Siguiente** para continuar la instalación.
12. Revise las opciones de instalación que ha elegido en la página **Comenzar a copiar archivos**. Para cambiar cualquier valor, pulse en **Anterior**. Pulse en **Instalar** para copiar los archivos del Centro de información de DB2 en el sistema.

También puede instalar el Centro de información de DB2 utilizando un archivo de respuestas.

Los archivos de anotaciones cronológicas de instalación db2setup.his, db2setup.log y db2setup.err están ubicados, por omisión, en el directorio /tmp.

El archivo db2setup.log capta toda la información de instalación del producto de DB2, incluidos los errores. El archivo db2setup.his registra todas las instalaciones de productos de DB2 en el sistema. DB2 añade el archivo db2setup.log al archivo db2setup.his. El archivo db2setup.err capta cualquier salida de errores devuelta por Java, como, por ejemplo, información de interrupciones y excepciones.

Cuando se haya completado la instalación, el Centro de información de DB2 estará instalado en uno de los directorios siguientes, según el sistema operativo UNIX:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Entorno operativo Solaris: /opt/IBM/db2/V8.1

Conceptos relacionados:

- "Centro de información de DB2" en la página 780
- "Escenarios de instalación del Centro de información de DB2" en la página 782

Tareas relacionadas:

- "Instalación de DB2 utilizando un archivo de respuestas (UNIX)" en la publicación *Suplemento de instalación y configuración*
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 791
- "Invocación del Centro de información de DB2" en la página 789
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 787

Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)

Se puede acceder a la documentación de los productos de DB2 de tres maneras: en el sitio Web de IBM, en un servidor de intranet o en una versión instalada en el sistema. Por omisión, el acceso de los productos de DB2 dentro de la documentación de DB2 se efectúa en el sitio Web de IBM. Si desea acceder a la documentación de DB2 en un servidor de intranet o en su propio sistema, deberá instalar la documentación de DB2 desde el *CD del Centro de información de DB2*. Mediante el asistente de instalación de DB2, puede definir sus preferencias de instalación e instalar el Centro de información de DB2 en un sistema que utilice un sistema operativo Windows.

Prerrequisitos:

Este apartado lista los requisitos de hardware, sistema operativo, software y comunicaciones para instalar el Centro de información de DB2 en Windows.

- **Requisitos de hardware**

  Necesita uno de los procesadores siguientes:

  – Sistemas de 32 bits: una CPU Pentium o compatible con Pentium
- **Requisitos de sistema operativo**

  Necesita uno de los sistemas operativos siguientes:

  – Windows 2000

  – Windows XP

  Nota: El Centro de información de DB2 se ejecuta en un subconjunto de los sistemas operativos Windows en los que están soportados los clientes DB2. Por consiguiente, es recomendable que acceda al Centro de información de DB2 en el sitio Web de IBM o que instale el Centro de información de DB2 y acceda al mismo en un servidor de intranet.
- **Requisitos de software**

  – Están soportados los navegadores siguientes:

    - Mozilla 1.0 o superior

    - Internet Explorer Versión 5.5 ó 6.0 (Versión 6.0 para Windows XP)
- **Requisitos de comunicaciones**

  – TCP/IP

Restricciones:

- Necesita una cuenta con privilegios administrativos para instalar el Centro de información de DB2.

Procedimiento:

Para instalar el Centro de información de DB2 utilizando el asistente de instalación de DB2:

1. Inicie una sesión en el sistema con la cuenta que ha definido para la instalación del Centro de información de DB2.
2. Inserte el CD en la unidad. Si está habilitada, la característica de ejecución automática inicia el Área de ejecución para la instalación de IBM DB2.
3. El asistente de instalación de DB2 determina el idioma del sistema y ejecuta el programa de instalación para ese idioma. Si desea ejecutar el programa de

instalación en un idioma distinto del inglés o bien el programa de instalación no se inicia de forma automática, puede iniciar el asistente de instalación de DB2 manualmente.

Para iniciar el asistente de instalación de DB2 manualmente:

a. Pulse en **Inicio** y seleccione **Ejecutar**.
b. En el campo **Abrir**, escriba el mandato siguiente:

```
x:\setup.exe /i identificador de idioma de 2 letras
```

donde *x:* representa la unidad de CD, e *identificador de idioma de 2 letras* representa el idioma en el que se ejecutará el programa de instalación.

c. Pulse en **Aceptar**.

4. Se abrirá el Área de ejecución para la instalación de IBM DB2. Para continuar directamente con la instalación del Centro de información de DB2, pulse en **Instalar producto**. Existe ayuda en línea disponible para guiarle durante los pasos restantes. Para invocar la ayuda en línea, pulse en **Ayuda**. Puede pulsar en **Cancelar** en cualquier momento para interrumpir la instalación.
5. En la página **Seleccione el producto que desee instalar**, pulse en **Siguiente**.
6. Pulse en **Siguiente** en la página **Bienvenido al asistente de instalación de DB2**. El asistente de instalación de DB2 le guiará durante el proceso de instalación del programa.
7. Para continuar con la instalación, debe aceptar el contrato de licencia. En la página **Contrato de licencia**, seleccione **Acepto los términos del contrato de licencia** y pulse en **Siguiente**.
8. Seleccione **Instalar el Centro de información de DB2 en este sistema** en la página **Seleccionar la acción de instalación**. Si desea utilizar un archivo de respuestas para instalar el Centro de información de DB2 en éste o en otros sistemas más adelante, seleccione **Guardar los valores en un archivo de respuestas**. Pulse en **Siguiente**.
9. Seleccione los idiomas en los que se instalará el Centro de información de DB2 en la página **Seleccionar los idiomas a instalar**. Pulse en **Siguiente**.
10. Configure el Centro de información de DB2 para las comunicaciones entrantes en la página **Especificar el puerto del Centro de información de DB2**. Pulse en **Siguiente** para continuar la instalación.
11. Revise las opciones de instalación que ha elegido en la página **Comenzar a copiar archivos**. Para cambiar cualquier valor, pulse en **Anterior**. Pulse en **Instalar** para copiar los archivos del Centro de información de DB2 en el sistema.

