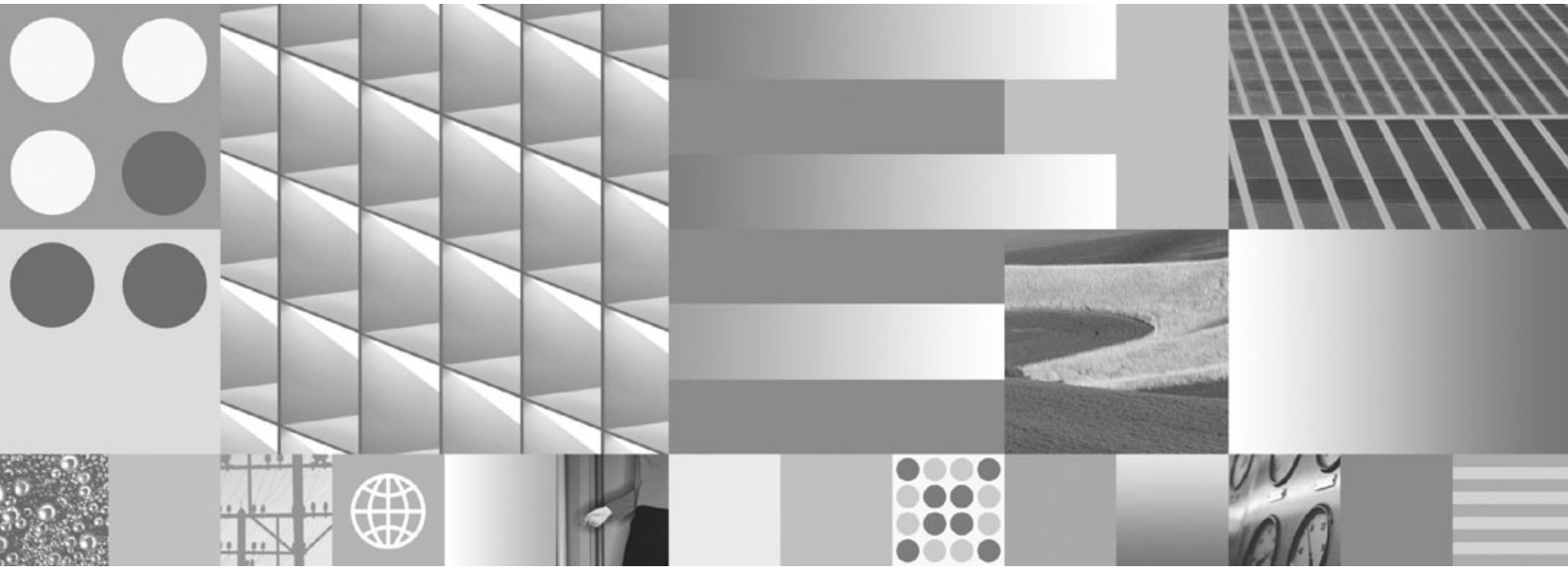
DB2 Versión 9.5
para Linux, UNIX y Windows

Versión 9 Release 5

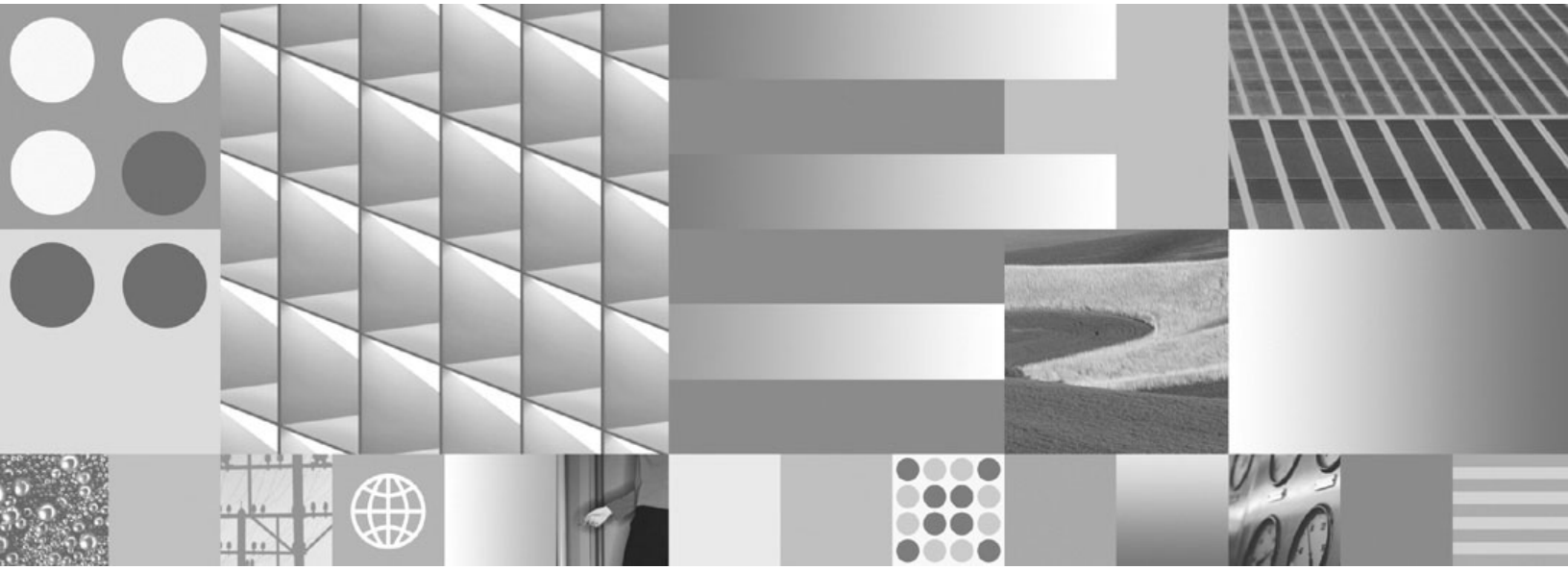


Consulta de SQL, Volumen 2
Actualizado en diciembre de 2010

SC11-3508-03

DB2 Versión 9.5
para Linux, UNIX y Windows

Versión 9 Release 5



Consulta de SQL, Volumen 2
Actualizado en diciembre de 2010

SC11-3508-03

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice B, "Avisos", en la página 1077.

Nota de edición

Este manual es la traducción del original en inglés *DB2 Version 9.5 for Linux, UNIX, and Windows Version 9 Release 5 SQL Reference, Volume 2 Updated December, 2010* (SC23-5862-03).

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

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

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

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

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

© Copyright IBM Corporation 1993, 2010.

Contenido

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

Sentencias 1
Sentencias de SQL 2
Cómo se invocan las sentencias de SQL 11
Acerca de las sentencias de control de SQL 15
Designadores de función, método y procedimiento 18
ALLOCATE CURSOR 23
ALTER AUDIT POLICY 25
ALTER BUFFERPOOL 29
ALTER DATABASE PARTITION GROUP 32
ALTER DATABASE 36
ALTER FUNCTION 40
ALTER HISTOGRAM TEMPLATE 43
ALTER METHOD 45
ALTER NICKNAME 47
ALTER PROCEDURE (externo) 56
ALTER PROCEDURE (con fuente) 59
ALTER PROCEDURE (SQL) 61
ALTER SECURITY LABEL COMPONENT 63
ALTER SECURITY POLICY 66
ALTER SEQUENCE 70
ALTER SERVER 74
ALTER SERVICE CLASS 78
ALTER TABLE 85
ALTER TABLESPACE 133
ALTER THRESHOLD 144
ALTER TRUSTED CONTEXT 151
ALTER TYPE (estructurado) 160
ALTER USER MAPPING 167
ALTER VIEW 169
ALTER WORK ACTION SET 171
ALTER WORK CLASS SET 183
ALTER WORKLOAD 188
ALTER WRAPPER 195
ALTER XSROBJECT 197
ASSOCIATE LOCATORS 198
AUDIT 200
BEGIN DECLARE SECTION 204
CALL 206
CASE 212
CLOSE 215
COMMENT 217
COMMIT 229
SQL compuesto (dinámico) 231
de SQL compuesto (incorporado) 236
SQL compuesto (procedimiento) 240
CONNECT (Tipo 1) 249
CONNECT (Tipo 2) 256
CREATE ALIAS 264
CREATE AUDIT POLICY 267
CREATE BUFFERPOOL 271
CREATE DATABASE PARTITION GROUP . . . 275
CREATE EVENT MONITOR 278
CREATE FUNCTION 299
CREATE FUNCTION (escalar externa) 300
CREATE FUNCTION (tabla externa) 326
CREATE FUNCTION (tabla externa OLE DB) . . 346
CREATE FUNCTION (con fuente o plantilla) . . . 355
CREATE FUNCTION (escalar de SQL, tabla o fila) 367
CREATE FUNCTION MAPPING 378
CREATE HISTOGRAM TEMPLATE 383
CREATE INDEX 385
SCREATE INDEX EXTENSION 403
CREATE METHOD 410
CREATE NICKNAME 416
CREATE PROCEDURE 429
CREATE PROCEDURE (externo) 430
CREATE PROCEDURE (con fuente) 445
CREATE PROCEDURE (SQL) 450
CREATE ROLE 456
CREATE SCHEMA 457
CREATE SECURITY LABEL COMPONENT . . . 460
CREATE SECURITY LABEL 463
CREATE SECURITY POLICY 465
CREATE SEQUENCE 467
CREATE SERVICE CLASS 472
CREATE SERVER 480
CREATE TABLE 484
CREATE TABLESPACE 558
CREATE THRESHOLD 573
CREATE TRANSFORM 582
CREATE TRIGGER 586
CREATE TRUSTED CONTEXT 599
CREATE TYPE (matriz) 606
CREATE TYPE (diferenciado) 608
CREATE TYPE (estructurado) 614
CREATE TYPE MAPPING 639
CREATE USER MAPPING 646
CREATE VARIABLE 648
CREATE VIEW 651
CREATE WORK ACTION SET 666
CREATE WORK CLASS SET 674
CREATE WORKLOAD 679
CREATE WRAPPER 687
DECLARE CURSOR 689
DECLARE GLOBAL TEMPORARY TABLE . . . 695
DELETE 703
DESCRIBE 710
DESCRIBE INPUT 711
DESCRIBE OUTPUT 715
DISCONNECT 720
DROP 723
END DECLARE SECTION 756

© Copyright IBM Corp. 1993, 2010

EXECUTE IMMEDIATE 757
EXECUTE 760
EXPLAIN 767
FETCH 773
FLUSH EVENT MONITOR 776
FLUSH OPTIMIZATION PROFILE CACHE . . . 777
FLUSH PACKAGE CACHE 779
FOR 780
FREE LOCATOR 783
GET DIAGNOSTICS 784
GOTO 787
GRANT (autorizaciones de bases de datos) . . . 789
GRANT (Exención) 793
GRANT (privilegios de variable global) 796
GRANT (privilegios de índice) 798
GRANT (privilegios de paquete) 800
GRANT (Función) 803
GRANT (privilegios de rutina) 806
GRANT (privilegios de esquema) 810
GRANT (Etiqueta de seguridad) 813
GRANT (privilegios de secuencia) 816
GRANT (privilegios de servidor) 819
GRANT (privilegio SETSESSIONUSER) 821
GRANT (privilegios de espacio de tablas) 823
GRANT (privilegios de tabla, vista o apodo) . . . 825
GRANT (privilegios de carga de trabajo) 832
GRANT (privilegios de objeto XSR) 834
IF 835
INCLUDE 837
INSERT 839
ITERATE 849
LEAVE 851
LOCK TABLE 853
LOOP 855
MERGE 857
OPEN 867
PREPARE 872
REFRESH TABLE 882
RELEASE (conexión) 885
RELEASE SAVEPOINT 887
RENAME TABLESPACE 888
RENAME 890
REPEAT 892
RESIGNAL 894
RETURN 896
REVOKE (autorizaciones de bases de datos) . . . 898
REVOKE (Exención) 902
REVOKE (privilegios de variable global) 904
REVOKE (privilegios de índice) 907
REVOKE (privilegios de paquete) 909
REVOKE (Función) 912
REVOKE (privilegios de rutina) 915
REVOKE (privilegios de esquema) 919
REVOKE (Etiqueta de seguridad) 921
REVOKE (privilegios de secuencia) 923
REVOKE (privilegios de servidor) 925
REVOKE (privilegio SETSESSIONUSER) 927
REVOKE (privilegios de espacio de tablas) . . . 929
REVOKE (privilegios de tabla, vista o apodo) . . 931
REVOKE (privilegios de carga de trabajo) 937
REVOKE (privilegios de objeto XSR) 939
ROLLBACK 940
SAVEPOINT 943
SELECT INTO 946
SELECT 948
SET COMPILATION ENVIRONMENT 949
SET CONNECTION 950
SET CURRENT DECFLOAT ROUNDING MODE 952
SET CURRENT DEFAULT TRANSFORM GROUP 954
SET CURRENT DEGREE 956
SET CURRENT EXPLAIN MODE 958
SET CURRENT EXPLAIN SNAPSHOT 961
SET CURRENT FEDERATED ASYNCHRONY . . 964
SET CURRENT IMPLICIT XMLPARSE OPTION 966
SET CURRENT ISOLATION 967
SET CURRENT LOCK TIMEOUT 968
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 970
SET CURRENT MDC ROLLOUT MODE 972
SET CURRENT OPTIMIZATION PROFILE . . . 974
SET CURRENT PACKAGE PATH 977
SET CURRENT PACKAGESET 981
SET CURRENT QUERY OPTIMIZATION 983
SET CURRENT REFRESH AGE 986
SET ENCRYPTION PASSWORD 988
SET EVENT MONITOR STATE 989
SET INTEGRITY 991
SET PASSTHRU 1011
SET PATH 1013
SET ROLE 1015
SET SCHEMA 1016
SET SERVER OPTION 1018
SET SESSION AUTHORIZATION 1020
SET variable 1023
SIGNAL 1030
TRANSFER OWNERSHIP 1033
UPDATE 1048
VALUES INTO 1059
VALUES 1061
WHENEVER 1062
WHILE 1064

Apéndice A. Visión general de la información técnica de DB2 1067
Biblioteca técnica de DB2 en copia impresa o en formato PDF 1068
Pedido de manuales de DB2 en copia impresa 1070
Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos . . . 1071
Acceso a diferentes versiones del Centro de información de DB2 1072
Visualización de temas en su idioma preferido en el Centro de información de DB2 1072
Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet 1073
Guías de aprendizaje de DB2 1074
Información de resolución de problemas de DB2 1075
Términos y condiciones 1075

Apéndice B. Avisos 1077

Índice 1081

Acerca de este manual

El manual Consulta de SQL en dos volúmenes define el lenguaje SQL utilizado por la base de datosDB2 para Linux®, UNIX® y Windows®. Éste incluye:

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

Quién debe utilizar este manual

Este manual va dirigido a aquellas personas que deseen utilizar el Lenguaje de consulta estructurada (SQL) para acceder a una base de datos. Principalmente, es para los programadores y los administradores de bases de datos, pero también pueden utilizarlo los usuarios que accedan a las bases de datos mediante el procesador de línea de mandatos (CLP).

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

Cómo está estructurado este manual

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

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

© Copyright IBM Corp. 1993, 2010

Cómo leer los diagramas de sintaxis

La información siguiente le ayudará a comprender la estructura de los diagramas de sintaxis:

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

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

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

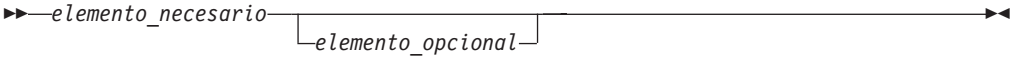
El símbolo ?--- indica que la sintaxis continúa de la línea anterior.

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

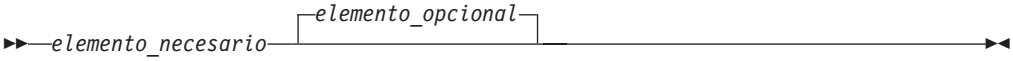
Los fragmentos de sintaxis empiezan con el símbolo +--- y finalizan con el símbolo ---¦.

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

Los elementos opcionales aparecen bajo la vía de acceso principal.

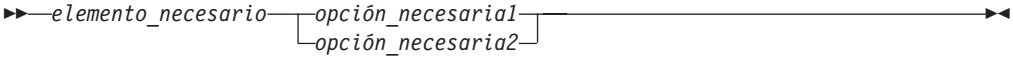


Si un elemento opcional aparece sobre la vía de acceso principal, ese elemento no tiene ningún efecto en la ejecución y sólo se utiliza para posibilitar la lectura.

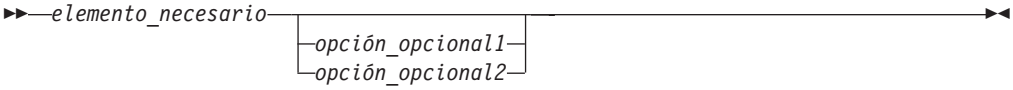


Si puede elegir entre dos o más elementos, éstos aparecen en una pila.

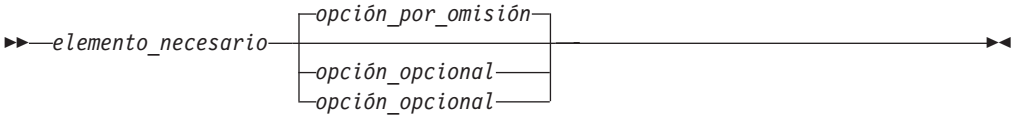
Si *debe* elegir uno de los elementos, un elemento de la pila aparece en la vía de acceso principal.



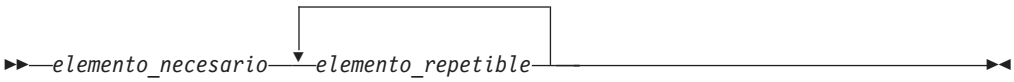
Si la elección de uno de los elementos es opcional, la pila entera aparece bajo la vía de acceso principal.



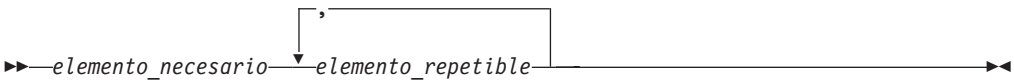
Si uno de los elementos es el valor por omisión, aparecerá sobre la vía de acceso principal y las opciones restantes se mostrarán debajo.



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



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

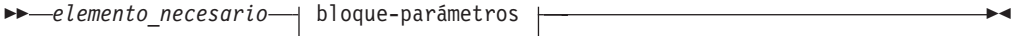


Una flecha de repetición sobre una pila indica que puede realizar más una elección en los elementos apilados o repetir una sola elección.

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

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

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



bloque-parámetros:

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

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

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

Convenios utilizados en este manual

Condiciones de error

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

```
Una signatura duplicada devuelve un error de SQL (SQLSTATE 42723).
```

Convenios de resaltado

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 resultarle útiles al preparar las aplicaciones:

- *Iniciación al desarrollo de aplicaciones de bases de datos*
  - Presenta el desarrollo de la aplicación DB2 e incluye los requisitos previos de la plataforma, el software de desarrollo soportado y una orientación sobre las ventajas y limitaciones de los API de programación soportados.
- *DB2 for i5/OS SQL Reference*
  - Este manual define el soporte para SQL de DB2 Query Manager y SQL Development Kit en System i. Contiene información de consulta para las tareas de administración del sistema, administración de la base de datos, programación de aplicaciones y operación. Este manual incluye sintaxis, notas acerca del uso, palabras claves y ejemplos para cada una de las sentencias de SQL utilizadas en sistemas i5/OS que ejecutan DB2.
- *DB2 for z/OS SQL Reference*
  - Este manual define el SQL utilizado en DB2 para z/OS. Proporciona formatos de consulta, sentencias de SQL, sentencias de procedimientos de SQL, límites de DB2, SQLCA, SQLDA, tablas de catálogos y palabras reservadas de SQL para sistemas z/OS que ejecutan DB2.
- *DB2 Spatial Extender User's Guide and Reference*

  - Este manual describe cómo escribir aplicaciones para crear y utilizar un sistema de información geográfica (GIS). Para crear y utilizar un GIS es necesario proporcionar una base de datos con recursos y luego consultar los datos para obtener información, tal como ubicaciones, distancias y distribuciones dentro de zonas geográficas.
- *IBM SQL Reference*
  - Este manual contiene todos los elementos comunes de SQL que están distribuidos por todos los productos de base de datos de IBM. Proporciona límites y normas que pueden servir de ayuda en la preparación de programas portátiles que utilicen bases de datos de IBM®. Este manual proporciona una lista de extensiones de SQL e incompatibilidades entre los siguientes estándares y productos: SQL92E, XPG4-SQL, IBM-SQL y los productos de bases de datos relacionales de IBM.
- *American National Standard X3.135-1992, Database Language SQL*
  - Contiene la definición estándar ANSI de SQL.
- *ISO/IEC 9075:1992, Database Language SQL*
  - Contiene la definición de SQL proporcionada por la norma ISO 1992.
- *ISO/IEC 9075-2:2003, Information technology -- Database Languages -- SQL -- Part 2: Foundation (SQL/Foundation)*
  - Contiene una gran parte de la definición de SQL proporcionada por la norma ISO 2003.
- *ISO/IEC 9075-4:2003, Information technology -- Database Languages -- SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
  - Contiene la definición de las sentencias de control de los procedimientos SQL, tal como aparece en la norma ISO 2003.