Puede instalar el Centro de información de DB2 utilizando un archivo de respuestas. También es posible utilizar el mandato **db2rspgn** a fin de generar un archivo de respuestas basado en una instalación existente.

Para obtener información sobre los errores encontrados durante la instalación, consulte los archivos db2.log y db2wi.log ubicados en el directorio 'Mis documentos'\DB2LOG\. La ubicación del directorio 'Mis documentos' dependerá de la configuración de su sistema.

El archivo db2wi.log capta la información de la instalación de DB2 más reciente. El archivo db2.log capta el historial de instalaciones de productos de DB2.

Conceptos relacionados:

- "Centro de información de DB2" en la página 780

- "Escenarios de instalación del Centro de información de DB2" en la página 782

Tareas relacionadas:

- "Instalación de un producto DB2 utilizando un archivo de respuestas (Windows)" en la publicación *Suplemento de instalación y configuración*
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Visualización de temas en el idioma preferido en el Centro de información de DB2" en la página 791
- "Invocación del Centro de información de DB2" en la página 789
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 784

Información relacionada:

- "db2rspgn - Mandato del Generador de archivos de respuestas (Windows)" en la publicación *Consulta de mandatos*

Invocación del Centro de información de DB2

El Centro de información de DB2 proporciona acceso a toda la información que necesita para utilizar productos de DB2 para los sistemas operativos Linux, UNIX y Windows, tales como DB2 Universal Database, DB2 Connect, DB2 Information Integrator y DB2 Query Patroller.

Puede invocar el Centro de información de DB2 desde una de las ubicaciones siguientes:

- Sistemas en los que está instalado un cliente o servidor DB2 UDB
- Un servidor de intranet o sistema local en el que está instalado el Centro de información de DB2
- El sitio Web de IBM

Prerrequisitos:

Antes de invocar el Centro de información de DB2:

- *Opcional*: Configure el navegador para que visualice los temas en su idioma preferido
- *Opcional*: Configure el cliente DB2 para que utilice el Centro de información de DB2 instalado en el sistema o servidor de intranet

Procedimiento:

Para invocar el Centro de información de DB2 en un sistema en el que está instalado un cliente o servidor DB2 UDB:

- Desde el menú Inicio (sistema operativo Windows): Pulse en **Inicio** → **Programas** → **IBM DB2** → **Información** → **Centro de información**.
- Desde el indicador de línea de mandatos:
  - En los sistemas operativos Linux y UNIX, emita el mandato **db2icdocs**.
  - En el sistema operativo Windows, emita el mandato **db2icdocs.exe**.

Para abrir el Centro de información de DB2 instalado en un servidor de intranet o sistema local en un navegador Web:

- Abra la página Web en http://<nombre-sistemaprincipal>:<número-puerto>/, donde <nombre-sistemaprincipal> representa el nombre de sistema principal y <número-puerto> representa el número de puerto en el que está disponible el Centro de información de DB2.

Para abrir el Centro de información de DB2 en el sitio Web de IBM en un navegador Web:

- Abra la página Web en publib.boulder.ibm.com/infocenter/db2help/.

Conceptos relacionados:

- "Centro de información de DB2" en la página 780
- "Escenarios de instalación del Centro de información de DB2" en la página 782

Tareas relacionadas:

- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 798
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 790
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 800
- "Establecimiento de la ubicación para acceder al Centro de información de DB2"

Información relacionada:

- "Mandato HELP" en la publicación *Consulta de mandatos*

Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet

El Centro de información de DB2 que hay disponible en http://publib.boulder.ibm.com/infocenter/db2help/ se actualizará periódicamente con documentación nueva o modificada. Asimismo, IBM puede efectuar actualizaciones del Centro de información de DB2 disponibles para descargar e instalar en el sistema o servidor de intranet. La actualización del Centro de información de DB2 no actualiza los productos de cliente o servidor DB2.

Prerrequisitos:

Es necesario tener acceso a un sistema que esté conectado a Internet.

Procedimiento:

Para actualizar el Centro de información de DB2 instalado en el sistema o servidor de intranet:

1. Abra el Centro de información de DB2 que se encuentra en el sitio Web de IBM de: http://publib.boulder.ibm.com/infocenter/db2help/
2. En la sección de descargas de la página de bienvenida, bajo la cabecera de servicio y soporte, pulse en el enlace de **documentación de DB2 Universal Database**.
3. Determine si la versión de su Centro de información de DB2 está anticuada comparando el nivel de la última imagen de documentación renovada con el

nivel de documentación que tenga instalado. El nivel de documentación que ha instalado aparece listado en la página de bienvenida del Centro de información de DB2.

4. Si se encuentra disponible una versión más reciente del Centro de información de DB2, descargue la última imagen renovada del *Centro de información de DB2* aplicable a su sistema operativo.
5. Para instalar la imagen renovada del *Centro de información de DB2*, siga las instrucciones proporcionadas en la página Web.

Conceptos relacionados:

- "Escenarios de instalación del Centro de información de DB2" en la página 782

Tareas relacionadas:

- "Invocación del Centro de información de DB2" en la página 789
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (UNIX)" en la página 784
- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Windows)" en la página 787

Visualización de temas en el 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 de navegador. Si un tema no se ha traducido al idioma preferido del usuario, el Centro de información de DB2 visualiza dicho tema en inglés.

Procedimiento:

Para visualizar temas en su idioma preferido en el navegador Internet Explorer:

1. En Internet Explorer, pulse 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 a fin de visualizar el Centro de información de DB2 en su idioma preferido.

Para visualizar temas en su idioma preferido en el navegador Mozilla:

1. En Mozilla, seleccione el botón **Edit** —> **Preferences** —> **Languages**. Se visualizará el panel Languages en la ventana Preferences.
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 **Add...** a fin de seleccionar un idioma en la ventana Add Languages.

- Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Move Up** hasta que el idioma esté en primer lugar en la lista de idiomas.

3. Renueve la página a fin de visualizar el Centro de información de DB2 en su idioma preferido.

Conceptos relacionados:

- "Centro de información de DB2" en la página 780

Documentación PDF e impresa de DB2

Las tablas siguientes proporcionan los nombres oficiales de los manuales, los números de documento y los nombres de los archivos PDF. Para solicitar manuales en copia impresa, debe conocer el nombre oficial del manual. Para imprimir un archivo PDF, debe conocer el nombre del archivo PDF.

La documentación de DB2 está categorizada según las cabeceras siguientes:

- Información básica de DB2
- Información de administración
- Información para el desarrollo de aplicaciones
- Información de Business Intelligence
- Información de DB2 Connect
- Información de iniciación
- Información de aprendizaje
- Información sobre componentes opcionales
- Notas del release

Las tablas siguientes describen, para cada manual de la biblioteca de DB2, la información necesaria para solicitar la copia impresa o para imprimir o ver el PDF correspondiente al manual en cuestión. Se encuentra una descripción completa de cada uno de los manuales de la biblioteca de DB2 en el Centro de publicaciones de IBM de www.ibm.com/shop/publications/order

Información básica de DB2

La información de estos manuales es fundamental para todos los usuarios de DB2; encontrará útil esta información tanto si es programador o administrador de bases de datos como si trabaja con DB2 Connect, DB2 Warehouse Manager u otros productos de DB2.

Tabla 13. Información básica de DB2

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Consulta de mandatos	SC10-3725	db2n0x81
IBM DB2 Universal Database Glosario	Sin número de documento	db2t0x81
IBM DB2 Universal Database Consulta de mensajes, Volumen 1	GC10-3728, no disponible en copia impresa	db2m1x81
IBM DB2 Universal Database Consulta de mensajes, Volumen 2	GC10-3729, no disponible en copia impresa	db2m2x81
IBM DB2 Universal Database Novedades	SC10-3734	db2q0x81

Información de administración

La información de estos manuales incluye los temas necesarios para diseñar, implementar y mantener de forma efectiva bases de datos de DB2, depósitos de datos y sistemas federados.

Tabla 14. Información de administración

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822	db2d1x81
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820	db2d2x81
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821	db2d3x81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference	SC09-4831	db2hax81
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx81
IBM DB2 Universal Database Consulta de SQL, Volumen 1	SC10-3730	db2s1x81
IBM DB2 Universal Database Consulta de SQL, Volumen 2	SC10-3731	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

Información para el desarrollo de aplicaciones

La información de estos manuales es de especial interés para los programadores de aplicaciones o programadores que trabajan con DB2 Universal Database (DB2 UDB). Hallará información acerca de los lenguajes y compiladores soportados, así como la documentación necesaria para acceder a DB2 UDB utilizando las diversas interfaces de programación soportadas, como, por ejemplo, SQL incorporado, ODBC, JDBC, SQLJ y CLI. Si utiliza el Centro de información de DB2, también podrá acceder a versiones HTML del código fuente para los programas de ejemplo.

Tabla 15. Información para el desarrollo de aplicaciones

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Creación y ejecución de aplicaciones	SC10-3733	db2axx81

Tabla 15. Información para el desarrollo de aplicaciones (continuación)

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Programación de aplicaciones de cliente	SC10-3723	db2a1x81
IBM DB2 Universal Database Guía de desarrollo de aplicaciones: Programación de aplicaciones de servidor	SC10-3724	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administración y programación	SC10-3750	db2sxx81

Información de Business Intelligence

La información de estos manuales describe cómo utilizar los componentes que mejoran las posibilidades de análisis y de depósito de datos de DB2 Universal Database.

Tabla 16. Información de Business Intelligence

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	GC27-1122	db2idx81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

Información de DB2 Connect

La información incluida en esta categoría describe cómo acceder a datos de servidores de sistema principal y de sistema medio utilizando DB2 Connect Enterprise Edition o DB2 Connect Personal Edition.

Tabla 17. Información de DB2 Connect

Nombre	Número de documento	Nombre de archivo PDF
IBM Connectivity Supplement	Sin número de documento	db2h1x81

Tabla 17. Información de DB2 Connect (continuación)

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Connect Guía rápida de iniciación para DB2 Enterprise Edition	GC10-3774	db2c6x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition	GC09-4834	db2c1x81
IBM DB2 Connect User's Guide	SC09-4835	db2c0x81

Información de iniciación

La información de esta categoría es útil cuando se van a instalar y configurar servidores, clientes y otros productos de DB2.