Documentación relacionada

Sentencias

© Copyright IBM Corp. 1993, 2010

Sentencias de SQL

Las tablas siguientes listan las sentencias de SQL clasificadas por tipo:

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

Tabla 1. Sentencias de esquema de SQL

Sentencia de SQL	Propósito
"ALTER AUDIT POLICY" en la página 25	Modifica la definición de una política de comprobación en el servidor actual.
"ALTER BUFFERPOOL" en la página 29	Cambia la definición de una agrupación de almacenamientos intermedios.
"ALTER DATABASE" en la página 36	Añade nuevas vías de acceso de almacenamiento a la colección de vías de acceso que se utilizan para los espacios de tablas de almacenamiento automático.
"ALTER DATABASE PARTITION GROUP" en la página 32	Cambia la definición de un grupo de particiones de base de datos.
"ALTER FUNCTION" en la página 40	Modifica una función existente cambiando las propiedades de la función.
"ALTER HISTOGRAM TEMPLATE" en la página 43	Modifica la plantilla que describe el tipo de histograma que se puede utilizar para alterar temporalmente uno o varios de los histogramas por omisión de una clase de servicio o de una clase de trabajo.
"ALTER METHOD" en la página 45	Modifica un método existente cambiando el cuerpo de método que se asocia con el método.
"ALTER NICKNAME" en la página 47	Cambia la definición de un apodo.
"ALTER PROCEDURE (externo)" en la página 56	Modifica un procedimiento externo existente cambiando las propiedades del procedimiento.
"ALTER PROCEDURE (con fuente)" en la página 59	Modifica un procedimiento con fuente existente cambiando el tipo de datos de uno o varios parámetros del procedimiento con fuente.
"ALTER PROCEDURE (SQL)" en la página 61	Modifica un procedimiento de SQL existente cambiando las propiedades del procedimiento.
"ALTER SECURITY LABEL COMPONENT" en la página 63	Modifica un componente de etiqueta de seguridad.
"ALTER SECURITY POLICY" en la página 66	Modifica una política de seguridad.
"ALTER SEQUENCE" en la página 70	Cambia la definición de una secuencia.
"ALTER SERVER" en la página 74	Cambia la definición de una fuente de datos en un sistema federado.
"ALTER SERVICE CLASS" en la página 78	Cambia la definición de una clase de servicio.
"ALTER TABLE" en la página 85	Cambia la definición de una tabla.
"ALTER TABLESPACE" en la página 133	Cambia la definición de un espacio de tablas.
"ALTER THRESHOLD" en la página 144	Cambia la definición de un umbral.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Propósito
"ALTER TRUSTED CONTEXT" en la página 151	Cambia la definición de un contexto fiable en el servidor actual.
"ALTER TYPE (estructurado)" en la página 160	Cambia la definición de un tipo estructurado.
"ALTER USER MAPPING" en la página 167	Cambia la definición de una correlación de autorizaciones de usuario.
"ALTER VIEW" en la página 169	Cambia la definición de una vista modificando una columna de tipo de referencia para añadir un ámbito.
"ALTER WORK ACTION SET" en la página 171	Añade, modifica o descarta acciones de trabajo dentro de un conjunto de acciones de trabajo.
"ALTER WORK CLASS SET" en la página 183	Añade, modifica o descarta clases de trabajo dentro de un conjunto de clases de trabajo.
"ALTER WORKLOAD" en la página 188	Cambia una carga de trabajo.
"ALTER WRAPPER" en la página 195	Actualiza las opciones que, junto con un módulo de derivador, se utilizan para acceder a las fuentes de datos de un tipo específico.
"ALTER XSROBJECT" en la página 197	Habilita o inhabilita el soporte de la descomposición para un esquema XML específico.
"AUDIT" en la página 200	Determina la política de comprobación que se debe utilizar para una base de datos o un objeto de base de datos en particular en el servidor actual.
"COMMENT" en la página 217	Sustituye o añade un comentario a la descripción de un objeto.
"CREATE ALIAS" en la página 264	Define un alias para una tabla, vista u otro alias.
"CREATE AUDIT POLICY" en la página 267	Define una política de comprobación en el servidor actual.
"CREATE BUFFERPOOL" en la página 271	Crea una nueva agrupación de almacenamientos intermedios.
"CREATE DATABASE PARTITION GROUP" en la página 275	Define un grupo de particiones de base de datos.
"CREATE EVENT MONITOR" en la página 278	Especifica sucesos de la base de datos que se deben supervisar.
"CREATE FUNCTION" en la página 299	Registra una función definida por el usuario.
"CREATE FUNCTION (escalar externa)" en la página 300	Registra una función escalar externa definida por el usuario.
"CREATE FUNCTION (tabla externa)" en la página 326	Registra una función de tabla externa definida por el usuario.
"CREATE FUNCTION (tabla externa OLE DB)" en la página 346	Registra una función de tabla externa OLE DB definida por el usuario.
"CREATE FUNCTION (con fuente o plantilla)" en la página 355	Registra una función con fuente definida por el usuario.
"CREATE FUNCTION (escalar de SQL, tabla o fila)" en la página 367	Registra y define una función SQL definida por el usuario.
"CREATE FUNCTION MAPPING" en la página 378	Define una correlación de funciones.
"CREATE HISTOGRAM TEMPLATE" en la página 383	Define una plantilla que describe el tipo de histograma que se puede utilizar para alterar temporalmente uno o varios de los histogramas por omisión de una clase de servicio o de una clase de trabajo.
"CREATE INDEX" en la página 385	Define un índice para una tabla.
"SCREATE INDEX EXTENSION" en la página 403	Define un objeto de extensión para su uso con índices sobre tablas con columnas de tipo estructurado o diferenciado.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Propósito
"CREATE METHOD" en la página 410	Asocia un cuerpo de método con una especificación de método definida previamente.
"CREATE NICKNAME" en la página 416	Define un apodo.
"CREATE PROCEDURE" en la página 429	Registra un procedimiento.
"CREATE PROCEDURE (externo)" en la página 430	Registra un procedimiento externo.
"CREATE PROCEDURE (con fuente)" en la página 445	Registra un procedimiento (procedimiento con fuente) que está basado en otro procedimiento (procedimiento fuente). En sistemas federados, se entiende por procedimiento federado un procedimiento con fuente cuyo procedimiento fuente se encuentra en una fuente de datos soportada.
"CREATE PROCEDURE (SQL)" en la página 450	Registra un procedimiento de SQL.
"CREATE ROLE" en la página 456	Define un rol en el servidor actual.
"CREATE SCHEMA" en la página 457	Define un esquema.
"CREATE SECURITY LABEL COMPONENT" en la página 460	Crea un componente que se utilizará como parte de una política de seguridad.
"CREATE SECURITY LABEL" en la página 463	Crea una etiqueta de seguridad.
"CREATE SECURITY POLICY" en la página 465	Crea una política de seguridad.
"CREATE SEQUENCE" en la página 467	Define una secuencia.
"CREATE SERVER" en la página 480	Define una fuente de datos para una base de datos federada.
"CREATE SERVICE CLASS" en la página 472	Define una clase de servicio.
"CREATE TABLE" en la página 484	Define una tabla.
"CREATE TABLESPACE" en la página 558	Define un espacio de tablas.
"CREATE THRESHOLD" en la página 573	Define un umbral.
"CREATE TRANSFORM" en la página 582	Define funciones de transformación.
"CREATE TRIGGER" en la página 586	Define un activador.
"CREATE TRUSTED CONTEXT" en la página 599	Define un contexto fiable en el servidor actual.
"CREATE TYPE (matriz)" en la página 606	Define un tipo de matriz.
"CREATE TYPE (diferenciado)" en la página 608	Define un tipo de datos diferenciado.
"CREATE TYPE (estructurado)" en la página 614	Define un tipo de datos estructurado.
"CREATE TYPE MAPPING" en la página 639	Define una correlación entre tipos de datos.
"CREATE USER MAPPING" en la página 646	Define una correlación entre autorizaciones de usuario.
"CREATE VARIABLE" en la página 648	Define una variable global.
"CREATE VIEW" en la página 651	Define una vista de una o más tablas, vistas o apodos.
"CREATE WORK ACTION SET" en la página 666	Define un conjunto de acciones de trabajo y las acciones de trabajo contenidas en el conjunto de acciones de trabajo.
"CREATE WORK CLASS SET" en la página 674	Define un conjunto de clases de trabajo.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Propósito
"CREATE WORKLOAD" en la página 679	Define una carga de trabajo.
"CREATE WRAPPER" en la página 687	Registra un derivador.
"DROP" en la página 723	Suprime objetos de la base de datos.
"GRANT (autorizaciones de bases de datos)" en la página 789	Otorga autorizaciones sobre toda una base de datos.
"GRANT (Exención)" en la página 793	Otorga una exención sobre una norma de acceso para una política de seguridad LBAC (control de acceso basado en etiquetas) especificada.
"GRANT (privilegios de variable global)" en la página 796	Otorga uno o más privilegios en una variable global creada.
"GRANT (privilegios de índice)" en la página 798	Otorga el privilegio CONTROL en índices en la base de datos.
"GRANT (privilegios de paquete)" en la página 800	Otorga privilegios para paquetes de la base de datos.
"GRANT (Función)" en la página 803	Otorga roles a usuarios, grupos o a otros roles.
"GRANT (privilegios de rutina)" en la página 806	Otorga privilegios para una rutina (función, método o procedimiento).
"GRANT (privilegios de esquema)" en la página 810	Otorga privilegios para un esquema.
"GRANT (Etiqueta de seguridad)" en la página 813	Otorga una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC) para un acceso de lectura, acceso de grabación o para ambos tipos de acceso.
"GRANT (privilegios de secuencia)" en la página 816	Otorga privilegios en una secuencia.
"GRANT (privilegios de servidor)" en la página 819	Otorga privilegios para consultar una fuente de datos específica.
"GRANT (privilegio SETSESSIONUSER)" en la página 821	Otorga el privilegio de utilización de la sentencia SET SESSION AUTHORIZATION.
"GRANT (privilegios de espacio de tablas)" en la página 823	Otorga privilegios para un espacio de tablas.
"GRANT (privilegios de tabla, vista o apodo)" en la página 825	Otorga privilegios para tablas, vistas y apodos.
"GRANT (privilegios de carga de trabajo)" en la página 832	Otorga el privilegio USAGE en una carga de trabajo.
"GRANT (privilegios de objeto XSR)" en la página 834	Otorga el privilegio USAGE en un objeto XSR.
"REFRESH TABLE" en la página 882	Renueva los datos de una tabla de consultas materializadas.
"RENAME" en la página 890	Cambia el nombre de una tabla existente.
"RENAME TABLESPACE" en la página 888	Cambia el nombre de un espacio de tablas existente.
"REVOKE (autorizaciones de bases de datos)" en la página 898	Revoca autorizaciones de toda una base de datos.
"REVOKE (Exención)" en la página 902	Revoca la exención sobre una norma de acceso para una política de seguridad LBAC (control de acceso basado en etiquetas).
"REVOKE (privilegios de variable global)" en la página 904	Revoca uno o más privilegios en una variable global creada.
"REVOKE (privilegios de índice)" en la página 907	Revoca el privilegio CONTROL en índices determinados.

Tabla 1. Sentencias de esquema de SQL (continuación)

Sentencia de SQL	Propósito
"REVOKE (privilegios de paquete)" en la página 909	Revoca los privilegios de paquetes determinados en la base de datos.
"REVOKE (Función)" en la página 912	Revoca roles de usuarios, grupos o a otros roles.
"REVOKE (privilegios de rutina)" en la página 915	Revoca privilegios para una rutina (función, método o procedimiento).
"REVOKE (privilegios de esquema)" en la página 919	Revoca privilegios para un esquema.
"REVOKE (Etiqueta de seguridad)" en la página 921	Revoca una etiqueta de seguridad LBAC (control de acceso basado en etiquetas) para un acceso de lectura, acceso de grabación o ambos tipos de acceso.
"REVOKE (privilegios de secuencia)" en la página 923	Revoca los privilegios de una secuencia.
"REVOKE (privilegios de servidor)" en la página 925	Revoca privilegios para consultar una fuente de datos específica.
"REVOKE (privilegio SETSESSIONUSER)" en la página 927	Revoca el privilegio de utilizar la sentencia SET SESSION AUTHORIZATION.
"REVOKE (privilegios de espacio de tablas)" en la página 929	Revoca el privilegio de utilización (USE) para un espacio de tablas determinado.
"REVOKE (privilegios de tabla, vista o apodo)" en la página 931	Revoca privilegios para determinadas tablas, vistas o apodos.
"REVOKE (privilegios de carga de trabajo)" en la página 937	Revoca el privilegio USAGE en una carga de trabajo.
"REVOKE (privilegios de objeto XSR)" en la página 939	Revoca el privilegio USAGE en un objeto XSR.
"SET INTEGRITY" en la página 991	Establece el estado Establecer integridad pendiente y comprueba los datos para detectar violaciones de restricción.
"TRANSFER OWNERSHIP" en la página 1033	Transfiere la propiedad de un objeto de base de datos.

Tabla 2. Sentencias de cambio de datos de SQL

Sentencia de SQL	Propósito
"DELETE" en la página 703	Suprime una o más filas de una tabla.
"INSERT" en la página 839	Inserta una o más filas en una tabla.
"MERGE" en la página 857	Actualiza un destino (tabla o vista) utilizando los datos de una fuente (resultado de una referencia de tabla).
"UPDATE" en la página 1048	Actualiza los valores de una o varias columnas en una o más filas de una tabla.

Tabla 3. Sentencias de datos de SQL

Sentencia de SQL	Propósito
"ALLOCATE CURSOR" en la página 23	Asigna un cursor para el conjunto de resultados identificados por la variable del localizador de conjunto de resultados.
"ASSOCIATE LOCATORS" en la página 198	Obtiene el valor del localizador para cada conjunto de resultados devuelto por un procedimiento.
"CLOSE" en la página 215	Cierra un cursor.
"DECLARE CURSOR" en la página 689	Define un cursor SQL.

Tabla 3. Sentencias de datos de SQL (continuación)

Sentencia de SQL	Propósito
"DELETE" en la página 703	Suprime una o más filas de una tabla.
"FETCH" en la página 773	Asigna valores de una fila a variables del lenguaje principal.
"FLUSH EVENT MONITOR" en la página 776	Graba el almacenamiento intermedio interno activo de un supervisor de sucesos.
"FLUSH PACKAGE CACHE" en la página 779	Elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes.
"FREE LOCATOR" en la página 783	Elimina la asociación entre una variable localizadora y su valor.
"INSERT" en la página 839	Inserta una o más filas en una tabla.
"LOCK TABLE" en la página 853	Impide que los procesos simultáneos cambien una tabla o impide que los procesos simultáneos utilicen una tabla.
"MERGE" en la página 857	Actualiza un destino (tabla o vista) utilizando los datos de una fuente (resultado de una referencia de tabla).
"OPEN" en la página 867	Prepara un cursor que se utilizará para recuperar valores cuando se emita la sentencia FETCH.
"SELECT INTO" en la página 946	Especifica una tabla de resultados que no tiene más de una fila y asigna los valores a variables del lenguaje principal.
"SET variable" en la página 1023	Asigna valores a variables de transición NEW.
"UPDATE" en la página 1048	Actualiza los valores de una o varias columnas en una o más filas de una tabla.
"VALUES INTO" en la página 1059	Especifica una tabla de resultados que no tiene más de una fila y asigna los valores a variables del lenguaje principal.

Tabla 4. Sentencias de transacciones de SQL

Sentencia de SQL	Propósito
"COMMIT" en la página 229	Finaliza una unidad de trabajo y confirma los cambios que esa unidad de trabajo ha realizado en la base de datos.
"RELEASE SAVEPOINT" en la página 887	Libera un punto de salvaguarda dentro de una transacción.
"ROLLBACK" en la página 940	Termina una unidad de trabajo y restituye los cambios realizados por dicha unidad de trabajo.
"SAVEPOINT" en la página 943	Define un punto de salvaguarda dentro de una transacción.

Tabla 5. Sentencias de conexión de SQL

Sentencia de SQL	Propósito
"CONNECT (Tipo 1)" en la página 249	Conecta a un servidor de aplicaciones según las normas para una unidad de trabajo remota.
"CONNECT (Tipo 2)" en la página 256	Conecta a un servidor de aplicaciones según las normas para la unidad de trabajo distribuida dirigida por aplicación.
"DISCONNECT" en la página 720	Finaliza una o más conexiones cuando no hay ninguna unidad de trabajo activa.
"RELEASE (conexión)" en la página 885	Coloca una o más conexiones en el estado pendiente de liberación.
"SET CONNECTION" en la página 950	Cambia el estado de una conexión de inactivo a actual, haciendo que la ubicación especificada sea el servidor actual.

Sentencias de SQL

Tabla 6. Sentencias dinámicas de SQL

Sentencia de SQL	Propósito
"DESCRIBE" en la página 710	Obtiene información acerca de un objeto.
"DESCRIBE INPUT" en la página 711	Obtiene información sobre los marcadores de parámetro de entrada de una sentencia preparada.
"DESCRIBE OUTPUT" en la página 715	Obtiene información sobre una sentencia preparada o información sobre las columnas de lista de selección de una sentencia SELECT preparada.
"EXECUTE" en la página 760	Ejecuta una sentencia de SQL preparada.
"EXECUTE IMMEDIATE" en la página 757	Prepara y ejecuta una sentencia de SQL.
"PREPARE" en la página 872	Prepara una sentencia de SQL (con parámetros opcionales) para su ejecución.

Tabla 7. Sentencias de sesiones de SQL

Sentencia de SQL	Propósito
"DECLARE GLOBAL TEMPORARY TABLE" en la página 695	Define la Tabla Temporal Global.
"EXPLAIN" en la página 767	Captura información acerca del plan de acceso elegido.
"SET COMPILATION ENVIRONMENT" en la página 949	Cambia el entorno de compilación actual en la conexión para que coincida con los valores incluidos en el entorno de compilación proporcionado por un supervisor de sucesos de punto muerto.
"SET CURRENT DECFLOAT ROUNDING MODE" en la página 952	Verifica que la modalidad de redondeo especificada sea el valor que está establecido actualmente para el registro especial CURRENT DECFLOAT ROUNDING MODE.
"SET CURRENT DEFAULT TRANSFORM GROUP" en la página 954	Cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP.
"SET CURRENT DEGREE" en la página 956	Cambia el valor del registro especial CURRENT DEGREE.
"SET CURRENT EXPLAIN MODE" en la página 958	Cambia el valor del registro especial CURRENT EXPLAIN MODE.
"SET CURRENT EXPLAIN SNAPSHOT" en la página 961	Cambia el valor del registro especial CURRENT EXPLAIN SNAPSHOT.
"SET CURRENT FEDERATED ASYNCHRONY" en la página 964	Cambia el valor del registro especial CURRENT FEDERATED ASYNCHRONY.
"SET CURRENT IMPLICIT XMLPARSE OPTION" en la página 966	Cambia el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION.
"SET CURRENT ISOLATION" en la página 967	Cambia el valor del registro especial CURRENT ISOLATION.
"SET CURRENT LOCK TIMEOUT" en la página 968	Cambia el valor del registro especial CURRENT LOCK TIMEOUT.
"SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION" en la página 970	Cambia el valor del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.
"SET CURRENT MDC ROLLOUT MODE" en la página 972	Asigna un valor al registro especial CURRENT MDC ROLLOUT MODE.
"SET CURRENT OPTIMIZATION PROFILE" en la página 974	Asigna un valor al registro especial CURRENT OPTIMIZATION PROFILE.
"SET CURRENT PACKAGE PATH" en la página 977	Asigna un valor al registro especial CURRENT PACKAGE PATH.

Tabla 7. Sentencias de sesiones de SQL (continuación)

Sentencia de SQL	Propósito
"SET CURRENT PACKAGESET" en la página 981	Establece el nombre de esquema para la selección de paquetes.
"SET CURRENT QUERY OPTIMIZATION" en la página 983	Cambia el valor del registro especial CURRENT QUERY OPTIMIZATION.
"SET CURRENT REFRESH AGE" en la página 986	Cambia el valor del registro especial CURRENT REFRESH AGE.
"SET ENCRYPTION PASSWORD" en la página 988	Establece la contraseña para el cifrado.
"SET EVENT MONITOR STATE" en la página 989	Activa o desactiva un supervisor de sucesos.
"SET PASSTHRU" en la página 1011	Abre una sesión para someter SQL nativo de fuente de datos directamente a la fuente de datos.
"SET PATH" en la página 1013	Cambia el valor del registro especial CURRENT PATH.
"SET ROLE" en la página 1015	Verifica que el ID de autorización de la sesión sea un miembro de un rol específico.
"SET SCHEMA" en la página 1016	Cambia el valor del registro especial CURRENT SCHEMA.
"SET SERVER OPTION" en la página 1018	Establece valores de opciones del servidor.
"SET SESSION AUTHORIZATION" en la página 1020	Cambia el valor del registro especial SESSION USER.

Tabla 8. Sentencias de lenguaje principal de SQL incorporadas

Sentencia de SQL	Propósito
"BEGIN DECLARE SECTION" en la página 204	Marca el principio de una sección de declaración de variables del lenguaje principal.
"END DECLARE SECTION" en la página 756	Marca el final de una sección de declaración de variables del lenguaje principal.
"GET DIAGNOSTICS" en la página 784	Se utiliza para obtener información acerca de la sentencia de SQL ejecutada previamente.
"INCLUDE" en la página 837	Inserta código o declaraciones en un programa fuente.
"RESIGNAL" en la página 894	Se utiliza para retransmitir una condición de error o aviso.
"SIGNAL" en la página 1030	Se utiliza para transmitir una condición de error o aviso.
"WHENEVER" en la página 1062	Define las acciones que se deben tomar sobre la base de los códigos de retorno SQL.

Tabla 9. Sentencias de control de SQL

Sentencia de SQL	Propósito
"CALL" en la página 206	Llama a un procedimiento.
"CASE" en la página 212	Selecciona una vía de acceso de ejecución basándose en múltiples condiciones.
"SQL compuesto (dinámico)" en la página 231	Combina una o más sentencias de SQL diferentes en un bloque dinámico.
"de SQL compuesto (incorporado)" en la página 236	Combina una o varias sentencias de SQL para formar un bloque ejecutable.
"SQL compuesto (procedimiento)" en la página 240	Agrupa otras sentencias en un procedimiento de SQL.