Tabla 18. Información de iniciación

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Universal Database Guía rápida de iniciación para clientes DB2	GC10-3775, no disponible en copia impresa	db2itx81
IBM DB2 Universal Database Guía rápida de iniciación para servidores DB2	GC10-3773	db2isx81
IBM DB2 Universal Database Guía rápida de iniciación para DB2 Personal Edition	GC10-3771	db2i1x81
IBM DB2 Universal Database Suplemento de instalación y configuración	GC10-3772, no disponible en copia impresa	db2iyx81
IBM DB2 Universal Database Guía rápida de iniciación para DB2 Data Links Manager	GC10-3726	db2z6x81

Información de aprendizaje

La información de aprendizaje presenta las características de DB2 y explica cómo realizar diversas tareas.

Tabla 19. Información de aprendizaje

Nombre	Número de documento	Nombre de archivo PDF
Guía de aprendizaje de Business Intelligence: Introducción al Centro de depósito de datos	Sin número de documento	db2tux81
Guía de aprendizaje de Business Intelligence: Lecciones ampliadas sobre depósito de datos	Sin número de documento	db2tax81
Information Catalog Center Tutorial	Sin número de documento	db2aix81
Guía de aprendizaje de Video Central para e-business	Sin número de documento	db2twx81
Guía de aprendizaje de Visual Explain	Sin número de documento	db2tvx81

Información sobre componentes opcionales

La información de esta categoría describe cómo trabajar con los componentes opcionales de DB2.

Tabla 20. Información sobre componentes opcionales

Nombre	Número de documento	Nombre de archivo PDF
IBM DB2 Cube Views Guía y consulta	SC10–3868	db2aax81
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender Guía del usuario y de consulta	SC10-3755	db2sbx81
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
DB2 Net Search Extender Administración y guía del usuario **Nota:** El HTML para este documento *no* se instala desde el CD de documentación HTML.	SH10-9305	N/D

Notas del release

Las notas del release proporcionan información adicional específica del release y nivel de FixPak del producto. Las notas del release también proporcionan resúmenes de las actualizaciones de la documentación que se han incorporado en cada release, actualización y FixPak.

Tabla 21. Notas del release

Nombre	Número de documento	Nombre de archivo PDF
Notas del release de DB2	Ver nota.	Ver nota.
Notas de instalación de DB2	Sólo disponible en el CD-ROM del producto.	No disponible.

Nota: Las Notas del release están disponibles en:

- XHTML y formato de texto, en los CD de los productos
- Formato PDF, en el CD de documentación PDF

Además, las partes de las Notas del release que tratan *Problemas conocidos y soluciones alternativas* e *Incompatibilidades entre releases* también aparecen en el Centro de información de DB2.

Para ver las Notas del release en formato de texto en las plataformas basadas en UNIX, consulte el archivo Release.Notes. Este archivo se encuentra en el directorio DB2DIR/Readme/%L, donde *%L* representa el nombre de entorno nacional y `DB2DIR` representa:

- En los sistemas operativos AIX: /usr/opt/db2_08_01
- En los otros sistemas operativos basados en UNIX: /opt/IBM/db2/V8.1

Conceptos relacionados:

- "Documentación y ayuda de DB2" en la página 779

Tareas relacionadas:

- "Impresión de manuales de DB2 desde archivos PDF" en la página 797
- "Solicitud de manuales de DB2 impresos" en la página 798
- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 798

Impresión de manuales de DB2 desde archivos PDF

Puede imprimir los manuales de DB2 desde los archivos PDF del *CD de documentación PDF de DB2*. Mediante la utilización de Adobe Acrobat Reader, puede imprimir el manual entero o un rango específico de páginas.

Prerrequisitos:

Asegúrese de que tiene instalado Adobe Acrobat Reader. Si ha de instalar Adobe Acrobat Reader, está disponible desde el sitio Web de Adobe en www.adobe.com

Procedimiento:

Para imprimir un manual de DB2 desde un archivo PDF:

1. Inserte el *CD de documentación PDF de DB2*. En sistemas operativos UNIX, monte el CD de documentación PDF de DB2. Consulte el manual *Iniciación rápida* para obtener detalles sobre cómo montar un CD en sistemas operativos UNIX.
2. Abra index.htm. El archivo de abre en una ventana de navegador.
3. Pulse el título del PDF que desee ver. El PDF se abrirá en Acrobat Reader.
4. Seleccione **Archivo** → **Imprimir** para imprimir cualquier parte que desee del manual.

Conceptos relacionados:

- "Centro de información de DB2" en la página 780

Tareas relacionadas:

- "Montaje del CD-ROM (AIX)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Cómo montar el CD-ROM (HP-UX)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Montaje del CD-ROM (Linux)" en la publicación *Guía rápida de iniciación para servidores DB2*
- "Solicitud de manuales de DB2 impresos" en la página 798
- "Montaje del CD-ROM (Entorno operativo Solaris)" en la publicación *Guía rápida de iniciación para servidores DB2*

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 792

Solicitud de manuales de DB2 impresos

Si prefiere utilizar manuales en copia impresa, puede solicitarlos de tres modos distintos.

Procedimiento:

Los manuales impresos se pueden solicitar en algunos países o regiones. Compruebe, en el sitio Web de publicaciones de IBM correspondiente a su país o región, si este servicio está disponible en su país o región. Cuando las publicaciones estén disponibles para su solicitud, puede realizar lo siguiente:

- Póngase en contacto con el distribuidor autorizado o representante de marketing de IBM. Para encontrar un representante local de IBM, consulte el directorio mundial de contactos de IBM en la página Web www.ibm.com/planetwide
- Llame al teléfono `1-800-879-2755`, si está en los EE.UU. o al `1-800-IBM-4YOU`, si está en Canadá.
- Visite el Centro de publicaciones de IBM en http://www.ibm.com/shop/publications/order. La capacidad de solicitar manuales desde el Centro de publicaciones de IBM puede no estar disponible en todos los países.

En el momento en que un producto de DB2 se encuentra disponible, los manuales impresos son los mismos que aparecen en formato PDF en el *CD de documentación PDF de DB2*. El contenido de los manuales impresos que se halla en el *CD del Centro de información de DB2* también es el mismo. No obstante, existe contenido adicional en el CD del Centro de información de DB2 que no aparece en ninguno de los manuales PDF (por ejemplo, rutinas de administración de SQL y ejemplos de HTML). No todos los manuales incluidos en el CD de documentación PDF de DB2 se pueden solicitar en copia impresa.

Nota: El Centro de información de DB2 se actualiza con mayor frecuencia que los manuales PDF o en copia impresa; instale las actualizaciones de la documentación a medida que estén disponibles o consulte el Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2help/ para obtener la información más actualizada.

Tareas relacionadas:

- "Impresión de manuales de DB2 desde archivos PDF" en la página 797

Información relacionada:

- "Documentación PDF e impresa de DB2" en la página 792

Invocación de ayuda según contexto desde una herramienta de DB2

La ayuda según contexto proporciona información sobre las tareas o controles que están asociados con una ventana, cuaderno, asistente o asesor determinado. La ayuda según contexto está disponible desde las herramientas de administración y desarrollo de DB2 que tienen interfaces gráficas de usuario. Existen dos tipos de ayuda según contexto:

- Ayuda a la que se accede mediante el botón **Ayuda** ubicado en cada ventana o cuaderno.

- Ventanas emergentes de información, que son ventanas que se visualizan cuando el cursor del ratón se coloca sobre un campo o control o cuando se selecciona un campo o control en una ventana, cuaderno, asistente o asesor y se pulsa F1.

El botón **Ayuda** proporciona acceso a la información de visión general, de prerrequisitos y de tareas. Las ventanas emergentes de información describen los campos y controles individuales.

Procedimiento:

Para invocar la ayuda según contexto:

- Para la ayuda de ventana y de cuaderno, inicie una de las herramientas de DB2 y, luego, abra cualquier ventana o cuaderno. Pulse el botón **Ayuda** situado en la esquina inferior derecha de la ventana o del cuaderno a fin de invocar la ayuda según contexto.

  También puede acceder a la ayuda según contexto desde el elemento de menú **Ayuda** situado en la parte superior de cada uno de los centros de herramientas de DB2.

  Para los asistentes y asesores, pulse en el enlace Visión general de tareas, de la primera página, si desea ver ayuda según contexto.
- Para obtener ayuda sobre controles individuales de una ventana o un cuaderno en una ventana emergente de información, pulse el control y, a continuación, pulse **F1**. La información emergente que contiene detalles sobre el control se visualizará en una ventana amarilla.

  Nota: Para visualizar ventanas emergentes de información simplemente manteniendo el cursor del ratón sobre un campo o control, seleccione el recuadro de selección **Visualizar automáticamente ventanas emergentes de información** en la página **Documentación** del cuaderno Valores de herramientas.

  Similar a las ventanas emergentes de información, la información emergente de diagnóstico es otra forma de ayuda según contexto; en ella se incluyen reglas para la entrada de datos. La información emergente de diagnóstico se visualiza en una ventana de color morado que aparece cuando se entran datos que no son válidos o que son insuficientes. La información emergente de diagnóstico puede aparecer para:
  - Campos obligatorios.
  - Campos cuyos datos tengan un formato preciso como, por ejemplo, un campo de fecha.

Tareas relacionadas:

- “Invocación del Centro de información de DB2” en la página 789
- “Invocación de la ayuda de mensajes desde el procesador de línea de mandatos” en la página 800
- “Invocación de la ayuda de mandatos desde el procesador de línea de mandatos” en la página 800
- “Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos” en la página 801
- “Acceso al Centro de información de DB2”
- “Cómo utilizar la ayuda de DB2 UDB”
- “Establecimiento de la ubicación para acceder al Centro de información de DB2”
- “Configuración del acceso a documentación y ayuda contextual de DB2”

Invocación de la ayuda de mensajes desde el procesador de línea de mandatos

La ayuda de mensajes describe la causa de un mensaje y describe la acción que se debe realizar en respuesta al error.

Procedimiento:

Para invocar la ayuda de mensajes, abra el procesador de línea de mandatos y entre:

```
? XXXnnnnn
```

donde *XXXnnnnn* representa un identificador de mensaje válido.

Por ejemplo, `? SQL30081` muestra la ayuda acerca del mensaje SQL30081.

Conceptos relacionados:

- "Introducción a los mensajes" en la publicación *Consulta de mensajes Volumen 1*

Información relacionada:

- "db2: Mandato de invocación del procesador de línea de mandatos" en la publicación *Consulta de mandatos*

Invocación de la ayuda de mandatos desde el procesador de línea de mandatos

La ayuda de mandatos explica la sintaxis de los mandatos del procesador de línea de mandatos.

Procedimiento:

Para invocar la ayuda de mandatos, abra el procesador de línea de mandatos y entre:

```
? mandato
```

donde *mandato* representa una palabra clave o el mandato completo.

Por ejemplo, `? catalog` visualiza ayuda para todos los mandatos CATALOG, mientras que `? catalog database` visualiza ayuda solamente para el mandato CATALOG DATABASE.