Tabla 9. Sentencias de control de SQL (continuación)

Sentencia de SQL	Propósito
"FOR" en la página 780	Ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.
"GOTO" en la página 787	Se utiliza para ramificar a una etiqueta definida por el usuario dentro de un procedimiento de SQL.
"IF" en la página 835	Selecciona una vía de acceso de ejecución basándose en la evaluación de una condición.
"ITERATE" en la página 849	Provoca que el flujo de control vuelva al principio de un bucle con etiqueta.
"LEAVE" en la página 851	Transfiere el control del programa fuera de un bucle o de una sentencia compuesta.
"LOOP" en la página 855	Repite la ejecución de una sentencia o grupo de sentencias.
"REPEAT" en la página 892	Ejecuta una sentencia o grupo de sentencias hasta que una condición de búsqueda es verdadera.
"RESIGNAL" en la página 894	Se utiliza para retransmitir una condición de error o aviso.
"RETURN" en la página 896	Se utiliza para volver de una rutina.
"SIGNAL" en la página 1030	Se utiliza para transmitir una condición de error o aviso.
"WHILE" en la página 1064	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 de SQL.
- Preparada y ejecutada dinámicamente
- Emitida interactivamente

Se pueden utilizar algunos o todos estos métodos, dependiendo de la sentencia. (Las sentencias incorporadas en REXX se preparan y se 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.

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.

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

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

Preparación y ejecución dinámicas

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

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

La ejecución satisfactoria o no satisfactoria de la sentencia se indica por medio de un código de retorno de SQL en la SQLCA tras completarse la sentencia EXECUTE (o EXECUTE IMMEDIATE). El código de retorno de SQL debe comprobarse, tal como se describía anteriormente. Para obtener más información, consulte el apartado “Códigos de retorno de SQL (SQLCODE y SQLSTATE)” en la página 13.

Invocación estática de una sentencia de selección

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

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

Invocación dinámica de una sentencia de selección

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

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

Invocación interactiva

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

Utilización de SQL con otros sistemas principales

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

Se puede encontrar información adicional sobre las sentencias de SQL utilizadas en otros sistemas principales en la publicación *DB2 for i5/OS SQL Reference* y la publicación *DB2 for z/OS SQL Reference*.

Códigos de retorno de SQL (SQLCODE y SQLSTATE)

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

- Incluir una estructura llamada SQLCA. La SQLCA incluye una variable entera llamada SQLCODE y una variable de serie de caracteres llamada SQLSTATE. En REXX, se proporciona automáticamente una SQLCA. En otros lenguajes, la SQLCA se puede obtener utilizando la sentencia INCLUDE SQLCA.
- Si LANGLEVEL SQL92E se ha especificado como opción de precompilación, puede declararse una variable denominada SQLCODE o SQLSTATE en la

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

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

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

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

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

Comentarios de SQL

Las sentencias de SQL estático pueden incluir comentarios de SQL o del lenguaje principal. Las sentencias de SQL dinámico pueden incluir comentarios de SQL. Existen dos tipos de comentarios de SQL:

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

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

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

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

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

- El /* debe estar en la misma línea y no debe estar separado por un espacio.
- El */ debe estar en la misma línea y no debe estar separado por un espacio.

- Los comentarios compuestos pueden empezar dondequiera que un espacio sea válido (excepto dentro de un símbolo delimitador o entre 'EXEC' y 'SQL').
- Los comentarios compuestos pueden continuar en las líneas posteriores.

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

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

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

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

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.

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

Puede hacerse referencia a los parámetros SQL, las variables de SQL y a las variables globales en cualquier punto de la sentencia de procedimiento de SQL donde pueda especificarse una expresión o una variable. No pueden especificarse variables del lenguaje principal en las rutinas de SQL, activadores de SQL o sentencias compuestas dinámicas. Puede hacerse referencia a los parámetros de SQL en cualquier punto 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. Puede hacerse referencia a las variables globales de cualquier expresión siempre y cuando la expresión no tenga que ser determinante. Los escenarios siguientes requieren expresiones determinantes, que impiden el uso de variables globales:

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

Todos los parámetros de SQL, las variables de SQL y las variables globales se consideran anulables. El nombre de un parámetro de SQL, de una variable de SQL o de una variable global de una rutina de SQL puede ser igual al nombre de una columna de una tabla o vista a la que se haga referencia en la rutina. El nombre de una variable de SQL también puede ser el mismo que el de otra variable de SQL

declarada en la misma rutina. Esto puede ocurrir cuando las dos variables de SQL están declaradas en sentencias compuestas diferentes. La sentencia compuesta que contiene la declaración del nombre de una variable de SQL determina el ámbito del nombre de dicha variable. Para obtener más información, consulte "SQL compuesto (procedimiento)".

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

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

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

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

Referencias a etiquetas

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

Sólo se puede hacer referencia al nombre de una etiqueta dentro de la sentencia compuesta en la que está definida, incluidas las sentencias compuestas anidadas dentro de dichas sentencia compuesta. Una etiqueta se puede utilizar para calificar

el nombre de una variable SQL o se puede especificar como destino de una sentencia GOTO, LEAVE, o ITERATE.

Referencias a nombres de condición de SQL

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

Referencias a nombres de sentencia de SQL

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

Referencias a nombre de cursor de SQL

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

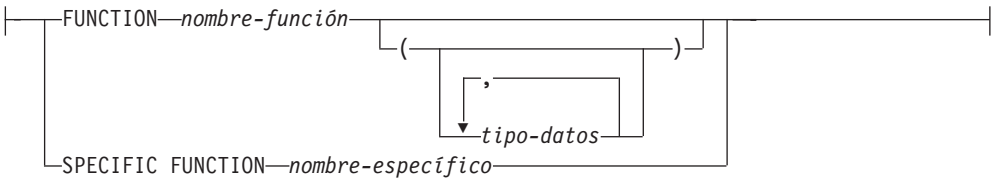
Designadores de función, método y procedimiento

Las secciones siguientes describen fragmentos de sintaxis que se utilizan para identificar, de manera exclusiva, una función, método o procedimiento.

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).

designador-función:



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

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

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

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

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

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

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

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

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

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

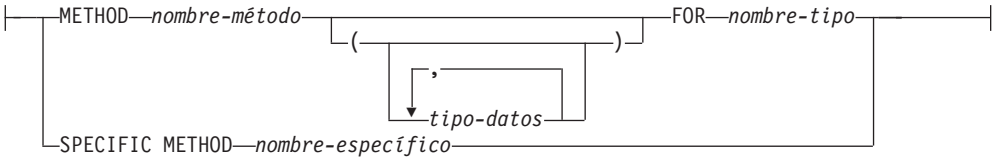
Si en el esquema nombrado o implícito no hay ninguna función con la signatura especificada, se genera un error (SQLSTATE 42883).

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

Designador de método

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

designador-método:



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

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

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

(tipo-datos,...)
Los valores deben coincidir con los tipos de datos que se han especificado (en la posición correspondiente) en la sentencia CREATE TYPE. Para

identificar la instancia de método específica, se utilizan el número de tipos de datos y la concatenación lógica de los tipos de datos.

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

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

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

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

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

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

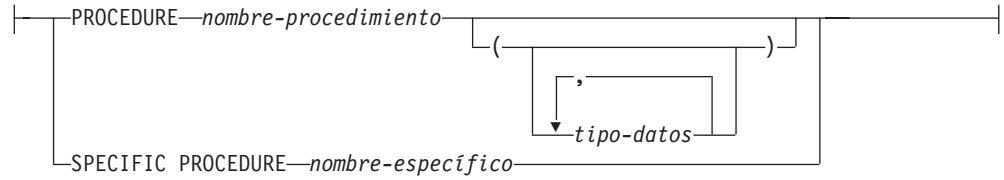
FOR *nombre-tipo*
Especifica el nombre del tipo al que va a asociarse el método especificado. El nombre debe identificar un tipo que ya esté descrito en el catálogo (SQLSTATE 42704). En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto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

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

Designador de procedimiento

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

designador-procedimiento:



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

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

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

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

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

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

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

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

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

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

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

ALLOCATE CURSOR

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

Invocación

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

Autorización

No se necesita.

Sintaxis

Descripción

nombre-cursor
Indica el cursor. El nombre no debe identificar un cursor que ya esté declarado en el procedimiento de 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 de SQL fuente, de acuerdo con las normas para declarar variables localizadoras de conjuntos de resultados. Si desea más información acerca de la declaración de variables de SQL, consulte "Sentencia de SQL compuesto (procedimiento)".

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

Normas

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

Ejemplos

Este ejemplo de procedimiento de 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 de SQL:

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

ALTER AUDIT POLICY

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

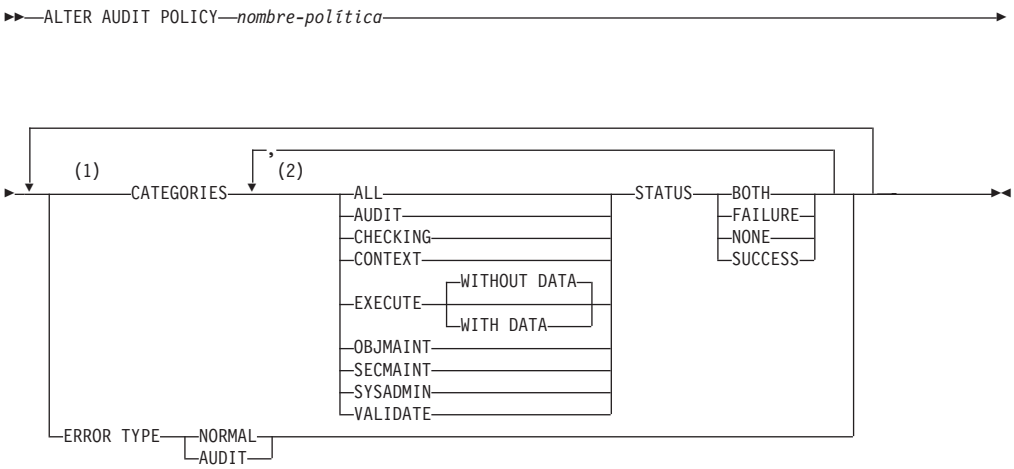
Invocación

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

Autorización

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

Sintaxis



Notas:

1 Cada una de las cláusulas CATEGORIES y ERROR TYPE puede especificarse una vez como máximo (SQLSTATE 42614).

2 Cada categoría puede especificarse una vez como máximo (SQLSTATE 42614) y no puede especificarse ninguna otra categoría si se especifica ALL (SQLSTATE 42601).

Descripción

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

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

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

AUDIT
Genera registros cuando se han cambiado valores de comprobación o cuando se accede al archivo de anotaciones cronológicas de auditoría.

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

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

EXECUTE
Genera registros para mostrar la ejecución de sentencias de SQL.

WITHOUT DATA o WITH DATA
Especifica si se proporcionan valores de datos de entrada para las variables de lenguaje principal y deben registrarse marcadores de parámetro como parte de la categoría EXECUTE.

WITHOUT DATA
Se proporcionan valores de datos de entrada para las variables de lenguaje principal y no se registran marcadores de parámetro como parte de la categoría EXECUTE.

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

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

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

SYSADMIN
Genera registros cuando se realicen operaciones que necesitan autorización SYSADM, SYSMAINT o SYSCTRL.

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

STATUS
Especifica un estado para la categoría especificada.

BOTH
Se comprobarán los sucesos satisfactorios y anómalos.

FAILURE
Sólo se comprobarán los sucesos anómalos.

SUCCESS
Sólo se comprobarán los sucesos satisfactorios.

NONE
No se comprobarán los sucesos de esta categoría.

ERROR TYPE
Especifica si van a devolverse o ignorarse los errores de comprobación.

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

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

Normas

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

Notas

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

Ejemplo

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

ALTER AUDIT POLICY

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

ALTER BUFFERPOOL

La sentencia ALTER BUFFERPOOL se utiliza para realizar lo siguiente:

- Modificar el tamaño de al agrupación de almacenamientos intermedios en todas las particiones de base de datos o en una única partición de base de datos
- Habilitar o inhabilitar el redimensionamiento automático de la agrupación de almacenamientos intermedios
- Añadir esta definición de agrupación de almacenamientos intermedios a un nuevo grupo de particiones de base de datos
- Modificar el área de bloque de la agrupación de almacenamientos intermedios para la E/S basada en bloques

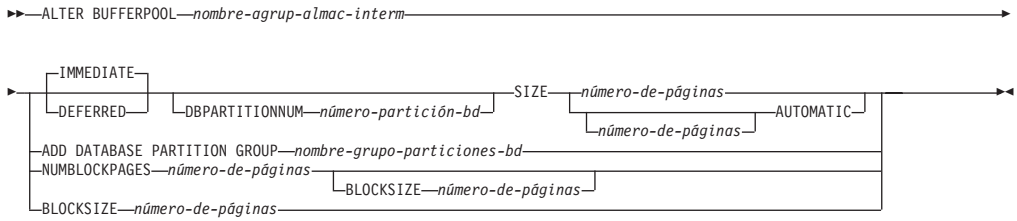
Invocación

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

Autorización

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

Sintaxis



Descripción

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

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

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

DEFERRED
El tamaño de la agrupación de almacenamientos intermedios cambiará al reactivar la base de datos (todas las aplicaciones deben desconectarse de la

base de datos). No es necesario tener espacio de memoria reservado; DB2 asignará la memoria necesaria del sistema en el momento de la activación.

DBPARTITIONNUM *número-partición-base-de-datos*
Especifica la partición de base de datos en la que se modifica el tamaño de la agrupación de almacenamientos intermedios. Se crea una entrada de excepción en la vista de catálogo del sistema SYSCAT.BUFFERPOOLDBPARTITIONS. La partición de base de datos debe encontrarse en uno de los grupos de particiones de base de datos para la agrupación de almacenamientos intermedios (SQLSTATE 42729). Si la cláusula no se especifica, el tamaño de la agrupación de almacenamientos intermedios se modifica en todas las particiones de bases de datos con la excepción de aquellas que presentan una entrada de excepción en SYSCAT.BUFFERPOOLDBPARTITIONS.

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

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

AUTOMATIC
Habilita el ajuste automático para la agrupación de almacenamientos intermedios. El gestor de bases de datos ajusta el tamaño de la agrupación de almacenamientos intermedios como respuesta a los requisitos de carga de trabajo. Si se especifica AUTOMATIC, la cláusula DBPARTITIONNUM no puede especificarse (SQLSTATE 42601).

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

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

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

- 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 las tablas de catálogo durante la confirmación, ningún cambio realizado en la agrupación de almacenamientos intermedios real entrará en vigor hasta la próxima vez que se inicie la base de datos. Los atributos actuales de la agrupación de almacenamientos intermedios existirán hasta entonces y no afectará a la agrupación de almacenamientos intermedios mientras tanto. Las tablas que se han creado en los espacios de tablas de los nuevos grupos de particiones de base de datos utilizarán la agrupación de almacenamientos intermedios por omisión. Por omisión, la sentencia es IMMEDIATE cuando se aplica dicha palabra clave.
- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones.
- ***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

ALTER DATABASE PARTITION GROUP

La sentencia ALTER DATABASE PARTITION GROUP se utiliza para:

- Añadir una o más particiones de base de datos a un grupo de particiones de base de datos
- Desactivar una o más particiones de base de datos de un grupo de particiones de base de datos

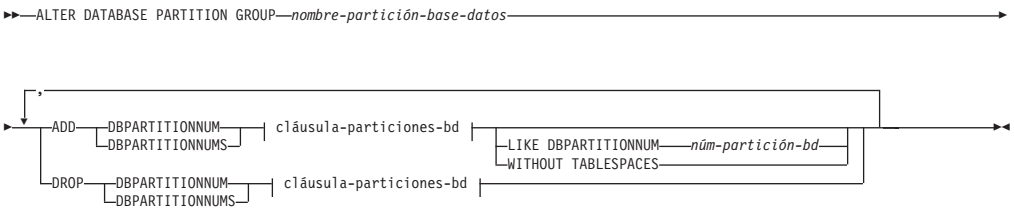
Invocación

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

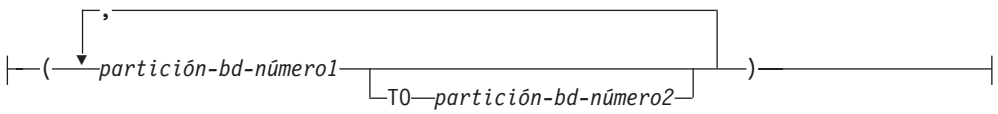
Autorización

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

Sintaxis



cláusula-particiones-bd:



Descripción

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

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

DROP DBPARTITIONNUM
Especifica las particiones de base de datos específicas que deben desactivarse del grupo de particiones de base de datos. DBPARTITIONNUMS es sinónimo

de DBPARTITIONNUM. Las particiones de base de datos especificadas deben estar definidas en el grupo de particiones de base de datos (SQLSTATE 42729).

cláusula-particiones-bd
Especifica las particiones de base de datos que deben añadirse o desactivarse.

partición-bd-núm-1
Especifica un número de partición de base de datos específica.

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

LIKE DBPARTITIONNUM *núm_particiónbd*
Especifica que los contenedores de los espacios de tablas existentes del grupo de particiones de base de datos serán iguales a los contenedores del *número-partición-base-de-datos* especificado. La partición de base de datos especificada debe ser una partición que ya 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.

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

WITHOUT TABLESPACES
Especifica que los contenedores para los espacios de tablas existentes en el grupo de particiones de base de datos no se crean en las particiones de base de datos añadidas recientemente. La sentencia ALTER TABLESPACE que utiliza la *cláusula-particiones-bd* debe utilizarse para definir los contenedores que deben utilizarse con los espacios de tablas que se han definido en este grupo de particiones de base de datos. Si no se especifica esta opción, los contenedores por omisión se especifican en las particiones de base de datos añadidas recientemente para cada espacio de tablas definido en el grupo de particiones de base de datos.

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

Normas

- Cada partición de base de datos especificada por número debe estar definida en el archivo `db2nodes.cfg` (SQLSTATE 42729).

- Cada *númpartiónbd* listado en la *cláusula-particiones-bd* debe ser para una partición de base de datos exclusiva (SQLSTATE 42728).
- Un número de partición de base de datos válido oscila entre 0 y 999, ambos inclusive (SQLSTATE 42729).
- Una partición de base de datos no puede aparecer en la cláusula ADD y en la cláusula DROP al mismo tiempo (SQLSTATE 42728).
- Debe haber como mínimo una partición de base de datos que permanezca en el grupo de particiones de base de datos. La última partición de base de datos de un grupo de particiones de base de datos no puede desactivarse (SQLSTATE 428C0).
- Si no se especifica la cláusula LIKE DBPARTITIONNUM ni la cláusula WITHOUT TABLESPACES al añadir una partición de base de datos, por omisión se utiliza el número de partición de base de datos más bajo de las particiones de base de datos existentes en el grupo de particiones de base de datos (es decir, el 2) y se procede como si se hubiera especificado LIKE DBPARTITIONNUM 2. Para que una partición de base de datos existente se utilice como partición por omisión, debe tener contenedores definidos para todos los espacios de tablas del grupo de particiones de base de datos (la columna IN_USE de SYSCAT.DBPARTITIONGROUPDEF no es 'T').

Notas

- Cuando se añade una partición de base de datos a un grupo de particiones de base de datos, se crea una entrada de catálogo para la partición de base de datos (consulte SYSCAT.DBPARTITIONGROUPDEF). El mapa de distribución cambia inmediatamente para incluir la nueva partición de base de datos, junto con un indicador (IN_USE) que indica que la partición de base de datos se encuentra en el mapa de distribución si se cumple una de las condiciones siguientes:
  - No se ha definido ningún espacio de tablas para el grupo de particiones de base de datos, o bien
  - No se ha definido ninguna tabla en los espacios de tablas definidos en el grupo de particiones de base de datos y no se ha especificado la cláusula WITHOUT TABLESPACES.

  El mapa de distribución no cambia y el indicador (IN_USE) se establece para indicar que la partición de base de datos no se incluye en el mapa de distribución si se cumple una de las condiciones siguientes:
  - Existen tablas en los espacios de tablas en el grupo de particiones de base de datos, o bien
  - Existen espacios de tablas en el grupo de particiones de base de datos y se ha especificado la cláusula WITHOUT TABLESPACES (a menos que todos los espacios de tablas estén definidos para utilizar el almacenamiento automático, en cuyo caso la cláusula WITHOUT TABLESPACES se pasa por alto)

  Para cambiar el mapa de distribución, debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP. Este mandato redistribuye los datos, cambia el mapa de distribución y cambia el indicador. Si se especifica la cláusula WITHOUT TABLESPACES, antes de intentar redistribuir los datos deben añadirse contenedores de espacios de tablas.
- Cuando se desactiva una partición de base de datos de un grupo de particiones de base de datos, la entrada de catálogo para la partición de base de datos (consulte SYSCAT.DBPARTITIONGROUPDEF) se actualiza. Si no hay tablas definidas en los espacios de tablas definidos en el grupo de particiones de base de datos, el mapa de distribución cambia de forma inmediata para excluir la partición de base de datos desactivada y la entrada para la partición de base de datos se desactiva del grupo de particiones de bases de datos. Si existen tablas,

el mapa de distribución no cambia y el indicador (IN_USE) se establece para indicar que la partición de base de datos está a la espera de ser desactivada. Para redistribuir los datos y desactivar la entrada para la partición de base de datos del grupo de particiones de base de datos, debe utilizarse el mandato REDISTRIBUTE DATABASE PARTITION GROUP.