Tareas relacionadas:

- "Invocación de ayuda según contexto desde una herramienta de DB2" en la página 798
- "Invocación del Centro de información de DB2" en la página 789
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 800
- "Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos" en la página 801

Información relacionada:

- "db2: Mandato de invocación del procesador de línea de mandatos" en la publicación *Consulta de mandatos*

Invocación de la ayuda para estados de SQL desde el procesador de línea de mandatos

DB2 Universal Database devuelve un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

Procedimiento:

Para invocar 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.

Tareas relacionadas:

- "Invocación del Centro de información de DB2" en la página 789
- "Invocación de la ayuda de mensajes desde el procesador de línea de mandatos" en la página 800
- "Invocación de la ayuda de mandatos desde el procesador de línea de mandatos" en la página 800

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 ayudan a conocer los diversos aspectos de DB2 Universal Database. Las guías de aprendizaje proporcionan ejercicios con instrucciones paso a paso en las áreas de desarrollo de aplicaciones, ajuste del rendimiento de las consultas de SQL, trabajo con depósitos de datos, gestión de metadatos y desarrollo de servicios Web utilizando DB2.

Antes de empezar:

Puede ver las versiones XHTML de las guías de aprendizaje desde el Centro de información en http://publib.boulder.ibm.com/infocenter/db2help/.

Algunos ejercicios de las guías de aprendizaje utilizan datos o código de ejemplo. Consulte cada guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

Guías de aprendizaje de DB2 Universal Database:

Pulse en el título de una guía de aprendizaje de la lista siguiente para ver esa guía de aprendizaje.

Guía de aprendizaje de Business Intelligence: Introducción al Centro de depósito de datos
Realizar tareas de introducción de depósito de datos utilizando el Centro de depósito de datos.

Guía de aprendizaje de Business Intelligence: Lecciones ampliadas sobre depósito de datos
Realizar tareas avanzadas de depósito de datos utilizando el Centro de depósito de datos.

Information Catalog Center Tutorial
Crear y gestionar un catálogo de información para localizar y usar metadatos utilizando el Centro de catálogos de información.

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 de problemas y la determinación de problemas para ayudarle a utilizar los productos DB2®.

Documentación de DB2
La información de resolución de problemas se puede encontrar en todo el Centro de información de DB2, así como en todos los manuales PDF que componen la biblioteca de DB2. Puede consultar la rama sobre soporte y resolución de problemas, del árbol de navegación del Centro de información de DB2 (en el panel izquierdo de la ventana del navegador), para obtener un listado completo de la documentación de resolución de problemas de 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 las soluciones posibles. El sitio de soporte técnico tiene enlaces con las últimas publicaciones de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR), FixPaks y el listado más reciente de códigos de error internos de DB2, además de otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Para acceder al sitio Web de soporte de DB2, vaya a http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 Problem Determination Tutorial Series (Serie de guías de aprendizaje para la determinación de problemas de DB2)
Consulte el sitio Web DB2 Problem Determination Tutorial Series para encontrar información sobre cómo identificar y resolver rápidamente los problemas que puedan surgir mientras trabaje con DB2. Una de las guías de aprendizaje ofrece una presentación de los recursos y las herramientas de determinación de problemas de DB2 disponibles y le ayuda a decidir cuándo utilizarlos. Otras de las guías de aprendizaje tratan temas relacionados como, por ejemplo, ″Determinación de problemas del motor de base de datos″, ″Determinación de problemas de rendimiento″ y ″Determinación de problemas de aplicaciones″.

Consulte el conjunto completo de guías de aprendizaje de determinación de problemas de DB2 en el sitio de soporte técnico de DB2 de http://www.ibm.com/software/data/support/pdm/db2tutorials.html

Conceptos relacionados:

- "Centro de información de DB2" en la página 780
- "Introduction to problem determination - DB2 Technical Support tutorial" en la publicación *Troubleshooting Guide*

Accesibilidad

Las características de accesibilidad ayudan a los usuarios con discapacidades físicas, por ejemplo movilidad o visión limitada, a utilizar los productos de software satisfactoriamente. La lista siguiente especifica las características de accesibilidad principales de los productos de DB2® Versión 8:

- Toda la funcionalidad de DB2 está disponible utilizando el teclado para la navegación en lugar del ratón. Si desea más información, consulte el apartado "Entrada de teclado y navegación".
- Puede personalizar el tamaño y color de los fonts en las interfaces de usuario de DB2. Si desea más información, consulte el apartado "Pantalla accesible".
- Los productos de DB2 dan soporte a aplicaciones de accesibilidad que utilizan la API de accesibilidad de Java™. Si desea más información, consulte el apartado "Compatibilidad con tecnologías de asistencia" en la página 804.
- La documentación de DB2 se proporciona en un formato accesible. Si desea más información, consulte el apartado "Documentación accesible" en la página 804.

Entrada de teclado y navegación

Entrada de teclado

Puede trabajar con las herramientas de DB2 utilizando solamente el teclado. Puede utilizar teclas o combinaciones de teclas para llevar a cabo operaciones que también se pueden realizar con el ratón. Las pulsaciones estándares del sistema operativo se utilizan para operaciones estándares del sistema operativo.

Para obtener más información sobre el uso de teclas o combinaciones de teclas al realizar operaciones, consulte Accesos directos y aceleradores del teclado.

Navegación de teclado

Puede navegar por la interfaz de usuario de las herramientas de DB2 mediante teclas o combinaciones de teclas.

Para obtener más información sobre el uso de teclas o combinaciones de teclas al navegar por las herramientas de DB2, consulte Accesos directos y aceleradores del teclado.

Foco del teclado

En los sistemas operativos UNIX®, se resalta el área de la ventana activa en la que las pulsaciones tendrán efecto.