- *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

Ejemplo

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

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

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

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

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

ALTER DATABASE

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

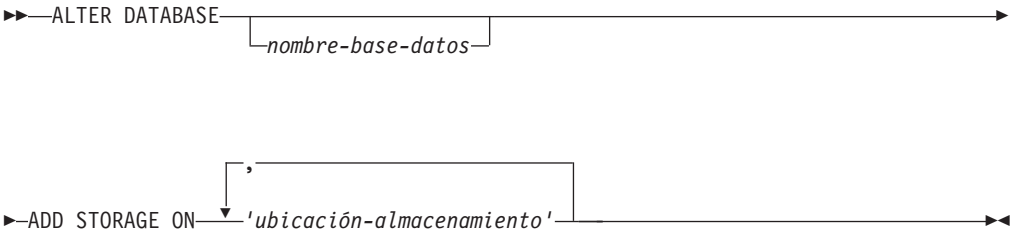
Invocación

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

Autorización

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

Sintaxis



Descripción

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

ADD STORAGE ON
Especifica que una o más ubicaciones de almacenamiento nuevas se añadirán a la colección de ubicaciones de almacenamiento que se utilizan para los espacios de tablas de almacenamiento automático.

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

Normas

- Si no está definido el almacenamiento automático para la base de datos (es decir, se especificó la cláusula AUTOMATIC STORAGE NO en el mandato CREATE DATABASE), no se podrán añadir nuevas vías de acceso de almacenamiento (SQLSTATE 55060).
- La vía de acceso de almacenamiento debe existir y ser accesible (SQLSTATE 57019). De igual modo, en un entorno de bases de datos particionadas, la vía de acceso de almacenamiento debe existir y ser accesible en cada partición de base de datos (SQLSTATE 57019).

Notas

- Cuando se añaden nuevas vías de acceso de almacenamiento:
  - Los espacios de tablas normales y grandes existentes que utilicen el almacenamiento automático no utilizarán estas nuevas vías de acceso inicialmente. El gestor de bases de datos puede elegir crear nuevos contenedores de espacio de tablas en estas vías de acceso únicamente si se produce una condición de falta de espacio.
  - Los espacios de tablas temporales existentes que utilicen el almacenamiento automático pueden utilizar estas nuevas vías de acceso de inmediato. Si un espacio de tablas temporal no está en uso en el momento de una operación ADD STORAGE ON, es posible que los nuevos contenedores se creen en dichas vías de acceso de almacenamiento inmediatamente. No obstante, si el espacio de tablas temporal está en uso, los nuevos contenedores no se crearán hasta que no queden más usuarios del espacio de tablas.
- Aunque ADD STORAGE es una operación que se anota cronológicamente, si se rehace o no durante una operación de recuperación en avance depende de cómo se haya restaurado la base de datos. Si la operación de restauración no redefine las vías de acceso de almacenamiento asociadas a la base de datos, el registro de anotación cronológica que contiene la definición de las vías de acceso de almacenamiento se rehace y las vías de acceso de almacenamiento descritas en el registro de anotación cronológica se añaden durante la operación de recuperación en avance. No obstante, si las vías de acceso de almacenamiento *se redefinen* durante la operación de restauración, la operación de recuperación en avance no rehará este registro de anotación cronológica, porque se supone que ya se han configurado las vías de acceso de almacenamiento.
- No utilice la cláusula ADD STORAGE para añadir vías de acceso de almacenamiento a una base de datos mientras se añade una partición nueva a la base de datos. Si las nuevas vías de acceso de almacenamiento se añaden durante o después de la adición de una partición a la base de datos, pero antes de que se haya iniciado esa nueva partición, las nuevas vías de acceso de almacenamiento no quedarán reflejadas en la nueva partición de base de datos, y las particiones de base de datos quedarán fuera de la sincronización.
- Cuando se calcula el espacio libre para una vía de acceso de almacenamiento en una partición de base de datos, el gestor de bases de datos comprueba la existencia de los siguientes directorios o puntos de montaje en la vía de acceso de almacenamiento, y utiliza la primera que se encuentre.

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

  Donde:

  - <vía acceso almacenamiento> es una vía de acceso de almacenamiento asociada con la base de datos
  - <nombre instancia> es la instancia bajo la que reside la base de datos

  - NODE#### corresponde al número de la partición de base de datos (por ejemplo, NODE0000 o NODE0001)
  - <nombre base de datos> es el nombre de la base de datos

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

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

- En general, se deben utilizar las mismas vías de acceso de almacenamiento para cada partición en una base de datos de varias particiones. Una excepción es el caso en que las expresiones de partición de base de datos se utilizan en la vía de acceso de almacenamiento. Con ello se permite que el número de la partición de base de datos quede reflejado en la vía de acceso de almacenamiento, de manera que el nombre de vía de acceso resultante sea diferente en cada partición.

  Se utiliza el argumento " $N" ([blanco]$N) para indicar una expresión de partición de base de datos. Se puede utilizar una expresión de partición de base de datos en cualquier lugar de la vía de acceso de almacenamiento, y se pueden especificar varias expresiones de partición de base de datos. Termine la expresión de partición de base de datos con un carácter de espacio; lo que haya después del espacio se añadirá a la vía de acceso de almacenamiento después de que se haya evaluado la expresión de partición de base de datos. Si no hay un carácter de espacio en la vía de acceso de almacenamiento después de la expresión de partición de base de datos, se supone que el resto de la serie forma parte de la expresión. Sólo se puede utilizar el argumento de uno de los modos siguientes.

Tabla 10. Argumentos para la creación de vías de acceso de almacenamiento. Los operadores se evalúan de izquierda a derecha. En los ejemplos, se presupone que el número de particiones de base de datos es 10.

Sintaxis	Ejemplo	Valor
[blanco]$N	" $N"	10
[blanco]$N+[número]	" $N+100"	110
[blanco]$N%[número]	" $N%5" [a]	0
[blanco]$N+[número]%[número]	" $N+1%5"	1
[blanco]$N%[número]+[número]	" $N%4+2"	4

[a] % representa el operador del módulo.

Ejemplos

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

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

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

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

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

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

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

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

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

ALTER FUNCTION

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

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

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

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

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

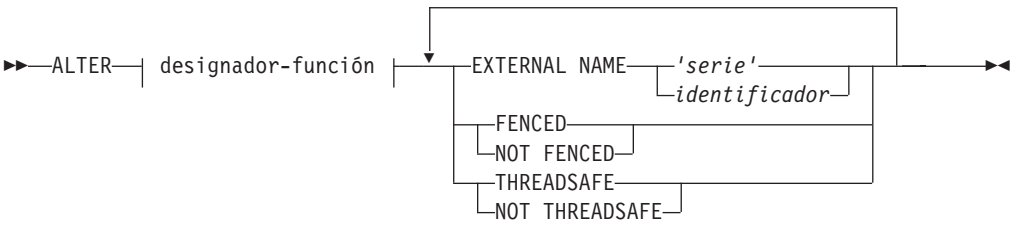
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos
- Autorización SYSADM o DBADM

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 CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM

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

Sintaxis



Descripción

designador-función

Identifica de forma exclusiva la función que va a modificarse. Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 18.

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

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

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

PRECAUCIÓN:
El uso de NOT FENCED para funciones que no se han codificado, revisado ni probado adecuadamente 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 al mismo tiempo la función en el mismo proceso que otra rutina. Sólo una función FENCED puede ser NOT THREADSAFE (SQLSTATE 42613).

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

Notas

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

- Las funciones declaradas como LANGUAGE SQL, las funciones con fuente 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
```

ALTER HISTOGRAM TEMPLATE

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

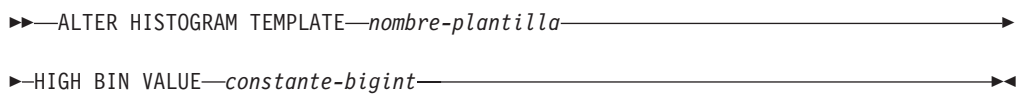
Invocación

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

Autorización

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

Sintaxis



Descripción

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

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

Normas

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

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

Notas

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

Ejemplo

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

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

ALTER METHOD

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

Invocación

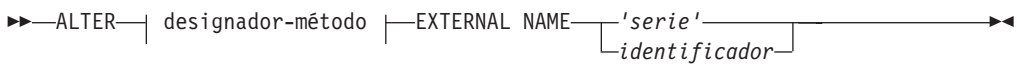
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

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

Sintaxis



Descripción

designador-método
Identifica de forma exclusiva el método que va a modificarse. Para obtener más información, consulte el apartado "Designadores de función, método y procedimiento" en la página 18.

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

Notas

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

Ejemplo

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

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

ALTER 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
- Modificar el almacenamiento en antememoria de datos en un servidor federado

Invocación

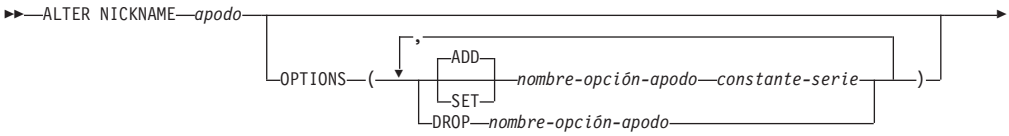
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

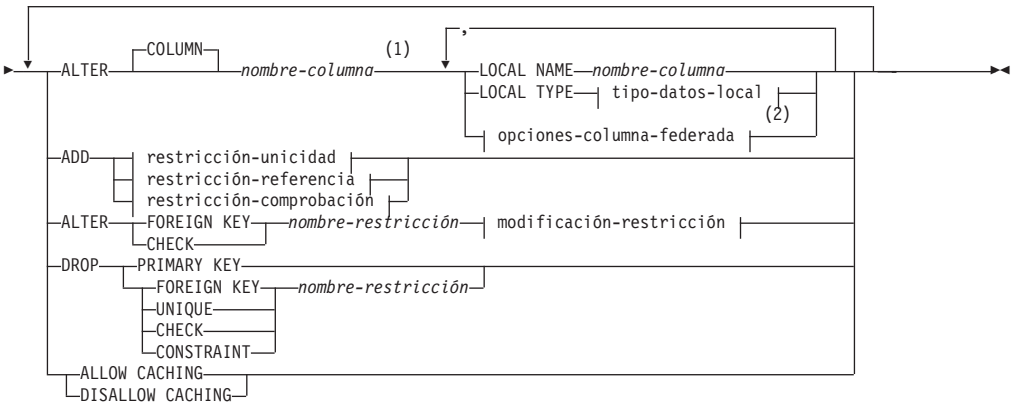
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

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

Sintaxis

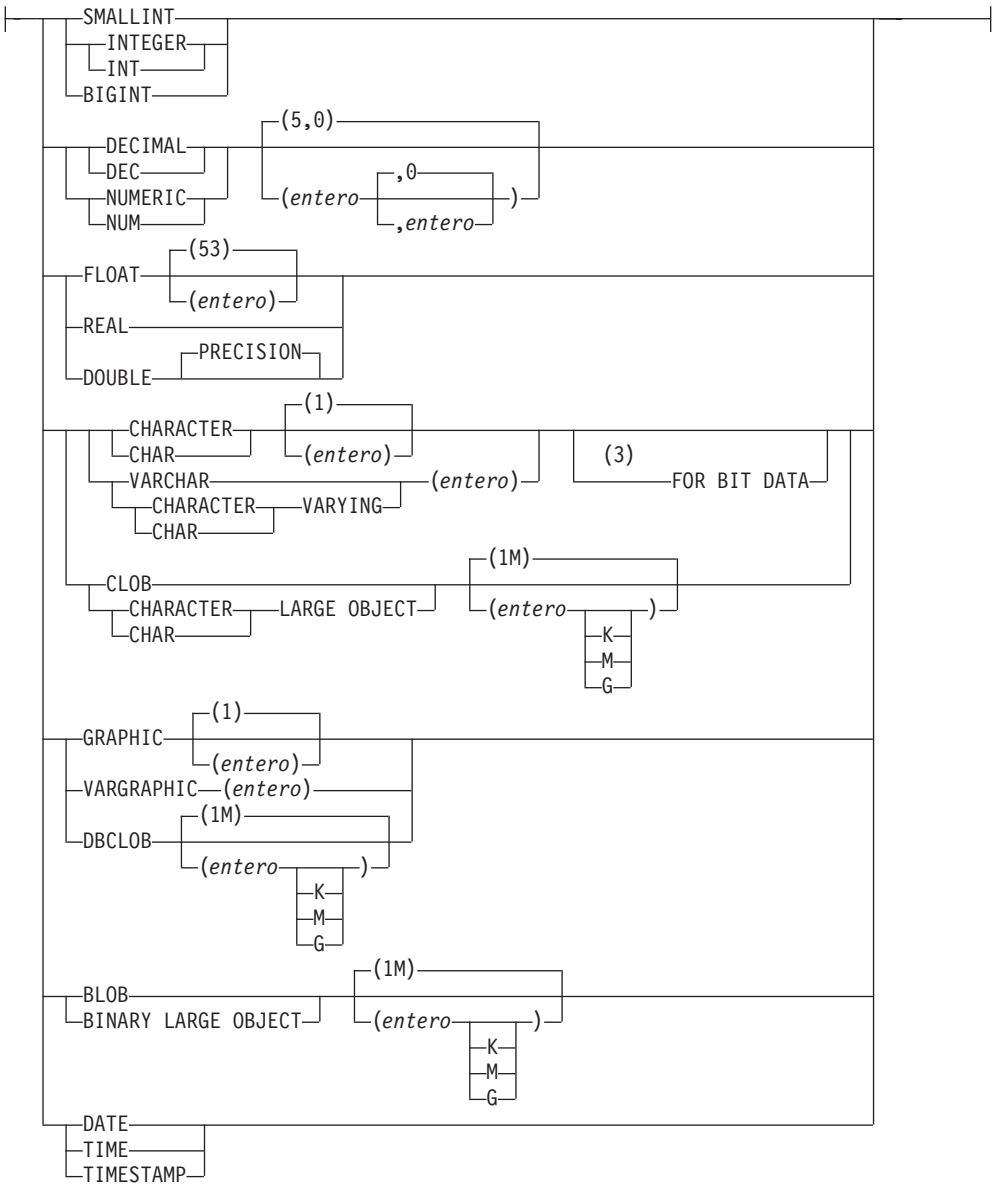


ALTER NICKNAME

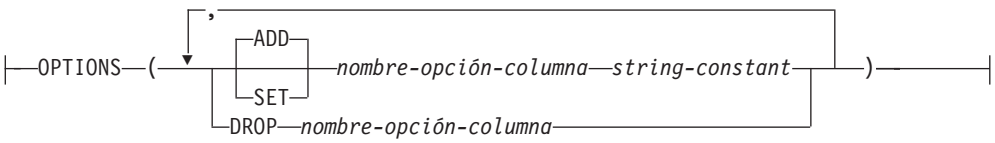


tipo-datos-locales:

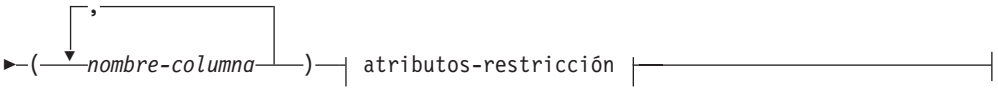
tipo-incorporado:



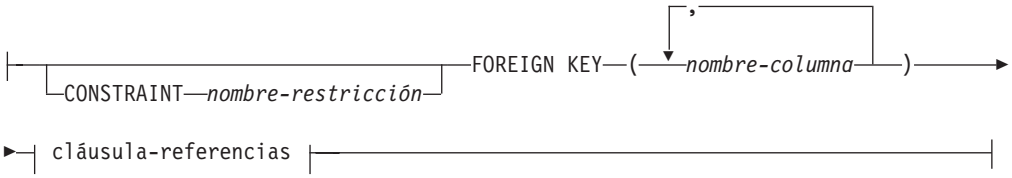
opciones-columna-federada:



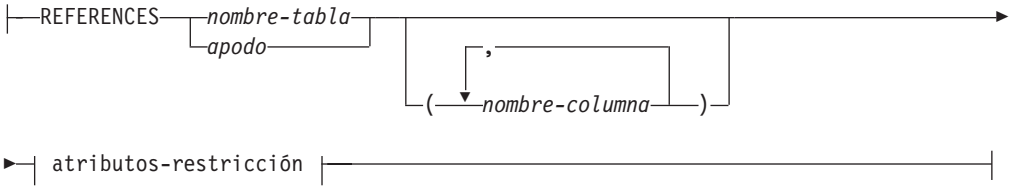
restricción-unicidad:



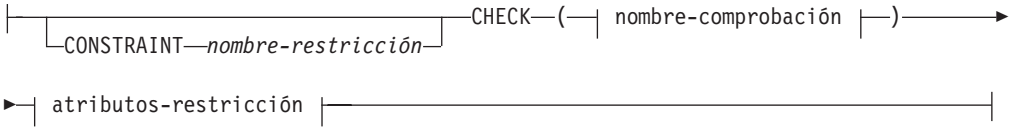
restricción-referencia:



cláusula-referencias:



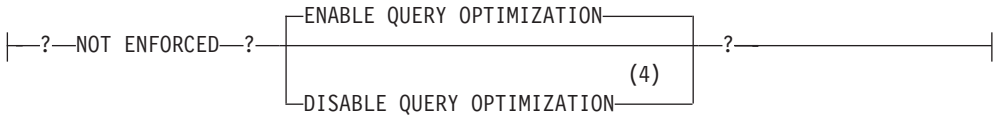
restricción-comprobación:



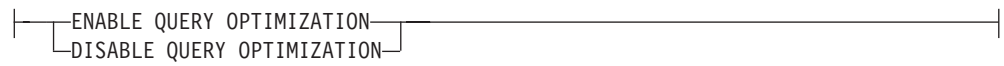
condición-error:

dependencia-funcional:

atributos-restricción:



modificación-restricción:



Notas:

1 No se puede especificar la cláusula ALTER COLUMN y una cláusula de restricción informativa ADD, ALTER o DROP en la misma sentencia ALTER NICKNAME.

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

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

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

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

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

ADD
Añade una opción de columna.

SET
Cambia el valor de una opción de columna.

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

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

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

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

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

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

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

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

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

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

DISABLE QUERY OPTIMIZATION
No se puede utilizar la restricción para la optimización de consulta.

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.

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

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

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

Normas

- Si se utiliza un apodo en una vista, método SQL o función SQL, o se definen sobre la misma restricciones informativas, la sentencia ALTER NICKNAME no se puede utilizar para cambiar los nombres locales o los tipos de datos para las columnas del apodo (SQLSTATE 42893). Sin embargo, la sentencia se puede utilizar para añadir, establecer o descartar opciones de columna, opciones de apodo o restricciones informativas.
- Si una definición de tabla de 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). Además, la sentencia no se puede utilizar para inhabilitar el almacenamiento en antememoria (SQLSTATE 42917). Sin embargo, la sentencia se puede utilizar para añadir, modificar o descartar restricciones informativas.

- No se puede especificar una opción de columna más de una vez en la misma sentencia ALTER NICKNAME (SQLSTATE 42853). Cuando se habilita, restaura o 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.
- ***Almacenamiento en antememoria y objetos protegidos:*** Para apodos cuyo objeto de la fuente de datos está protegido, especifique DISALLOW CACHING. Esto garantiza que cada vez que se utiliza el apodo, se devuelven los datos del ID de autorización apropiados desde la fuente de datos en el momento de la ejecución de la consulta. Esto se realiza limitando el uso del apodo en la definición de una tabla de consulta materializada en el servidor federado, que podría utilizarse para guardar en la antememoria los datos de apodo.

Ejemplos

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

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

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

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

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

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

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

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

ALTER PROCEDURE (externo)

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

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

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

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

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

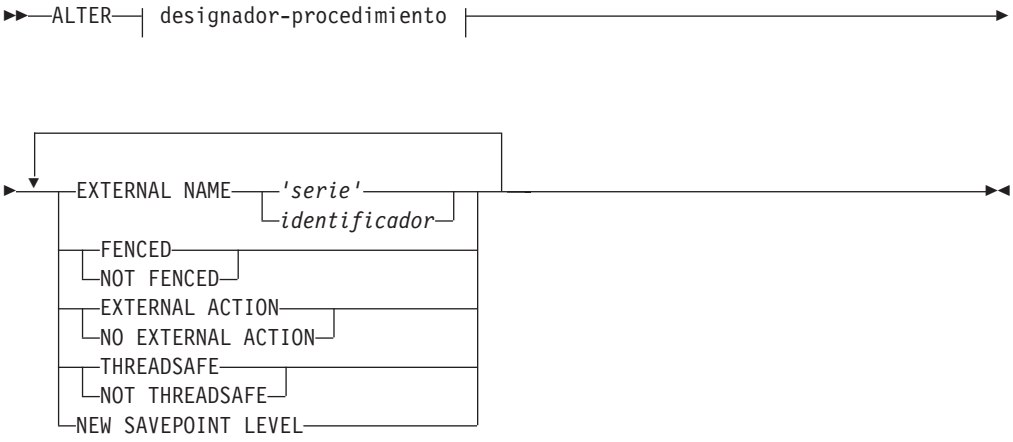
- Autorización CREATE_EXTERNAL_ROUTINE para la base de datos
- Autorización SYSADM o DBADM

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 CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM

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

Sintaxis



Descripción

designador-procedimiento
Identifica el procedimiento a modificar. El *designador-procedimiento* debe identificar un procedimiento que exista en el servidor actual. Se conservan el propietario del procedimiento y todos los privilegios en el procedimiento. Para obtener más información, consulte el apartado "Designadores de función, método y procedimiento" en la página 18.

EXTERNAL NAME *'serie'* **o** *identificador*
Identifica el nombre del código escrito por el usuario que implementa el procedimiento.

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

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

PRECAUCIÓN:
El uso de NOT FENCED en procedimientos que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de 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 utiliza procedimientos almacenados NOT FENCED.

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

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

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

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

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

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

- Si el procedimiento se ha definido como THREADSAFE, el gestor de bases de datos puede invocar el procedimiento en el mismo proceso que otras rutinas. En general, para ser THREADSAFE, un procedimiento no debe utilizar áreas de datos globales o estáticas. En la mayoría de los manuales de

consulta de programación se incluye una explicación acerca de cómo se escriben las rutinas THREADSAFE. Los procedimientos FENCED y NOT FENCED pueden ser THREADSAFE.

- Si el procedimiento se ha definido como NOT THREADSAFE, el gestor de bases de datos nunca invocará el procedimiento en el mismo proceso que otra rutina. Sólo un procedimiento FENCED puede ser NOT THREADSAFE (SQLSTATE 42613).

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

NEW SAVEPOINT LEVEL

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

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

Normas

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

Ejemplo

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

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

ALTER PROCEDURE (con fuente)

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

Invocación

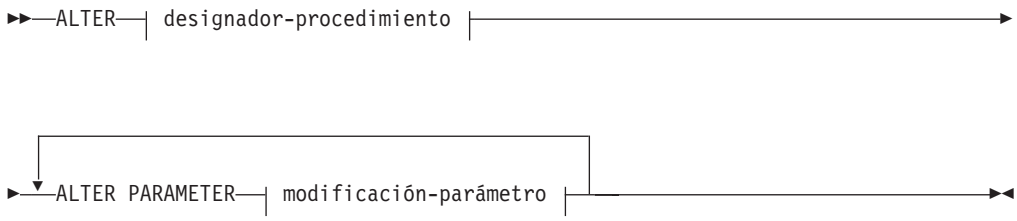
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

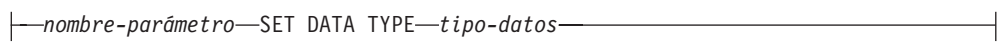
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

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

Sintaxis



modificación-parámetro:



Descripción

designador-procedimiento
Identifica de forma exclusiva el procedimiento que va a modificarse. El procedimiento identificado debe ser un procedimiento con fuente (SQLSTATE 42849). Para obtener más información, consulte el apartado "Designadores de función, método y procedimiento" en la página 18.

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

tipo-datos
Especifica el nuevo tipo de datos local del parámetro. Se pueden especificar abreviaturas y especificaciones de tipo de datos SQL que sean válidas para la definición de *tipo-datos* de una sentencia CREATE TABLE. Los tipos BLOB,

CLOB, DBCLOB, DECFLOAT, XML, REFERENCE y definidos por el usuario no están soportados (SQLSTATE 42815).

Ejemplo

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

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

ALTER PROCEDURE (SQL)

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

Invocación

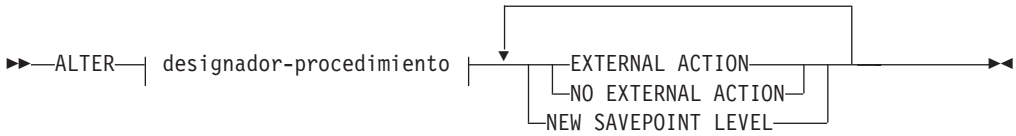
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

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

Sintaxis



Descripción

designador-procedimiento
Identifica el procedimiento a modificar. El *designador-procedimiento* debe identificar un procedimiento que exista en el servidor actual. Se conservan el propietario del procedimiento y todos los privilegios en el procedimiento. Para obtener más información, consulte el apartado "Designadores de función, método y procedimiento" en la página 18.

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

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

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

Normas

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

Ejemplo

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

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

ALTER SECURITY LABEL COMPONENT

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

Invocación

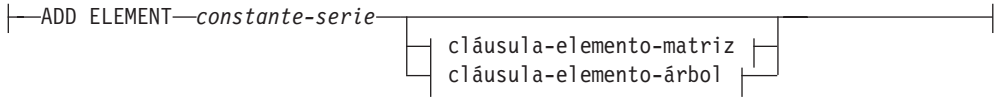
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

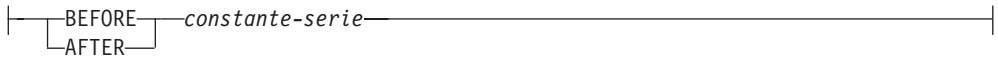
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis

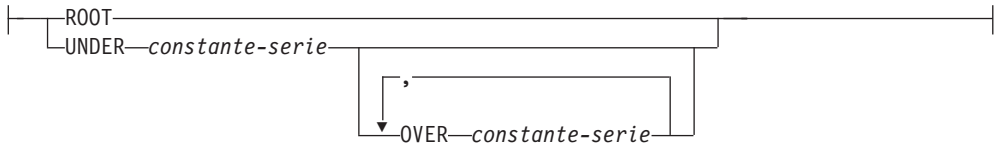
cláusula-añadir-elemento:



cláusula-elemento-matriz:



cláusula-elemento-árbol:



Descripción

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

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

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

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

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

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

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

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

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

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

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

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

Normas

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

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

Notas

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

Ejemplos

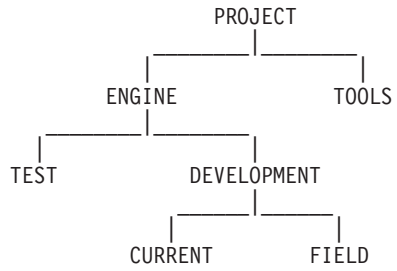
Ejemplo 1: añada el elemento 'High classified' al componente de matriz de etiqueta de seguridad LEVEL entre los elementos 'Secret' y 'Classified'.

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

Ejemplo 2: añada el elemento 'Funding' al componente de conjunto de etiqueta de seguridad COMPARTMENTS.

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

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




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

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

ALTER SECURITY POLICY

La sentencia ALTER SECURITY POLICY modifica una política de seguridad.

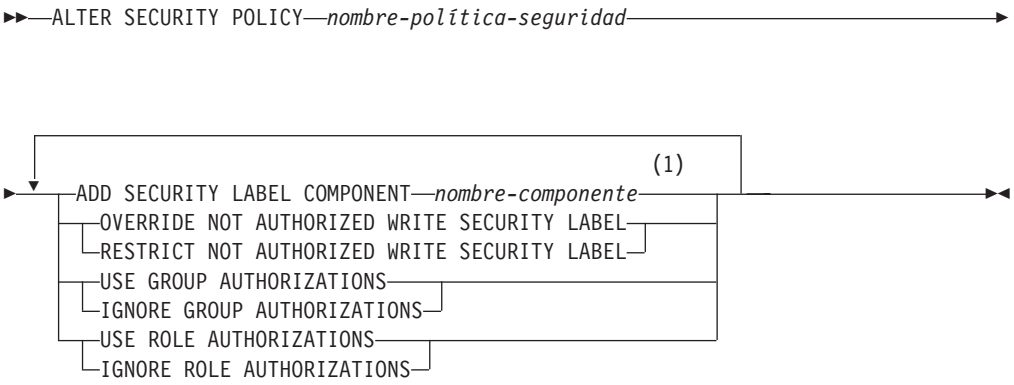
Invocación

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

Autorización

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

Sintaxis



Notas:

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

Descripción

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

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

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL o RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Especifica la acción que se lleva a cabo cuando un usuario no está autorizado para grabar la etiqueta de seguridad explícitamente especificada que se proporciona en la sentencia INSERT o UPDATE emitida sobre una tabla que está protegida con esta política de seguridad. Una etiqueta de seguridad del usuario y las credenciales de exención determinan la autorización del usuario para grabar una etiqueta de seguridad proporcionada explícitamente.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
Indica que se utilizará el valor de la etiqueta de seguridad del usuario para el acceso de grabación durante una operación de inserción o actualización, en lugar de la etiqueta de seguridad explícitamente especificada.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Indica que la operación de inserción o actualización no será satisfactoria si el usuario no está autorizado a grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia INSERT o UPDATE (SQLSTATE 42519).

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

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

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

USE ROLE AUTHORIZATION o IGNORE ROLE AUTHORIZATION
Especifica si se van a tener en cuenta o no las etiquetas de seguridad y exenciones otorgadas a funciones, directa o indirectamente.

USE ROLE AUTHORIZATION
Indica que se tendrá en cuenta cualquier etiqueta de seguridad o exención otorgada a funciones, directa o indirectamente.

IGNORE ROLE AUTHORIZATION
Indica que no se tienen en cuenta las etiquetas de seguridad o exenciones otorgadas a funciones.

Normas

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

Notas

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

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

  Los mecanismos para combinar etiquetas de seguridad varían en función del tipo de componente. Los componentes de la etiqueta de seguridad resultante son los siguientes:
  - Los componentes de conjunto contienen la unión de todos los valores exclusivos encontrados en las etiquetas de seguridad que pueden elegirse
  - Los componentes de matriz contienen el elemento de orden más elevado encontrado en las etiquetas de seguridad que pueden elegirse
  - Los componentes de árbol contienen la unión de todos los valores exclusivos encontrados en las etiquetas de seguridad que pueden elegirse
- Si hay más de una exención que se pueda tener en cuenta durante un intento de acceso, todas las exenciones encontradas se aplican en el intento de acceso.

Ejemplos

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

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

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