Pantalla accesible

Las herramientas de DB2 presentan características que mejoran la accesibilidad de los usuarios con poca visión u otras discapacidades visuales. Estas mejoras de la accesibilidad incluyen soporte para propiedades de font personalizables.

Valores de font

Puede seleccionar el color, tamaño y font del texto en menús y ventanas de diálogo utilizando el cuaderno Valores de herramientas.

Para obtener más información sobre cómo especificar valores de font, consulte Modificación de fonts para menús y texto.

No dependencia del color

No es necesario distinguir los colores para utilizar cualquiera de las funciones de este producto.

Compatibilidad con tecnologías de asistencia

Las interfaces de las herramientas de DB2 dan soporte a la API de accesibilidad de Java, que le permite utilizar lectores de pantalla y otras tecnologías de asistencia con los productos de DB2.

Documentación accesible

La documentación de DB2 se proporciona en formato XHTML 1.0, que se puede visualizar en la mayoría de los navegadores Web. XHTML le permite visualizar la documentación de acuerdo con las preferencias de pantalla establecidas en el navegador. También permite utilizar lectores de pantalla y otras tecnologías de asistencia.

Los diagramas de sintaxis se proporcionan en formato decimal con puntos. Este formato sólo está disponible si se accede a la documentación en línea mediante un lector de pantalla.

Conceptos relacionados:

- "Diagramas de sintaxis en formato decimal con puntos" en la página 804

Tareas relacionadas:

- "Accesos directos y aceleradores del teclado"
- "Modificación de fonts para menús y texto"

Diagramas de sintaxis en formato decimal con puntos

Se proporcionan diagramas de sintaxis en formato decimal con puntos para los usuarios que acceden al Centro de información utilizando un lector de pantalla.

En formato decimal con puntos, cada elemento de sintaxis se escribe en una línea distinta. Si dos o más elementos de sintaxis siempre aparecen juntos (o siempre están ausentes los dos a la vez), pueden aparecer en la misma línea, puesto que se pueden considerar un elemento de sintaxis compuesto.

Cada línea empieza por un número decimal con puntos; por ejemplo, 3 ó 3.1 ó 3.1.1. Para oír estos números correctamente, asegúrese de que su lector de pantalla esté configurado para leer la puntuación. Todos los elementos de sintaxis que tienen el mismo número decimal con puntos (por ejemplo, todos los elementos de sintaxis que tienen el número 3.1) son alternativas mutuamente excluyentes. Si oye las líneas 3.1 USERID y 3.1 SYSTEMID, sabrá que la sintaxis puede incluir o USERID o SYSTEMID, pero no ambos.

El nivel de numeración decimal con puntos denota el nivel jerárquico. Por ejemplo, si un elemento de sintaxis con el número decimal con puntos 3 va seguido de una serie de elementos de sintaxis con el número decimal 3.1, todos los elementos de sintaxis con la numeración 3.1 son subordinados de los elementos de sintaxis identificados por el número 3.

Junto a los números decimales con puntos se utilizan determinados símbolos y palabras para añadir información sobre los elementos de sintaxis. A veces, estos símbolos y palabras pueden aparecer al principio del propio elemento. Para facilitar la identificación, si la palabra o el símbolo forman parte del elemento de sintaxis, van precedidos por una barra inclinada invertida (\). El símbolo * se puede utilizar junto a un número decimal con puntos para indicar que el elemento de sintaxis se repite. Por ejemplo, el elemento de sintaxis *FILE con el número decimal con puntos 3 adopta el formato 3 * FILE. El formato 3* FILE indica que el elemento de sintaxis FILE se repite. El formato 3* * FILE indica que el elemento de sintaxis * FILE se repite.

Los caracteres como las comas, que se utilizan para separar una serie de elementos de sintaxis, se muestran en la sintaxis justo antes de los elementos que separan. Estos caracteres pueden aparecer en la misma línea que cada elemento o en una línea distinta con el mismo número decimal con puntos que los elementos en cuestión. En la línea también puede aparecer otro símbolo que proporcione información sobre los elementos de sintaxis. Por ejemplo, las líneas 5.1*, 5.1 LASTRUN y 5.1 DELETE significan que si se utiliza más de uno de los elementos de sintaxis LASTRUN y DELETE, los elementos deben estar separados por comas. Si no hay ningún separador, suponga que utiliza un espacio en blanco para separar cada elemento de sintaxis.

Si un elemento de sintaxis va precedido del símbolo %, esto indica una referencia que está definida en cualquier otro lugar. La serie que aparece después del símbolo % es el nombre de un fragmento de sintaxis en lugar de un literal. Por ejemplo, la línea 2.1 %OP1 significa que se debe hacer referencia al fragmento de sintaxis separado OP1.

Junto a los números decimales con puntos se utilizan los símbolos y las palabras siguientes:

- ? indica un elemento de sintaxis opcional. Un número decimal con puntos seguido del símbolo ? indica que todos los elementos de sintaxis con un número decimal con puntos correspondiente y elementos de sintaxis subordinados son opcionales. Si sólo hay un elemento de sintaxis con un número decimal con puntos, el símbolo ? aparecerá en la misma línea que el elemento de sintaxis (por ejemplo, 5? NOTIFY). Si hay más de un elemento de sintaxis con un número decimal con puntos, el símbolo ? aparecerá en una línea propia, seguido de los elementos de sintaxis opcionales. Por ejemplo, si oye las líneas 5 ?, 5 NOTIFY y 5 UPDATE, sabrá que los elementos de sintaxis NOTIFY y UPDATE son opcionales; es decir, puede seleccionar uno o ninguno de dichos elementos. El símbolo ? es equivalente a una línea de desvío de un diagrama de vías.
- ! indica un elemento de sintaxis por omisión. Un número decimal con puntos seguido del símbolo ! y un elemento de sintaxis indica que el elemento de sintaxis es la opción por omisión para todos los elementos de sintaxis que comparten el mismo número decimal con puntos. Sólo uno de los elementos de sintaxis que comparten el mismo número decimal con puntos puede especificar un símbolo !. Por ejemplo, si oye las líneas 2? FILE, 2.1! (KEEP) y 2.1 (DELETE), sabrá que (KEEP) es la opción por omisión correspondiente a la palabra clave FILE. En este ejemplo, si incluye la palabra clave FILE pero no especifica ninguna opción, se aplicará la opción por omisión KEEP. También se aplicará una opción por omisión al siguiente número decimal con puntos más alto. En este ejemplo, si se omite la palabra clave FILE, se utiliza el valor por omisión FILE(KEEP). No obstante, si oye las líneas 2? FILE, 2.1, 2.1.1! (KEEP) y 2.1.1 (DELETE), la opción por omisión KEEP sólo se aplicará al siguiente número

decimal con puntos más alto, 2.1 (que no tiene una palabra clave asociada) y no se aplicará a 2? FILE. Si se omite la palabra clave FILE, no se utilizará nada.

- * indica un elemento de sintaxis que se puede repetir 0 o más veces. Un número decimal con puntos seguido del símbolo * indica que este elemento de sintaxis se puede utilizar cero o más veces; es decir, es opcional y se puede repetir. Por ejemplo, si oye la línea 5.1* data area, sabrá que puede incluir un área de datos, más de un área de datos o ningún área de datos. Si oye las líneas 3*, 3 HOST y 3 STATE, sabrá que puede incluir HOST, STATE, los dos juntos o ninguno de los dos.

  Notas:

  1. Si un número decimal con puntos tiene un asterisco (*) al lado y sólo hay un elemento con dicho número decimal con puntos, podrá repetir el mismo elemento más de una vez.
  2. Si un número decimal con puntos tiene un asterisco al lado y hay varios elementos que tienen dicho número decimal con puntos, podrá utilizar más de un elemento de la lista, pero no podrá utilizar los elementos más de una vez cada uno. En el ejemplo anterior, podría escribir HOST STATE pero no podría escribir HOST HOST.
  3. El símbolo * es equivalente a una línea de bucle de retorno de un diagrama de sintaxis de vías.

- + indica un elemento de sintaxis que se debe incluir una o más veces. Un número decimal con puntos seguido del símbolo + indica que este elemento de sintaxis se debe incluir una o más veces; es decir, se debe incluir como mínimo una vez y se puede repetir. Por ejemplo, si oye la línea 6.1+ data area, deberá incluir como mínimo un área de datos. Si oye las líneas 2+, 2 HOST y 2 STATE, sabrá que debe incluir HOST, STATE o ambos. De manera similar al símbolo *, el símbolo + sólo puede repetir un elemento determinado si éste es el único elemento que tiene el número decimal con puntos en cuestión. El símbolo +, al igual que el símbolo *, es equivalente a una línea de bucle de retorno de un diagrama de sintaxis de vías.

Conceptos relacionados:

- "Accesibilidad" en la página 803

Tareas relacionadas:

- "Accesos directos y aceleradores del teclado"

Información relacionada:

- "Cómo se leen los diagramas de sintaxis" en la página vi

Certificación Common Criteria de productos DB2 Universal Database

Se está evaluando DB2 Universal Database para obtener la certificación Common Criteria en el nivel de garantía de evaluación 4 (EAL4). Para más información acerca de Common Criteria, consulte el sitio Web de Common Criteria en: http://niap.nist.gov/cc-scheme/.

Apéndice B. Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. 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 doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, 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 contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios Web.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido 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 puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo 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.

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

Los términos siguientes son marcas registradas de International Business Machines Corporation en los EE.UU. y/o en otros países y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeriesLAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Los términos siguientes son marcas registradas de otras empresas y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2 UDB:

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los EE.UU. y/o en otros países.

Intel y Pentium son marcas registradas de Intel Corporation en los EE.UU. y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los EE.UU. y/o en otros países.

UNIX es marca registrada de The Open Group en los EE.UU. 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

CH

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W

Cómo ponerse en contacto con IBM

En los EE.UU., puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (426-4968) para marketing y ventas de DB2

En Canadá, puede ponerse en contacto con IBM llamando a uno de los siguientes números:

- 1-800-IBM-SERV (1-800-426-7378) para servicio al cliente
- 1-800-465-9600 para obtener información sobre las opciones de servicio técnico disponibles
- 1-800-IBM-4YOU (1-800-426-4968) para marketing y ventas de DB2

Para localizar una oficina de IBM en su país o región, consulte IBM Directory of Worldwide Contacts en el sitio Web http://www.ibm.com/planetwide

Información sobre productos

La información relacionada con productos DB2 Universal Database se encuentra disponible por teléfono o a través de la World Wide Web en el sitio http://www.ibm.com/software/data/db2/udb

Este sitio contiene la información más reciente sobre la biblioteca técnica, pedidos de manuales, descargas de productos, grupos de noticias, FixPaks, novedades y enlaces con recursos de la Web.

Si vive en los EE.UU., puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

Para obtener información sobre cómo ponerse en contacto con IBM desde fuera de los EE.UU., vaya a la página IBM Worldwide en el sitio www.ibm.com/planetwide

SC10-3731-01

Spine information:

IBM® DB2 Universal Database™ Consulta de SQL, Volumen 2 Versión 8.2