```
ALTER SECURITY POLICY DATA_ACCESS
  USE ROLE AUTHORIZATIONS
```

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

Matriz = {TS, S, C, U}
Conjunto = {A, B, X, Y}

Las siguientes etiquetas de seguridad están definidas para SECUR_POL:

Etiqueta de seguridad L1 = C:A
Etiqueta de seguridad L2 = S:B
Etiqueta de seguridad L3 = TS:X
Etiqueta de seguridad L4 = U:Y

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

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

	Funciones habilitadas	**Funciones inhabilitadas**
Grupos habilitados	L1, L2, L3, L4	L1, L3
Grupos inhabilitados	L1, L2	L1

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

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

	Funciones habilitadas	**Funciones inhabilitadas**
Grupos habilitados	TS:(A, B, X, Y)	TS:(A, X)
Grupos inhabilitados	S:(A, B)	C:A

ALTER SEQUENCE

La sentencia ALTER SEQUENCE puede utilizarse para cambiar una secuencia 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

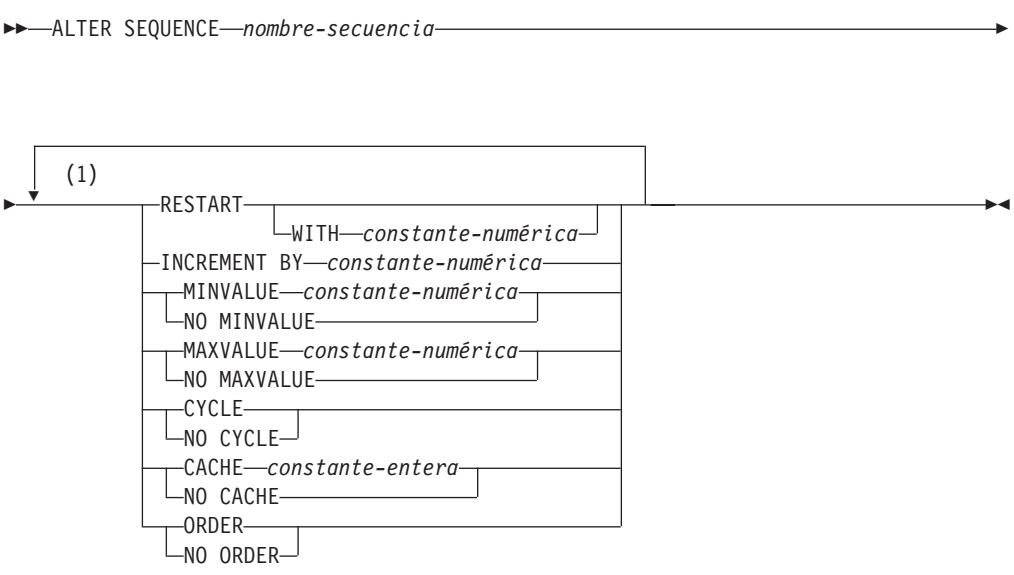
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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 en el esquema especificado implícita o explícitamente no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704). El *nombre-secuencia* no debe ser una secuencia generada por el sistema para una columna de identidad (SQLSTATE 428FB).

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

WITH *constante-numérica*
Reinicia la secuencia con el valor especificado. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos 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

- 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.
- ***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

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
```

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 terminan por *-servidor* sólo hacen referencia a las fuentes de datos de un sistema federado. No hacen referencia al servidor federado de ese sistema ni a los servidores de aplicaciones DRDA.

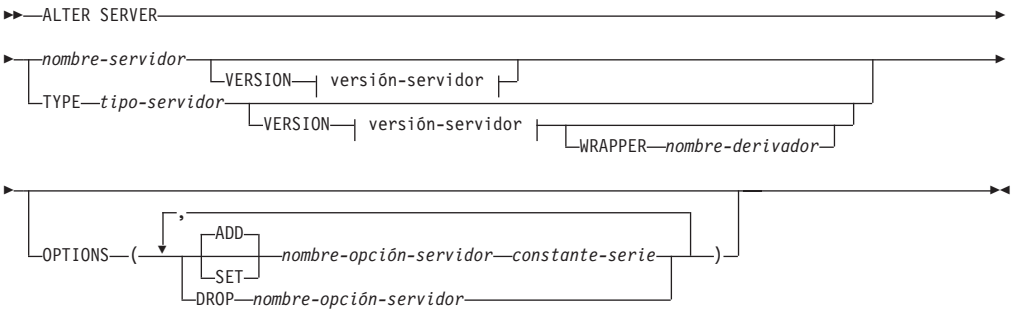
Invocación

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

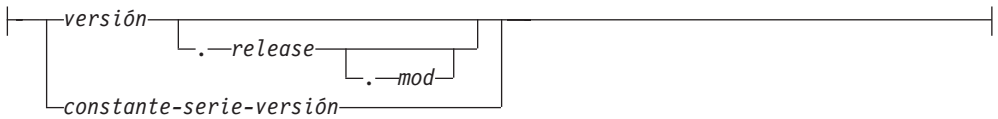
Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización 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-derivador*
Especifica el nombre del derivador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*. El derivador debe aparecer en la lista del catálogo.

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

ADD
Habilita una opción de servidor.

SET
Cambia el valor de una opción de servidor.

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

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

DROP *nombre-opción-servidor*
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).
- Al modificar la versión del servidor, DB2 no comprueba que la versión del servidor especificado coincida con la versión del servidor remoto. Si se especifica una versión de servidor incorrecta se pueden provocar errores de SQL cuando se accede a apodos que pertenecen a la definición del servidor DB2. Esto puede producirse con mucha probabilidad cuando se especifica una versión de servidor posterior a la versión del servidor remoto. En ese caso, al acceder a apodos que pertenecen a la definición del servidor, puede que DB2 envíe SQL que el servidor remoto no reconoce.

Ejemplos

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

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

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

```
ALTER SERVER DCTM_SVR_ASIA
  VERSION 4
```

ALTER SERVICE CLASS

La sentencia ALTER SERVICE CLASS modifica la definición de un servicio.

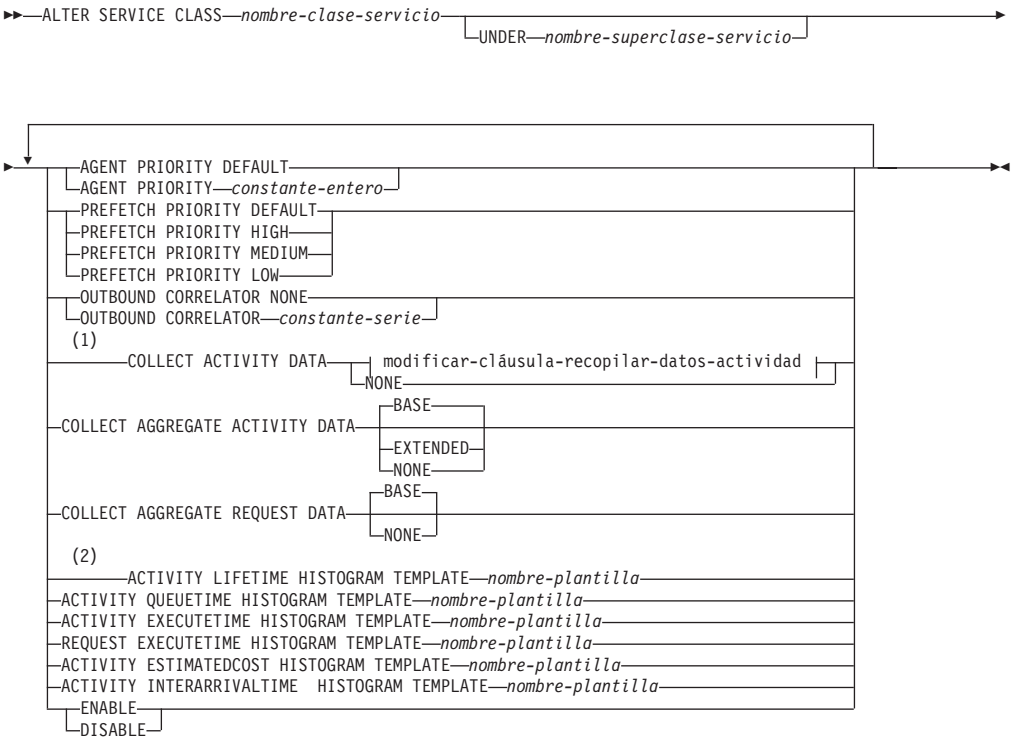
Invocación

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

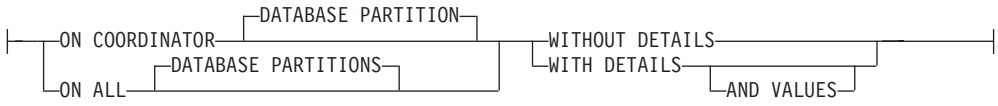
Autorización

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

Sintaxis



modificar-cláusula-recopilar-datos-actividad:



Notas:

1 Las cláusulas COLLECT sólo son válidas para una subclase de servicio.

2 Las cláusulas HISTOGRAM TEMPLATE sólo son válidas para una subclase de servicio.

Descripción

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

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

AGENT PRIORITY DEFAULT o AGENT PRIORITY *constante-entero*
Especifica la prioridad del sistema operativo (delta) relativa de los agentes que se ejecutan en la clase de servicio o la prioridad normal de las hebras que se ejecutan en DB2. El valor por omisión es DEFAULT. Cuando se establece en DEFAULT, no se adopta ninguna acción especial y los agentes de la clase de servicio se planifican con arreglo a la prioridad normal con la que el sistema operativo planifica todas las hebras de DB2. Cuando se establece este parámetro en un valor que no sea DEFAULT, los agentes se establecen en una prioridad que sea igual a la prioridad normal más AGENT PRIORITY cuando comienza la siguiente actividad. Por ejemplo, si la prioridad normal es 20 y AGENT PRIORITY se establece en -10, la prioridad de los agentes de la clase de servicio se establece en 20 – 10 = 10.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

OUTBOUND CORRELATOR NONE o OUTBOUND CORRELATOR *constante-serie*
Especifica si asociar o no hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa.

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

OUTBOUND CORRELATOR NONE
Para una superclase de servicio, especifica que no hay ninguna asociación de clase de servicio de gestor de carga de trabajo externa con esta clase de servicio y, para una subclase de servicio, especifica que la asociación de clase de servicio de gestor de carga de trabajo externa es la misma que la de su padre.

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

COLLECT ACTIVITY DATA
Especifica que la información sobre cada una de las actividades que se ejecuta en esta clase de servicio se va a enviar al supervisor de sucesos de actividades,

si hay alguno activo, cuando la actividad finaliza. La cláusula COLLECT ACTIVITY DATA sólo es válida para una subclase de servicio.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Sin embargo, los valores y detalles de actividad sólo se recopilarán en la partición de la base de datos del coordinador.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades que se ejecuta en esta clase de servicio se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza su ejecución. El entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que los datos de entorno de compilación y sentencia se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que se ejecutan en esta clase de servicio.

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

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

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo de cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED
Especifica que todos los datos de actividades agregados deben capturarse

para esta clase de servicio y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

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

NONE
Especifica que no debe capturarse ningún dato de actividad agregados para esta clase de servicio.

COLLECT AGGREGATE REQUEST DATA
Especifica que los datos de peticiones agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo especificado por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE REQUEST DATA NONE. La cláusula COLLECT AGGREGATE REQUEST DATA sólo es válida para una subclase de servicio.

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

NONE
Especifica que no debe capturarse ningún dato de petición agregado para esta clase de servicio.

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

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

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, en que las actividades de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en la partición de base de datos del coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. Esta información sólo se recopila cuando se especifica la cláusula COLLECT

AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Esta cláusula sólo es válida para una subclase de servicio.

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

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

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

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

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

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

Normas

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

  - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
  - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.
- Después de confirmar una sentencia ALTER SERVICE CLASS, los cambios en AGENT PRIORITY, PREFETCH PRIORITY, OUTBOUND CORRELATOR y COLLECT surtirán efecto para la siguiente actividad nueva de la clase de servicio. Las actividades de la clase de servicio siguen completando su trabajo utilizando los valores antiguos.

Ejemplos

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

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

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

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

ALTER TABLE

La sentencia ALTER TABLE modifica la definición de una tabla.

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

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 sobre 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 sobre la tabla
- Autorización SYSADM o DBADM

Para descartar la clave primaria o una restricción de unicidad en la tabla T, el ID de autorización de la sentencia debe mantener 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 sobre 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 sobre 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 (excluyendo los privilegios de grupo):

- Privilegio SELECT y privilegio ALTER (incluyendo los privilegios de grupo) para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Privilegio SELECT para la tabla o vista y privilegio ALTERIN (incluyendo los privilegios de grupo) para el esquema de la tabla o vista
- Autorización 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 sobre la tabla o vista
- Privilegio ALTERIN para el esquema de la tabla o vista
- Autorización SYSADM o DBADM

Para añadir una columna de tipo DB2SECURITYLABEL a una tabla, los privilegios que tiene el ID de autorización de la sentencia deben incluir al menos una etiqueta de seguridad de la política de seguridad asociada a la tabla.

Para eliminar la política de seguridad de una tabla, los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Para modificar una tabla a fin de enlazar una partición de datos, los privilegios que tiene el ID de autorización de la sentencia también deben incluir al menos uno de los elementos siguientes en la tabla fuente:

- Privilegio SELECT para la tabla y privilegio DROPIN para el esquema de la tabla
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

y como mínimo uno de los elementos siguientes en la tabla de destino:

- Privilegios ALTER e INSERT para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

Para modificar una tabla a fin de desenlazar una partición de datos, los privilegios que tiene el ID de autorización de la sentencia también deben incluir al menos uno de los elementos siguientes en la tabla de destino de la partición desenlazada:

- Autorización CREATETAB para la base de datos y privilegio USE para los espacios de tablas que utiliza la tabla, así como uno de los elementos siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla nueva no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla nueva hace referencia a un esquema existente
- Autorización SYSADM o DBADM

y como mínimo uno de los elementos siguientes en la tabla fuente:

- Privilegios SELECT, ALTER y DELETE para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

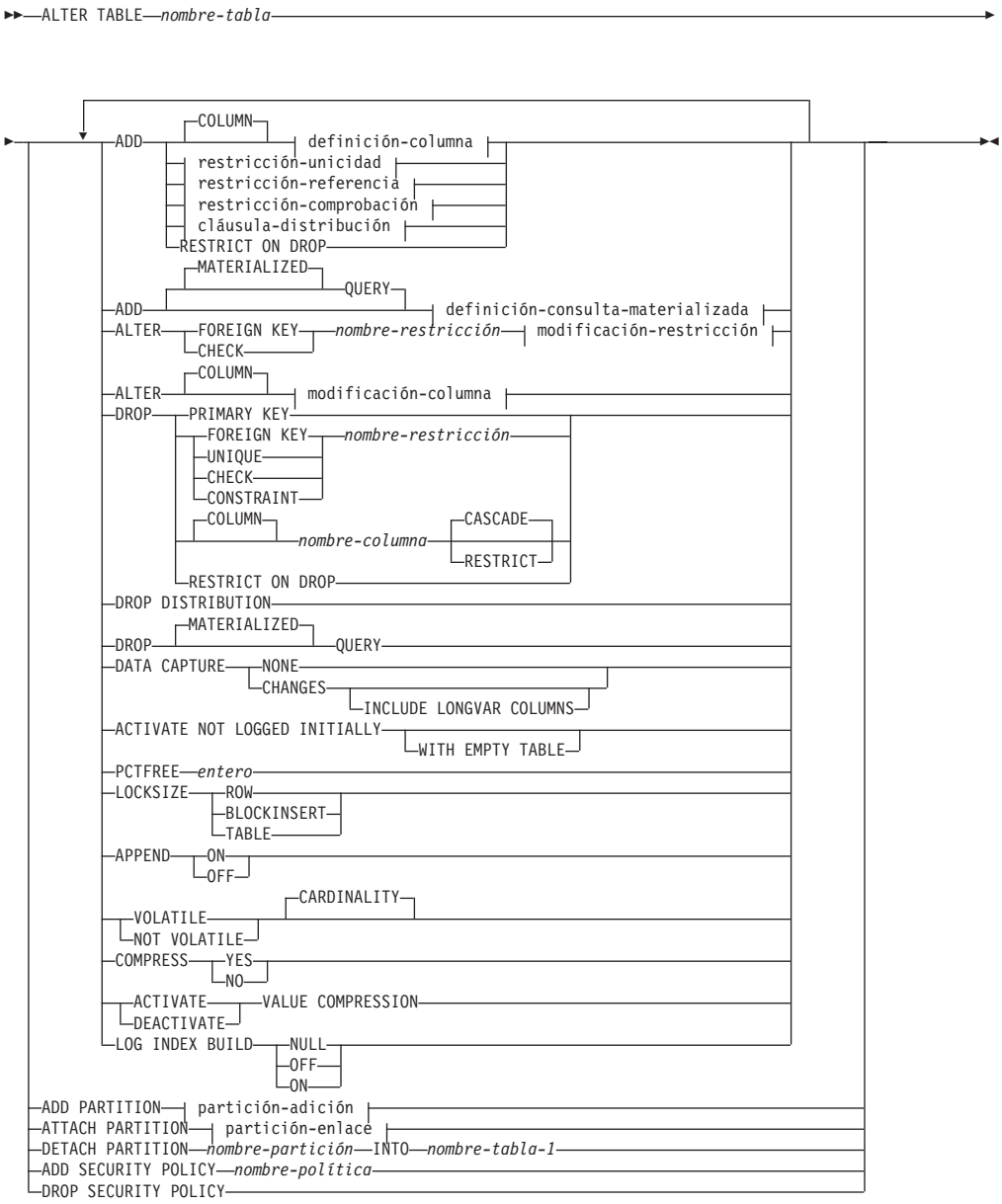
Para modificar una tabla para la activación no registrada inicialmente con tabla vacía, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

- Privilegios ALTER y DELETE para la tabla
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

Para modificar una tabla protegida mediante una política de seguridad para la activación no registrada inicialmente con tabla vacía, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

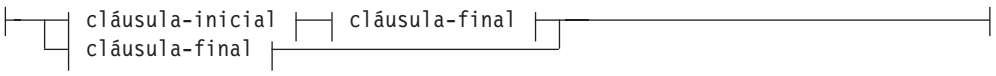
- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

Sintaxis

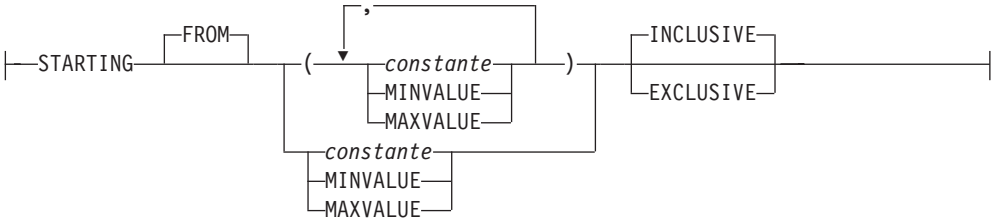


partición-adición:

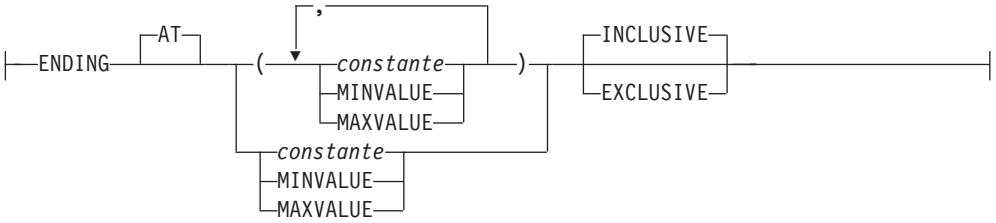
espec-límite:



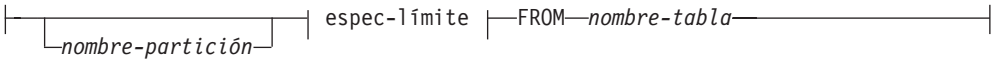
cláusula-inicial:



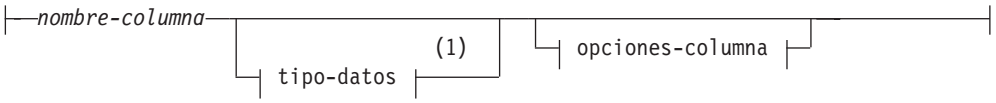
cláusula-final:



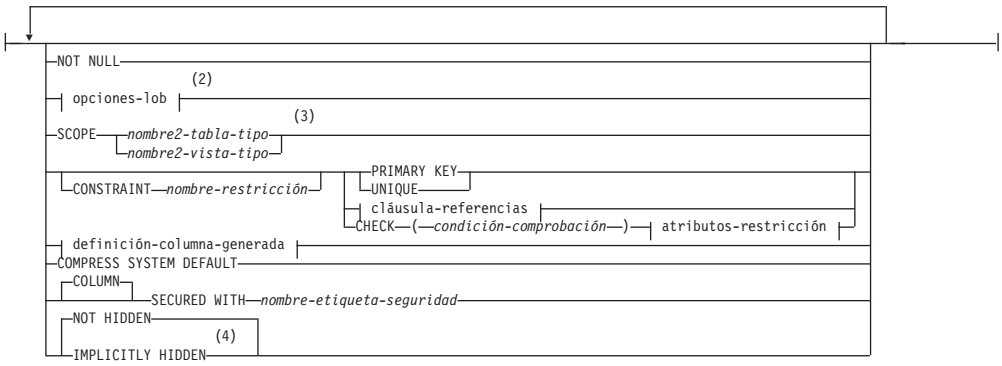
partición-enlace:



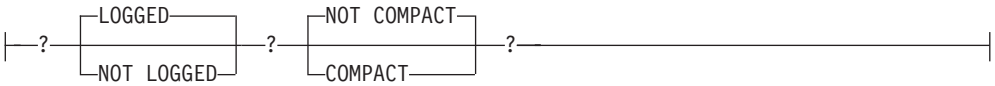
definición-columna:



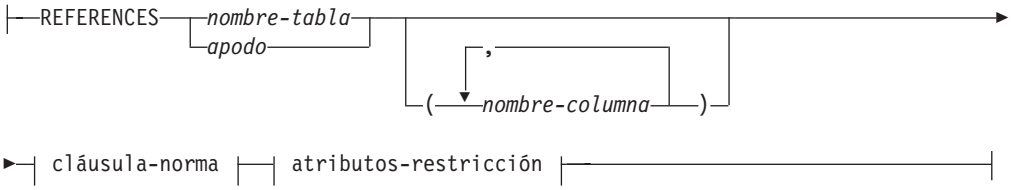
opciones-columna:



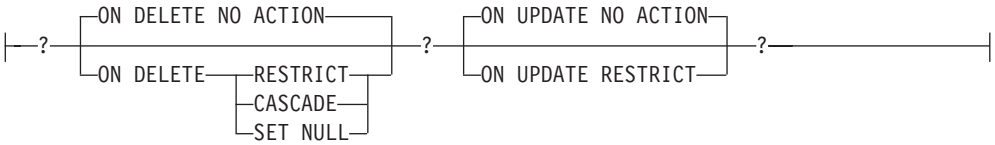
opciones-lob:



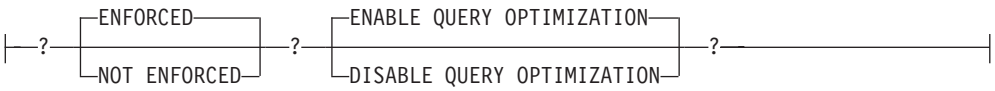
cláusula-referencias:



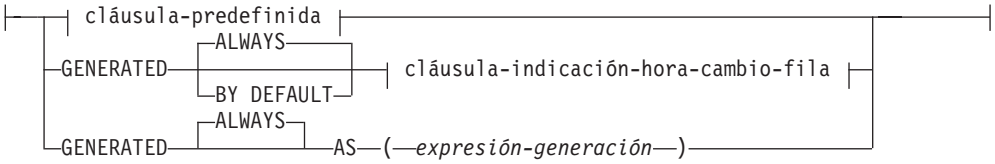
cláusula-norma:



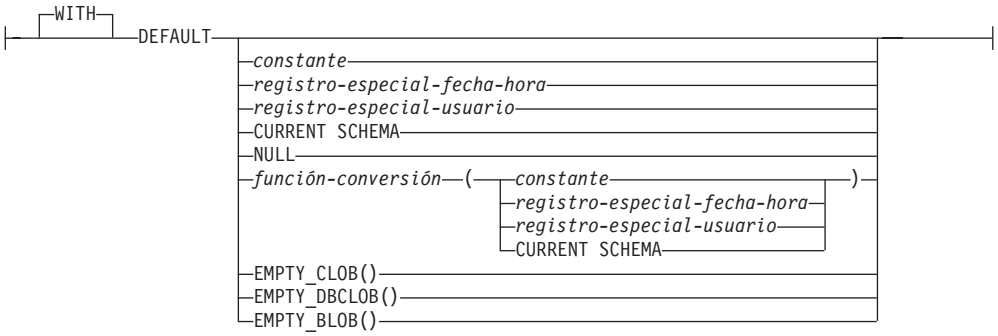
atributos-restricción:



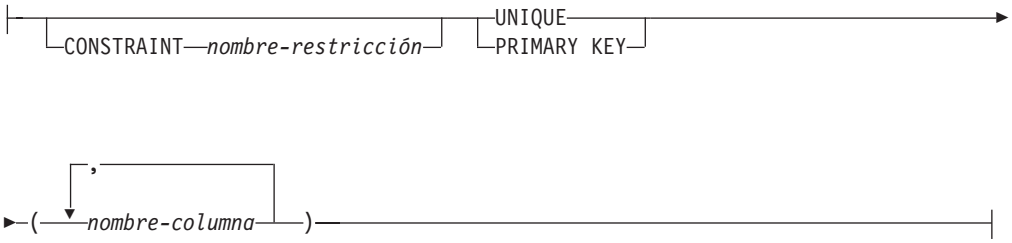
definición-columna-generada:



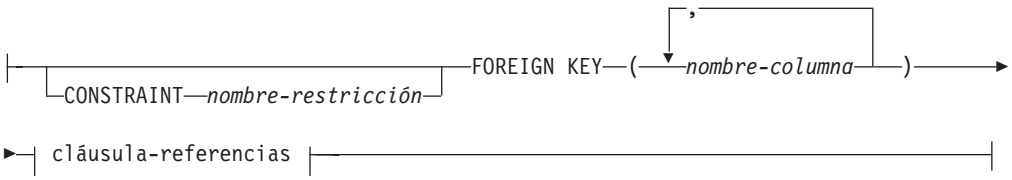
cláusula-predefinida:



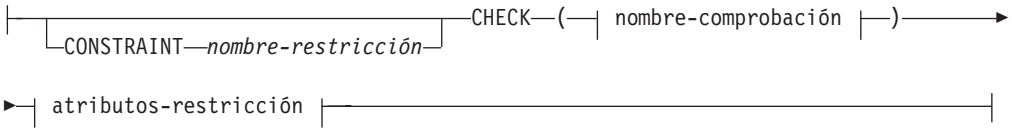
restricción-unicidad:



restricción-referencia:



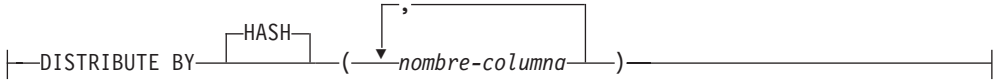
restricción-comprobación:



condición-error:

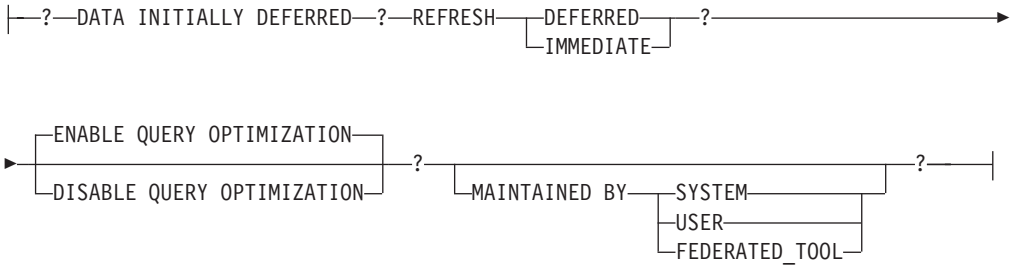
dependencia-funcional:

cláusula-distribución:

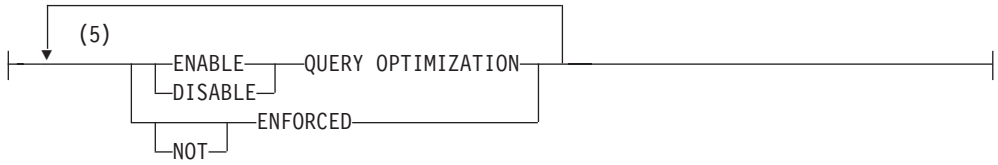


definición-consulta-materializada:

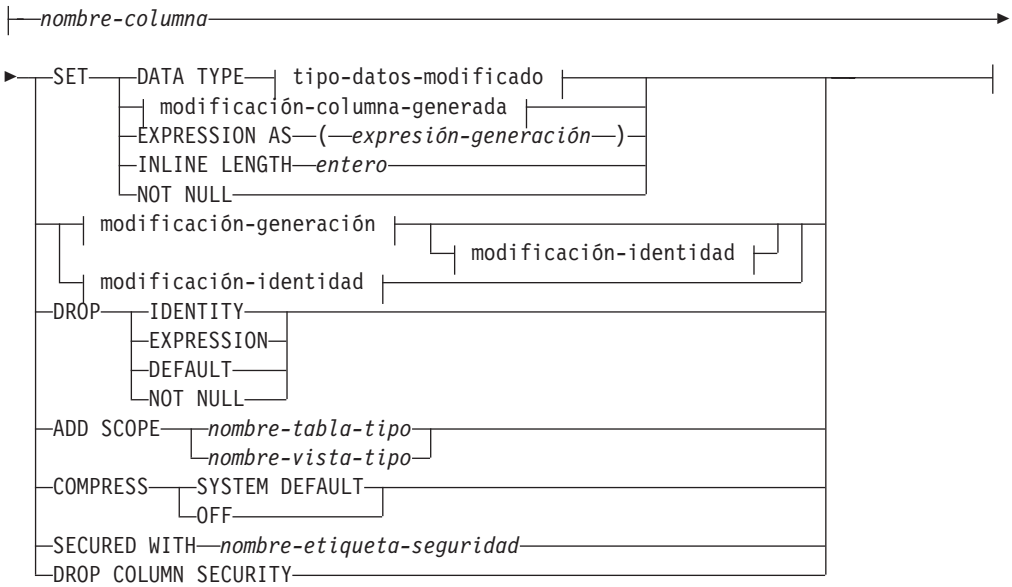
opciones-tabla-renovable:



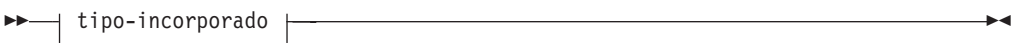
modificación-restricción:



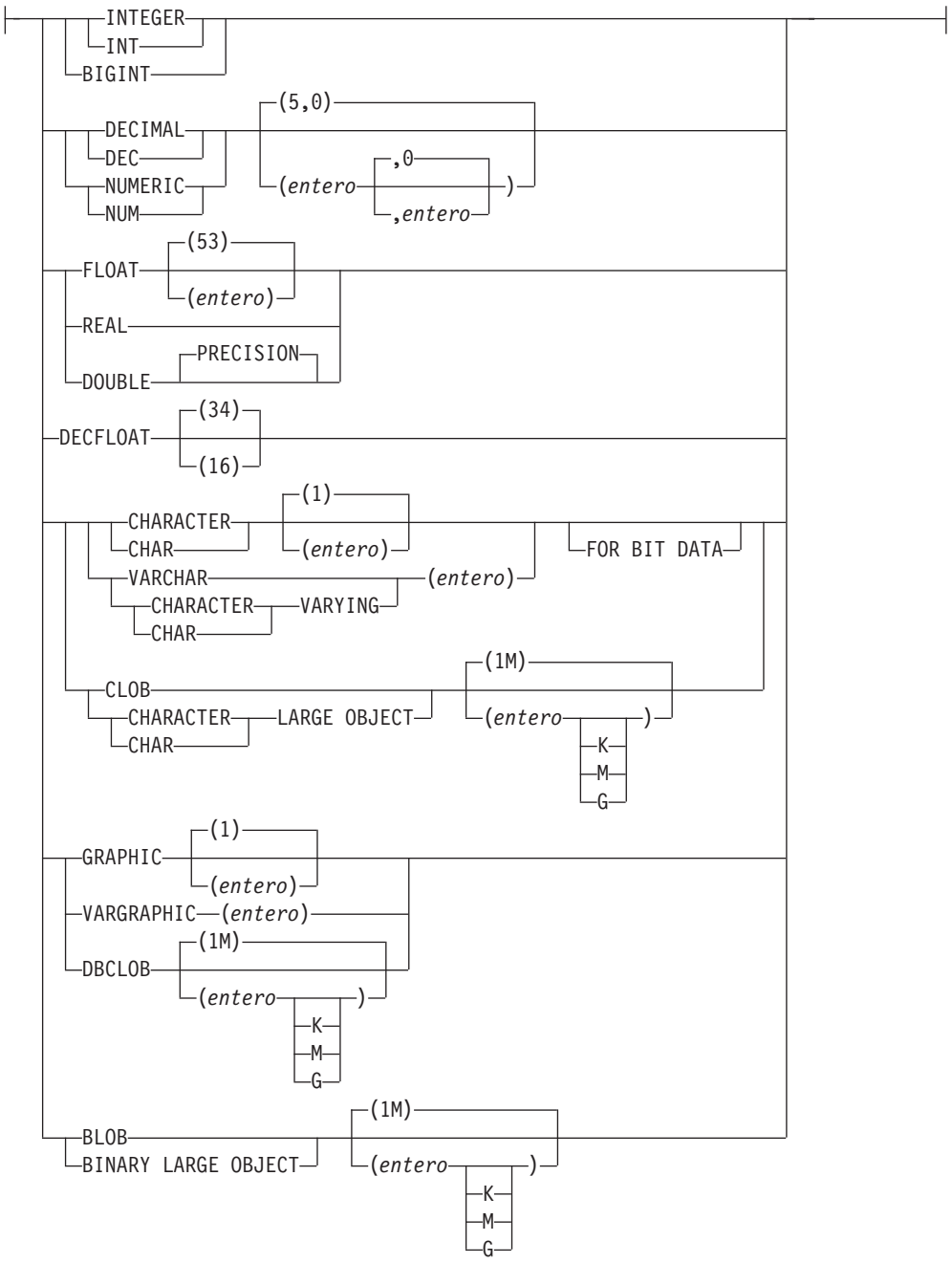
modificación-columna:



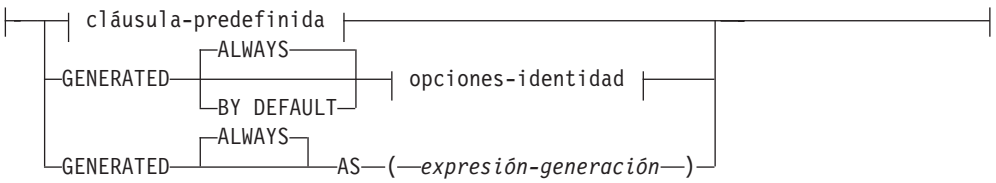
tipo-datos-modificado:



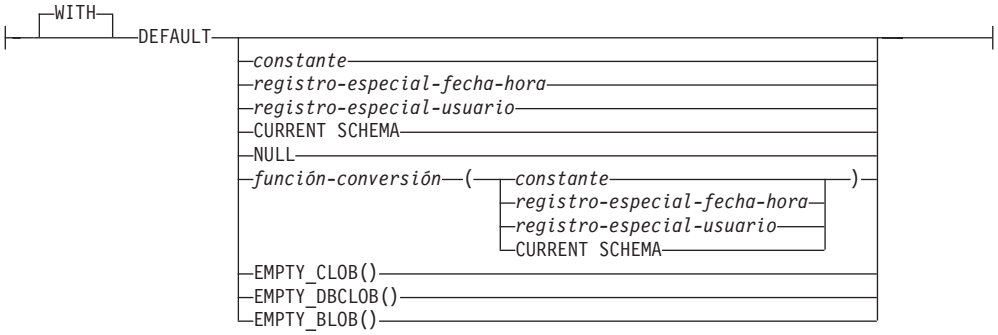
tipo-incorporado:



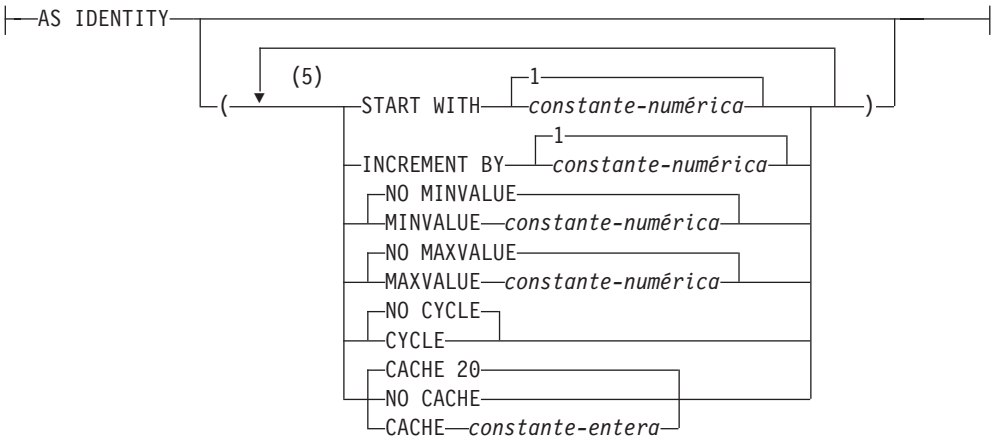
modificación-columna-generada:



cláusula-predefinida:



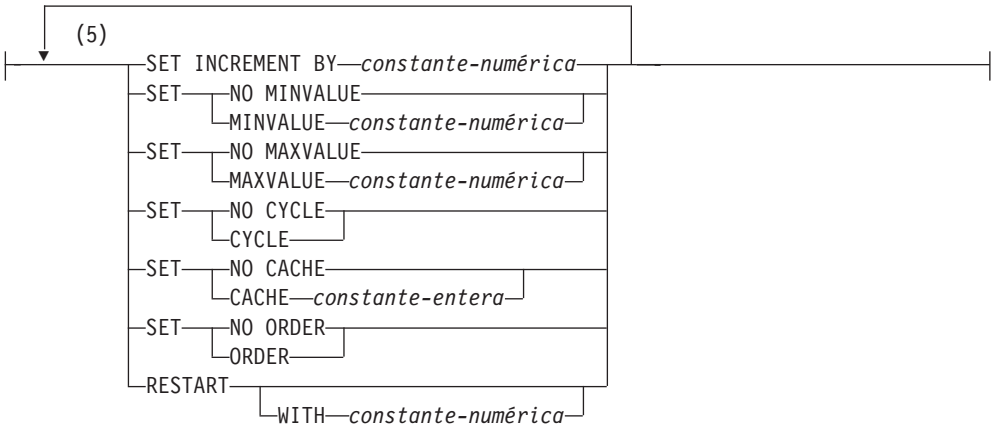
opciones-identidad:



cláusula-indicación-hora-cambio-fila:

modificación-generación:

modificación-identidad:



Notas:

1 Si la primera opción de columna seleccionada es *espec-columna-generada*, se puede omitir *tipo-datos*; la expresión de generación calculará este valor.

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

3 La cláusula SCOPE sólo se aplica al tipo REF.

4 IMPLICITLY HIDDEN sólo se puede especificar si también se especifica ROW CHANGE TIMESTAMP.

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

6 El tipo de dato es opcional para una columna de indicación de fecha y hora de cambio de fila.

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 consulta materializada, las modificaciones se limitan a añadir o descartar la tabla de consulta materializada, a realizar la activación no registrada inicialmente, a añadir o descartar RESTRICT ON DROP, a cambiar a una tabla regular y a modificar DATA CAPTURE, PCTFREE, LOCKSIZE, APPEND, VOLATILE o la compresión de filas de datos o valores. .

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 PARTITION *partición-adición*
Añade una o varias particiones de datos a una tabla particionada. Si la tabla especificada no es una tabla particionada, se devuelve un error (SQLSTATE 428FT). El número de particiones de datos no puede ser superior a 32 767.

nombre-partición
Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

espec-límite
Especifica el rango de valores de la partición de datos nueva. Este rango no puede solapar el de una partición de datos existente (SQLSTATE 56016). Para ver una descripción de la cláusula-inicial y la cláusula-final, consulte "CREATE TABLE".

Si se omite la cláusula-inicial, la partición de datos nueva se supone que se encuentra al final de la tabla. Si se omite la cláusula-final, la partición de datos nueva se supone que se encuentra al principio de la tabla. Si la primera columna de la clave de particionamiento es DESC, estas suposiciones se invierten.

IN *nombre-espacio-tablas*
Especifica el espacio de tablas donde debe almacenarse la partición de datos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas de la tabla particionada (SQLSTATE 42838). Puede ser un espacio de tablas que ya esté en uso para otra partición de datos de la misma tabla o un espacio de tablas que esta tabla no utilice actualmente, pero debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE (SQLSTATE 42727). Si no se especifica esta cláusula, se utiliza el espacio de tablas de la primera partición de datos visible o adjunta de la tabla.

LONG IN *nombre-espacio-tablas*
Especifica el espacio de tablas donde debe almacenarse la partición de datos que contiene datos de columna largos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas y particiones de datos de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE. El tamaño de página y el tamaño de extensión del espacio de tablas especificado pueden ser distintos al tamaño de página y al tamaño de extensión de las otras particiones de datos de la tabla particionada.

Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

ATTACH PARTITION *partición-enlace*
Enlaza otra tabla como una partición de datos nueva. El objeto de datos de la tabla que se enlaza se convierte en una partición nueva de la tabla a la que se enlaza. No existe ningún movimiento de datos. La tabla se coloca en estado pendiente de establecer integridad y la comprobación de la integridad de referencia se difiere hasta la ejecución de una sentencia SET INTEGRITY. La operación ALTER TABLE ATTACH no permite el uso de la cláusula IN o LONG. La colocación de LOB para la partición de datos en cuestión se determina al crear la tabla fuente. Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

nombre-partición
Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

espec-límite
Especifica el rango de valores de la partición de datos nueva. Este rango no puede solapar el de una partición de datos existente (SQLSTATE 56016). Para ver una descripción de la cláusula-inicial y la cláusula-final, consulte "CREATE TABLE".

Si se omite la cláusula-inicial, la partición de datos nueva se supone que se encuentra al final de la tabla. Si se omite la cláusula-final, la partición de datos nueva se supone que se encuentra al principio de la tabla.

FROM *nombre1-tabla*
Especifica la tabla que se utilizará como fuente de datos de la partición nueva. La definición de tabla de *nombre-tabla1* no puede tener varias particiones de datos y debe coincidir con la tabla modificada en los aspectos siguientes (SQLSTATE 428G3):

- El número de columnas debe ser el mismo.
- Los tipos de datos de las columnas de la misma posición ordinal en la tabla deben ser los mismos.
- La característica de capacidad de nulos de las columnas de la misma posición ordinal en la tabla debe ser la misma.
- Si los datos también se distribuyen, deben distribuirse en el mismo grupo de particiones de base de datos mediante el mismo método de distribución.
- Si los datos de cualquiera de las tablas están organizados, la organización debe coincidir.

Una vez enlazados correctamente los datos de *nombre-tabla1*, se realiza una operación equivalente a DROP TABLE *nombre-tabla1* para eliminar esta tabla, que ya no tiene datos, de la base de datos.

DETACH PARTITION *nombre-partición* **INTO** *nombre1-tabla*
Desenlaza la partición de datos *nombre-partición* de la tabla modificada y utiliza la partición de datos para crear una tabla nueva denominada *nombre-tabla1*. La partición de datos se enlaza lógicamente a la tabla nueva sin que exista ningún movimiento de datos. La partición de datos especificada no puede ser la última partición restante de la tabla modificada (SQLSTATE 428G2).

ADD SECURITY POLICY *nombre-política*
Añade una política de seguridad a la tabla. La política de seguridad debe existir en el servidor actual (SQLSTATE 42704). La tabla no puede tener ya una política de seguridad (SQLSTATE 55065) y no debe ser una tabla con tipos (SQLSTATE 428DH), una tabla de consultas materializadas (MQT) o tabla de etapas (SQLSTATE 428FG).

DROP SECURITY POLICY
Elimina la política de seguridad y toda la protección LBAC de la tabla. La tabla especificada por *nombre-tabla* debe estar protegida mediante una política de seguridad (SQLSTATE 428GT). Si la tabla tiene una columna con el tipo de datos DB2SECURITYLABEL, el tipo de datos se cambia a VARCHAR (128) FOR BIT DATA. Si la tabla tiene una o varias columnas protegidas, esas columnas pasarán a estar desprotegidas.

ADD *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 al principio hay *n* columnas, la columna añadida es la columna *n*+1.

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

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).

NOT HIDDEN o IMPLICITLY HIDDEN

Especifica si hay que definir la columna como oculta o no. El atributo oculto determina si la columna se incluye como referencia implícita a la tabla o si se le puede hacer referencia de manera explícita en sentencias de SQL. El valor por omisión es NOT HIDDEN.

NOT HIDDEN

Especifica que la columna se incluye como referencia implícita a la tabla y que se puede hacer referencia a la columna de manera explícita.

IMPLICITLY HIDDEN

Especifica que la columna no es visible en sentencias de SQL a menos que se haga referencia a la columna de manera explícita por el nombre. Por ejemplo, asumiendo que una tabla incluye una columna definida con la cláusula IMPLICITLY HIDDEN, el resultado de SELECT * no incluye la columna implícitamente oculta. Sin embargo, el resultado de SELECT que haga referencia de manera explícita al nombre de una columna implícitamente oculta incluirá dicha columna en la tala de resultados.

IMPLICITLY HIDDEN sólo debe especificarse para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42867). La expresión ROW CHANGE TIMESTAMP FOR *designador-tabla* se resolverá en una columna IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP. Por tanto, se puede añadir una columna ROW CHANGE TIMESTAMP a una tabla como IMPLICITLY HIDDEN y no será necesario modificar las aplicaciones existentes que hagan SELECT * desde esta tabla para manejar la columna. Al utilizar la expresión, las aplicaciones nuevas siempre pueden acceder a la columna sin conocer el nombre de la columna.

opciones-lob

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

SCOPE

Especifique un ámbito para una columna de tipo de referencia.

nombre2-tabla-tipo
El nombre de una tabla con tipo. El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo2* (SQLSTATE 428DM). No se realiza ninguna comprobación del valor por omisión de *nombre-columna* para asegurarse de si realmente el valor hace referencia a una fila existente de *nombre-tabla-tipo2*.

nombre2-vista-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-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 bytes dentro de los identificadores de las restricciones existentes definidas en la tabla. (El identificador se compone de la palabra "SQL" seguida de una secuencia de 15 caracteres numéricos que genera una función basada en la indicación de fecha y hora.)

Si se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* se puede utilizar como nombre de un índice creado para dar soporte a la restricción. Consulte el apartado Notas para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

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

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

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

Consulte UNIQUE en la descripción de *restricción-unicidad* a continuación.

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

Consulte *cláusula-referencias* en “CREATE TABLE”.

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

definición-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 13. Si una columna se define como un tipo XML o estructurado, no se puede especificar una cláusula DEFAULT.

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

Tabla 13. Valores por omisión (cuando no se especifica ningún valor)

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

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

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

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

- representar un valor que pueda asignarse a la columna de acuerdo con las normas de asignación descritas en el Capítulo 3
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- ser una constante numérica o un valor especial de coma flotante si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y después se convierten a coma flotante decimal. Para las columnas DECFLOAT(16), las constantes decimales deben tener una precisión igual o inferior a 16.
- no tener dígitos 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 bytes, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificado y paréntesis, cuando la constante es el argumento de una *función-conversión*.

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

Para las filas existentes, el valor del registro especial de CURRENT SCHEMA al procesarse la sentencia ALTER TABLE.

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 se trata de un tipo diferenciado. Si la función-conversión es BLOB, la constante debe ser de tipo serie.

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

registro-especial-usuario
Especifica CURRENT USER, SESSION_USER o SYSTEM_USER. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser el tipo de datos serie con una longitud mínima de 8 bytes. Si la *función-conversión* es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

CURRENT SCHEMA
Especifica el valor del registro especial CURRENT SCHEMA. El tipo de datos del tipo de fuente del tipo diferenciado de la columna debe ser una serie de caracteres con una longitud mayor que o igual al atributo de longitud

del registro especial CURRENT SCHEMA. Si la función-conversión es BLOB, el atributo de longitud debe ser como mínimo 8 bytes.

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()
Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

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

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.

AS (*expresión-generación***)**
Especifica que la definición de la columna se basa en una expresión. Requiere que la tabla se coloque en estado pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY con las opciones IMMEDIATE CHECKED y FORCE GENERATED para actualizar y comprobar todos los valores de esa columna con la nueva expresión. 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 puede ser DATE, TIME, TIMESTAMP, XML ni un tipo de datos estructurado (SQLSTATE 42842). Si el tipo de datos base es una serie de caracteres de longitud variable, esta cláusula se pasa por alto.

Los valores de serie de caracteres que tienen una longitud 0 se comprimen automáticamente si una tabla se ha establecido con VALUE COMPRESSION.

COLUMN SECURED WITH *nombre-etiqueta-seguridad*
Identifica una etiqueta de seguridad existente para la política de seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064).

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 exclusiva compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez. El nombre no puede estar calificado. El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar como parte de una clave exclusiva ningún LOB, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008). El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.) Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515).

Se realiza una comprobación para determinar si un índice existente coincide con la definición de clave exclusiva (omitiendo cualquier columna INCLUDE del índice). Una definición de índice coincide si identifica el mismo conjunto de columnas sin tener en cuenta el orden de las columnas ni las especificaciones de dirección (ASC/DESC). 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 bidireccional exclusivo para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado Notas para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

PRIMARY KEY *...(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 ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para los límites de longitud de clave, consulte "Límites de SQL". La tabla no debe tener una clave primaria y las columnas identificadas deben definirse como NOT NULL. No se puede utilizar como parte de una clave primaria ningún LOB, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008). El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.) Cualquier valor existente en el conjunto de columnas identificadas debe ser exclusivo (SQLSTATE 23515).

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 bidireccional exclusivo para las columnas, tal como se describe en CREATE TABLE. Consulte el apartado Notas para ver los detalles sobre los nombres de índice asociados a restricciones de unicidad.

En una tabla sólo se puede definir una clave primaria.

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

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

ADD *cláusula-distribución*
Define una clave de distribución. La tabla debe definirse en un espacio de tablas de un grupo de particiones de base de datos de partición única (SQLSTATE 55037) y no puede tener ya una clave de distribución (SQLSTATE 42889). Si ya existe una clave de distribución para la tabla, la clave existente debe eliminarse antes de añadir la clave de distribución nueva. No puede añadirse una clave de distribución a una tabla que sea una subtabla (SQLSTATE 428DH) o a una tabla con una columna de tipo de datos XML (SQLSTATE 42997).

DISTRIBUTE BY HASH (*nombre-columna***...)**
Define una clave de distribución utilizando las columnas especificadas. Cada *nombre-columna* debe identificar una columna de la tabla, y la misma columna no debe identificarse más de una vez. El nombre no puede estar calificado. No se puede utilizar una columna como parte de una clave de distribución si el tipo de datos de la columna es BLOB, CLOB, DBCLOB, XML, un tipo distinto de estos tipos o un tipo estructurado.

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 exclusivo ni activador definido.
- No debe hacer referencia a un apodo marcado con el almacenamiento en antememoria inhabilitado.
- No debe existir una referencia a ésta en la definición de otra tabla de consultas materializadas.
- No debe existir una referencia a ésta en la definición de una vista habilitada para la optimización de consulta.

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

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

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

que las columnas resultantes de la *selección completa* (SQLSTATE 428EW). Para obtener información detallada acerca de la especificación de la *selección completa* para una tabla de 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 consulta materializada definidas con este atributo no permiten 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*
Altera los atributos de restricción de la restricción de comprobación o

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

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

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

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

DISABLE QUERY OPTIMIZATION
No se puede utilizar la restricción para la optimización de consulta.

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

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

NOT ENFORCED
Cambie la restricción por NOT ENFORCED. Sólo debe especificarse si, independientemente, se sabe que los datos de tabla se ajustan a la restricción. Los resultados de la consulta pueden ser imprevisibles si los datos no se ajustan realmente a la restricción.

ALTER *modificación-columna*
Modifica la definición de una columna. Sólo se modificarán los atributos especificados; los demás permanecerán sin modificar. No se pueden alterar las columnas de una tabla con tipo (SQLSTATE 428DH).

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

SET DATA TYPE *tipo-datos-modificado*
Especifica el tipo de datos nuevo de la columna. El tipo de datos nuevo debe ser compatible con el tipo de datos actual de la columna (SQLSTATE 42837). Para obtener más información sobre los tipos de datos incorporados, consulte “CREATE TABLE”. En la Tabla 14 en la página 108 se listan los tipos de datos compatibles. La columna "Se recomienda Reorg" identifica las modificaciones de tipo de datos que requerirán la reorganización de la tabla para poder acceder de nuevo totalmente a una tabla (SQLSTATE 57016). En estos casos, la columna modificada no puede formar parte de una tabla que contiene una columna de tipo de datos XML (SQLSTATE 42997).

El tipo de datos de una columna de identidad no se puede modificar (SQLSTATE 42997).

La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997).

La longitud, precisión o escala especificada puede ser superior o igual a la longitud, precisión o escala actual, pero no inferior a ella (SQLSTATE 42837).

Tabla 14. Tipos de datos compatibles

Del tipo	Al tipo	Válido para la columna de la clave de particionamiento	Válido para la columna de dimensión de organización de MDC	Válido para la columna de clave de distribución	Reorganización recomendada
SMALLINT	INTEGER	Sí	Sí	Sí	Sí
SMALLINT	BIGINT	Sí	Sí	Sí	Sí
SMALLINT	DECIMAL(*p*,*m*); *p*-*m* >4	Sí	Sí	No	Sí
SMALLINT	REAL	Sí	Sí	No	Sí
SMALLINT	DOUBLE	Sí	Sí	No	Sí
SMALLINT	DECFLOAT(16) o DECFLOAT(34)	Sí	Sí	No	Sí
INTEGER	BIGINT	Sí	Sí	Sí	Sí
INTEGER	DECIMAL(*p*,*m*); *p*-*m* >9	Sí	Sí	No	Sí
INTEGER	DOUBLE	Sí	Sí	No	Sí
INTEGER	DECFLOAT(16) o DECFLOAT(34)	Sí	Sí	No	Sí
BIGINT	DECIMAL(*p*,*m*); *p*-*m* >19	Sí	Sí	No	Sí
BIGINT	DECFLOAT(34)	Sí	Sí	No	Sí
REAL	DOUBLE	Sí	Sí	Sí	Sí
REAL	DECFLOAT(16) o DECFLOAT(34)	Sí	Sí	No	Sí
DOUBLE	DECFLOAT(16) o DECFLOAT(34)	Sí	Sí	No	Sí
DECIMAL(*n*,*m*)	DECIMAL (*p*, *q*); *p* >= *n*; *q* >= *m*; (*p*-*q*) >= (*n*-*m*)	Sí	Sí	No	Sí
DECIMAL (*n*, *m*); *n* <= 15	DECFLOAT(16)	Sí	Sí	No	Sí
DECIMAL(*n*,*m*)	DECFLOAT(34)	Sí	Sí	No	Sí
DECFLOAT(16)	DECFLOAT(34)	Sí	Sí	Sí	Sí
CHARACTER(*n*)	CHARACTER (*n*+*x*)	No	Sí	Sí	Sí
CHARACTER(*n*)	VARCHAR (*n*+*x*)	No	Sí	Sí	Sí
VARCHAR (*n*)	CHARACTER (*n*+*x*)	No	Sí	Sí	Sí
VARCHAR (*n*)	VARCHAR (*n*+*x*)	No	Sí	Sí	No
GRAPHIC (*n*)	GRAPHIC (*n*+*x*)	No	Sí	Sí	Sí
GRAPHIC (*n*)	VARGRAPHIC (*n*+*x*)	No	Sí	Sí	Sí
VARGRAPHIC (*n*)	VARGRAPHIC (*n*+*x*)	No	Sí	Sí	No
VARGRAPHIC (*n*)	GRAPHIC (*n*+*x*)	No	Sí	Sí	Sí
BLOB (*n*)	BLOB (*n*+*x*)	n/d	n/d	n/d	Sí
CLOB (*n*)	CLOB (*n*+*x*)	n/d	n/d	n/d	Sí
DBCLOB (*n*)	DBCLOB (*n*+*x*)	n/d	n/d	n/d	Sí

Al modificar una columna, el número total de bytes de todas las columnas no puede superar el tamaño de registro máximo (SQLSTATE 54010). Si se utiliza la columna en una restricción de unicidad o en un índice, la longitud nueva no puede hacer que la suma de las longitudes almacenadas para la restricción de unicidad sean superiores al límite de la longitud de la clave de índice para el tamaño de página (SQLSTATE 54008). Para las

longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para los límites de longitud de clave, consulte "Límites de SQL".

Tabla 15. Efectos en cascada de modificar una columna

Operación	Efecto
Modificación de una columna a la que hace referencia una vista o restricción de comprobación	El objeto se vuelve a generar durante el proceso de modificación. En el caso de una vista, la resolución del método o la función para el objeto puede ser diferente tras la operación de modificación y cambiar la semántica del objeto. En el caso de una restricción de comprobación, si cambia la semántica del objeto como consecuencia de la operación de modificación, la operación falla.
Modificación de una columna de una tabla que tiene un paquete, activador o rutina SQL dependiente	El objeto se marca como no válido y se vuelve a validar la próxima vez que se utiliza.
Modificación del tipo de una columna de una tabla a la que hace referencia un XSROBJECT habilitado para la descomposición	El XSROBJECT se marca como inoperativo para la descomposición. Para volver a habilitar el XSROBJECT, puede que sea necesario volver a ajustar sus correlaciones; a continuación, emita una sentencia ALTER XSROBJECT ENABLE DECOMPOSITION para el XSROBJECT.
Modificación de una columna a la que se hace referencia en la expresión por omisión de una variable global	La expresión por omisión de la variable global se valida durante el proceso de modificación. Si no se puede resolver una función definida por el usuario en la expresión por omisión, la operación falla.

SET *modificación-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". Al modificar el valor por omisión, no se 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 puede hacer referencia a la columna en la columna de clave de distribución ni en la cláusula ORGANIZE BY (SQLSTATE 42997).

SET EXPRESSION AS (*expresión-generación***)**
Cambia la expresión de la columna por la *expresión-generación* especificada. SET EXPRESSION AS requiere que la tabla se coloque en estado pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF. Después de la sentencia ALTER TABLE, se debe utilizar la sentencia SET INTEGRITY con las opciones IMMEDIATE CHECKED y FORCE GENERATED para actualizar y comprobar todos los valores de esa columna con la nueva expresión. La columna ya debe estar definida como una columna generada basada en una expresión (SQLSTATE 42837) y no puede haber aparecido en las cláusulas PARTITIONING KEY, DIMENSIONS o KEY SEQUENCE de la tabla (SQLSTATE 42997). La expresión de generación debe seguir las mismas normas que se aplican al definir una columna generada. El tipo de datos resultante de la expresión de generación se debe poder asignar al tipo de datos de la columna (SQLSTATE 42821).

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

El tipo de datos de *nombre-columna* debe ser un tipo XML o 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 en la fila de tabla base los datos que ya se habían almacenado por separado del resto de la fila. Para utilizar la longitud en línea modificada de una columna de tipo estructurado, invoque el mandato REORG para la tabla especificada tras haber modificado la longitud en línea de su columna. Para utilizar la longitud en línea modificada de una columna de tipo XML en una tabla existente, deberá actualizar todas las filas con una sentencia UPDATE. El mandato REORG no tiene efecto alguno en el almacenamiento de filas de los documentos XML.

SET NOT NULL
Especifica que la columna no puede contener valores nulos. Ningún valor de esta columna en las filas existentes de la tabla puede ser nulo (SQLSTATE 23502). Esta cláusula no está permitida si la columna se especifica en la clave foránea de una restricción de referencia con la norma DELETE establecida en SET NULL y no existe ninguna otra columna anulable en la clave foránea (SQLSTATE 42831). Para modificar este atributo para una columna, es necesaria la reorganización de la tabla para poder acceder posteriormente a la tabla (SQLSTATE 57016). Tenga en cuenta que, dado que esta operación requiere la validación de los datos de la tabla, no se puede realizar si la tabla se encuentra en estado pendiente de reorganización (SQLSTATE 57016). La columna modificada no puede formar parte de una tabla que contiene una columna de tipo de datos XML (SQLSTATE 42997). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997).

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
Especifica que la columna es una columna de indicación de fecha y hora para la tabla. Se genera un valor para la columna en cada fila que se inserta y para cualquier fila de cualquier columna que se actualiza. El valor que se genera para una columna ROW CHANGE TIMESTAMP es una indicación de fecha y hora que corresponde a la hora de inserción o actualización de la fila. Si se insertan o actualizan varias filas con una sentencia única, el valor de la columna ROW CHANGE TIMESTAMP puede ser diferente para cada fila.

Una tabla sólo puede tener una columna ROW CHANGE TIMESTAMP (SQLSTATE 428C1). Si se especifica *tipo-datos*, debe ser TIMESTAMP (SQLSTATE 42842). Una columna ROW CHANGE TIMESTAMP no puede tener una cláusula DEFAULT (SQLSTATE 42623). Debe especificarse NOT NULL para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42831).

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).

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 se pueda asignar a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser superior o igual al valor mínimo (SQLSTATE 42815).

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

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

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

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

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

NO CACHE
Especifica que no se deben preasignar los valores de la columna de identidad. En un entorno 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 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).

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

DROP NOT NULL
Descarta el atributo NOT NULL de la columna, lo que permite a la columna tener un valor nulo. Esta cláusula no está permitida si la columna se especifica en la clave primaria o en una restricción de unicidad de la tabla (SQLSTATE 42831). Para modificar este atributo para una columna, es necesaria la reorganización de la tabla para poder acceder posteriormente a la tabla (SQLSTATE 57016). La columna modificada no puede formar parte de una tabla que contiene una columna de tipo de datos XML (SQLSTATE 42997). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997).

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

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

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

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

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

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

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

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

SECURED WITH *nombre-etiqueta-seguridad*
Identifica una etiqueta de seguridad existente para la política de seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064).

DROP COLUMN SECURITY
Modifica una columna para establecerla como no protegida.

DROP PRIMARY KEY
Elimina la definición de la clave primaria y todas las restricciones de referencia dependientes de esta clave primaria. La tabla ya debe tener una clave primaria (SQLSTATE 42888).

DROP FOREIGN KEY *nombre-restricción*
Elimina la restricción de referencia *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de referencia (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción de referencia, consulte el apartado Notas.

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 de unicidad existente (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción de unicidad, consulte el apartado Notas.

DROP CHECK *nombre-restricción*
Elimina la restricción de comprobación *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación existente definida en la tabla (SQLSTATE 42704).

DROP CONSTRAINT *nombre-restricción*
Elimina la restricción *nombre-restricción*. El *nombre-restricción* debe identificar una restricción de comprobación, restricción de referencia, clave primaria o restricción de unicidad existente definida en la tabla (SQLSTATE 42704). Para obtener información sobre las implicaciones de descartar una restricción, consulte el apartado Notas.

DROP COLUMN
Descarta la columna identificada de la tabla. La tabla no debe ser una tabla con tipo (SQLSTATE 428DH). La tabla no puede tener habilitada la captura de datos (SQLSTATE 42997). Si descarta una columna, debe volver a organizar la tabla para poder ejecutar UPDATE, INSERT, DELETE o realizar una exploración INDEX para la tabla (SQLSTATE 57016).

nombre-columna
Identifica la columna que se eliminará. El nombre de columna no puede estar calificado. El nombre debe identificar una columna de la tabla especificada (SQLSTATE 42703). El nombre no puede identificar la única columna de la tabla (SQLSTATE 42814). El nombre no puede identificar una columna que forma parte de la clave de distribución, clave de particionamiento de la tabla o dimensiones de organización de la tabla

(SQLSTATE 42997). El nombre no puede identificar una columna que forma parte de una tabla que contiene una columna de tipo de datos XML (SQLSTATE 42997).

CASCADE
Especifica que también se descartan las vistas, los índices, los activadores, las funciones SQL y las restricciones o las variables globales dependientes de la columna que se descarta, o que los objetos XSROBJECT con la descomposición habilitada que dependen de la tabla que contiene la columna se establecen como inoperativos para la descomposición. Un activador depende de la columna si se le hace referencia en la lista de columnas UPDATE OF o en cualquier ubicación de la acción activada. Un XSROBJECT con la descomposición habilitada depende de una tabla si contiene una correlación de un atributo o elemento XML con la tabla. Si una función SQL o variable global depende de otro objeto de base de datos, tal vez no sea posible descartar la función o variable global mediante la opción CASCADE. CASCADE es el valor por omisión.

RESTRICT
Especifica que no se puede descartar la columna si hay vistas, índices, activadores, restricciones o variables globales dependientes de la columna o si hay un XSROBJECT con la descomposición habilitada dependiente de la tabla que contiene la columna (SQLSTATE 42893). Un activador depende de la columna si se le hace referencia en la lista de columnas UPDATE OF o en cualquier ubicación de la acción activada. Un XSROBJECT con la descomposición habilitada depende de una tabla si contiene una correlación de un atributo o elemento XML con la tabla. El primer objeto dependiente que se detecta se identifica en las anotaciones cronológicas de administración.

Tabla 16. Efectos en cascada de eliminar una columna

Operación	Efecto de RESTRICT	Efecto de CASCADE
Descartar una columna a la que hace referencia una vista o un activador	No se puede descartar la columna.	El objeto y todos los objetos dependientes de ese objeto se descartan.
Descartar una columna a la que se hace referencia en la clave de un índice	Si todas las columnas a las que se hace referencia en el índice se descartan en la misma sentencia ALTER TABLE, se puede eliminar el índice. De lo contrario, no se puede descartar la columna.	Se descarta el índice.
Descartar una columna a la que se hace referencia en una restricción de unicidad	Si todas las columnas a las que se hace referencia en la restricción de unicidad se descartan en la misma sentencia ALTER TABLE, y no hace referencia a la restricción de unicidad una restricción de referencia, se descartan las columnas y la restricción. (El índice empleado para satisfacer la restricción también se descarta.) De lo contrario, no se puede descartar la columna.	La restricción de unicidad y las restricciones de referencia que hacen referencia a esa restricción de unicidad se descartan. (Los índices empleados por esas restricciones también se descartan.)

Tabla 16. Efectos en cascada de eliminar una columna (continuación)

Operación	Efecto de RESTRICT	Efecto de CASCADE
Descartar una columna a la que se hace referencia en una restricción de referencia	Si todas las columnas a las que se hace referencia en la restricción de referencia se descartan en la misma sentencia ALTER TABLE, se descartan las columnas y la restricción. De lo contrario, no se puede descartar la columna.	Se descarta la restricción de referencia.
Descartar una columna a la que hace referencia una columna generada por el sistema que no se descarta	No se puede descartar la columna.	No se puede descartar la columna.
Descartar una columna a la que se hace referencia en una restricción de comprobación	No se puede descartar la columna.	Se descarta la restricción de comprobación.
Descartar una columna a la que se hace referencia en un XSROBJECT con la descomposición habilitada	No se puede descartar la columna.	El XSROBJECT se marca como inoperativo para la descomposición. Para volver a habilitar el XSROBJECT, puede que sea necesario volver a ajustar sus correlaciones; a continuación, emita una sentencia ALTER XSROBJECT ENABLE DECOMPOSITION para el XSROBJECT.
Descartar una columna a la que se hace referencia en la expresión por omisión de una variable global	No se puede descartar la columna.	Se descarta la variable global, a menos que no esté permitido descartar la variable global porque hay otros objetos, que no permiten la cascada, que dependen de la variable global.

DROP RESTRICT ON DROP
Elimina la restricción, si existe, sobre la operación de eliminar la tabla y el espacio de tablas que contiene la tabla.

DROP DISTRIBUTION
Descarta la definición de distribución de la tabla. La tabla debe tener una definición de distribución (SQLSTATE 428FT). El espacio de tablas de la tabla debe estar definido en un grupo de particiones de base de datos de partición única.

DROP MATERIALIZED QUERY
Cambia una tabla de consultas materializadas de modo que ya no se considera 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 cambia, pero ya no se puede emplear la tabla para la optimización de consulta, y ya no se puede utilizar la sentencia REFRESH TABLE.

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

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

La captura de datos es incompatible con la compresión de filas (SQLSTATE 42997).

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 está definida de modo que admite datos de una partición de base de datos 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 particiones de base de datos distintas de la partición de catálogo), esta opción no está permitida (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 la ejecución de esta sentencia y la finalización de la unidad de trabajo en la que se ha ejecutado.

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

Para obtener más información sobre el atributo NOT LOGGED INITIALLY, consulte la descripción de este atributo en "CREATE TABLE".

Nota: Si se produce actividad que no es de anotaciones cronológicas para una tabla que tiene activado el atributo NOT LOGGED INITIALLY y si una sentencia no se ejecuta satisfactoriamente (dando lugar a una 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. Los índices existentes en la tabla también se suprimen.

No se puede vaciar una tabla particionada con particiones de datos enlazadas (SQLSTATE 42928).

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

LOCKSIZE
Indica el tamaño (granularidad) de los bloqueos utilizados cuando se accede a la tabla. El uso de esta opción en la definición de tabla no impedirá que se produzca una escala de bloqueo normal. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

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

BLOCKINSERT
Indica el uso de bloqueos de bloques en las operaciones de inserción. Esto

significa que se adquiere el bloqueo exclusivo adecuado en el bloque antes de la inserción y el bloqueo de fila no se realiza en la fila insertada. Esta opción resulta de utilidad al insertar distintas transacciones en diferentes celdas de la tabla. Las transacciones que efectúan inserciones en las mismas celdas pueden seguir haciéndolo de manera simultánea, pero realizarán las inserciones en distintos bloques, y esto puede afectar al tamaño de la celda si se necesitan más bloques. Esta opción solo es válida para las tablas MDC (SQLSTATE 628N).

TABLE
Indica la utilización de bloqueos de tabla. Esto significa que se adquiere el bloqueo de compartimiento o exclusivo adecuado en la tabla y no se utilizan bloqueos de intento (excepto el valor de ningún intento). En el caso de las tablas particionadas, esta técnica de bloqueo se aplica tanto al bloqueo de tabla como a los bloqueos de partición de datos para las particiones de datos a las que se accede. El uso de este valor puede mejorar el rendimiento de las consultas limitando el número de bloqueos que es necesario adquirir. Sin embargo, también se reduce la simultaneidad, dado que se retienen todos los bloqueos en toda la tabla.

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

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

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

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

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

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

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

COMPRESS
Especifica si se aplica la compresión de datos a las filas de la tabla.

YES
Especifica que la compresión de filas de datos está habilitada. Las operaciones de inserción y actualización de la tabla estarán sujetas a compresión. Si no existe ningún diccionario de compresión para la tabla, se crea uno automáticamente y las filas están sujetas a compresión después de que la tabla se haya llenado de datos de forma suficiente. Si existe un diccionario de compresión para la tabla, se reactiva la compresión para utilizar este diccionario y las filas están sujetas a compresión.

NO
Especifica que la compresión de filas de datos está inhabilitada. Las operaciones de inserción y actualización que se realicen en la tabla ya no estarán sujetas a compresión. Las filas de la tabla que tengan el formato comprimido permanecerán en formato comprimido hasta que se conviertan al formato no comprimido cuando se actualicen. Al efectuarse una reorganización no in situ de la tabla, se descomprimen todas las filas comprimidas. Si existe un diccionario de compresión, este se descarta al reinicializar o truncar la tabla (por ejemplo, en una operación de sustitución).

VALUE COMPRESSION
Determina el formato de fila que debe utilizarse. Cada tipo de datos tiene un número total de bytes distinto según el formato de fila que se utilice. Si desea más información, consulte "Número total de bytes" en "CREATE TABLE". Una operación de actualización causa que una fila existente pase a tener el nuevo formato de fila. Para mejorar el rendimiento de las operaciones de actualización en las filas existentes, se recomienda la reorganización de tablas fuera de línea. Esta acción también puede dar como resultado que la tabla ocupe menos espacio. Si el tamaño de la fila, calculado utilizando la columna correspondiente de la tabla llamada "Número total de bytes de las columnas por tipo de datos" (consulte "CREATE TABLE"), ya no cabe dentro del límite del tamaño de fila, indicado en la tabla llamada "Límites para el número de columnas y el tamaño de fila de cada tamaño de página del espacio de tablas", se devolverá un error (SQLSTATE 54010). Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

ACTIVATE
El valor NULL se almacena utilizando tres bytes. Es el mismo espacio o menor que cuando VALUE COMPRESSION no se ha activado para las columnas de todos los tipos de datos, con la excepción de CHAR(1). El hecho de definir o no una columna como anulable no afecta al cálculo del tamaño de fila. Los valores de datos de longitud cero para las columnas cuyo tipo de datos es VARCHAR, VARGRAPHIC, LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB o BLOB deben almacenarse utilizando dos bytes únicamente, que es un valor inferior al almacenamiento necesario cuando VALUE COMPRESSION no se ha activado. Cuando una columna se define con la opción COMPRESS SYSTEM DEFAULT, se permite también que el valor por omisión del sistema perteneciente a la columna se almacene utilizando tres bytes de almacenamiento total. El formato de fila utilizado para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos al actualizar al valor NULL, a un valor de longitud cero o al valor por omisión del sistema o bien al actualizar desde estos valores.

DEACTIVATE
El valor NULL se almacena con espacio establecido aparte para posibles actualizaciones futuras. Este espacio no se establece aparte para las columnas de longitud variable. Además, no da soporte al almacenamiento eficaz de los valores por omisión del sistema correspondientes a una columna. Si la columna ya existe y tiene el atributo COMPRESS SYSTEM DEFAULT, se devolverá un mensaje de aviso (SQLSTATE 01648).

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

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

OFF
Especifica que se realizará una anotación cronológica mínima de cualquier operación de creación de índice de esta tabla. Este valor altera temporalmente el valor del parámetro de configuración de base de datos **logindexbuild**.

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

Normas

- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de distribución, si existe (SQLSTATE 42997).
- Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).
- Únicamente se puede hacer referencia a una columna en una cláusula ADD, ALTER o DROP COLUMN de una sola sentencia ALTER TABLE (SQLSTATE 42711).
- No se puede modificar una longitud de columna o un tipo de datos, ni se puede eliminar la columna, si la tabla tiene alguna tabla de consultas materializadas dependiente de la tabla (SQLSTATE 42997).
- Las columnas de tipo VARCHAR y VARGRAPHIC que se han modificado para ser mayores que 4000 y 2000 respectivamente, no se deben utilizar como parámetros de entrada en las funciones del esquema SYSFUN (SQLSTATE 22001).
- No se puede modificar una longitud de columna si la tabla tiene alguna vista habilitada para la optimización de consulta dependiente de la tabla (SQLSTATE 42997).
- La tabla debe colocarse en estado pendiente de establecer integridad utilizando la sentencia SET INTEGRITY con la opción OFF (SQLSTATE 55019) antes de:
  - Añadir una columna con una expresión de generación
  - Modificar la expresión generada de una columna
  - Cambiar una columna para que contenga una expresión generada

- No se puede añadir una columna de tipo de datos XML a una tabla si hay índices de tipo 1 en esa tabla (SQLSTATE 42997). Los índices se pueden convertir en índices de tipo 2 utilizando el mandato REORG INDEXES con la opción CONVERT.
- No se puede modificar una columna existente para establecerla como de tipo DB2SECURITYLABEL (SQLSTATE 42837).
- La definición de una columna de tipo DB2SECURITYLABEL falla si la tabla no tiene asociada una política de seguridad (SQLSTATE 55064).
- No se puede modificar ni eliminar una columna de tipo DB2SECURITYLABEL (SQLSTATE 42817).
- Una operación ALTER TABLE para marcar una tabla como protegida falla si hay una MQT dependiente de esa tabla (SQLSTATE 55067).
- El enlace de una partición a una tabla particionada protegida falla si la tabla fuente y la tabla de destino no están protegidas con la misma política de seguridad, no tienen la misma columna de etiqueta de seguridad de fila y no tienen el mismo conjunto de columnas protegidas (SQLSTATE 428GE).
- Si se hace referencia a una columna generada en una clave de particionamiento de tabla, no se puede modificar la expresión de columna generada (SQLSTATE 42837).
- La adición o el enlace de una partición de datos a una tabla particionada no se ejecutarán correctamente y se generará el error SQL0612N tras desenlazar el mismo nombre de partición si la limpieza de índices asíncrona no ha completado la supresión de entradas de índice para la partición (SQLSTATE 42711).

Notas

- Se produce una *operación de reorganización recomendada* cuando los cambios resultantes de una sentencia ALTER TABLE afectan al formato de fila de los datos. En este caso, la mayoría de las operaciones posteriores de la tabla quedan restringidas hasta que se ejecuta correctamente una operación de reorganización de la tabla. Pueden ejecutarse hasta tres sentencias ALTER TABLE de este tipo para una tabla antes de que deba realizarse la reorganización (SQLSTATE 57016). Pueden realizarse varias modificaciones que constituirían una operación de REORG recomendada en una única sentencia ALTER TABLE (una por columna); esto se considera una sola operación de REORG recomendada. Por ejemplo, eliminar dos columnas en una única sentencia ALTER TABLE no se considera dos operaciones de REORG recomendada. Sin embargo, si eliminara dos columnas en dos sentencias ALTER TABLE independientes, se consideraría que son dos sentencias que contienen operaciones de REORG recomendada.
- Las operaciones de tabla siguientes están permitidas tras realizarse correctamente una operación de REORG recomendada:
  - ALTER TABLE (sin necesidad de validación de los datos de fila) Sin embargo, las operaciones siguientes no están permitidas (SQLSTATE 57007):
    - ADD CHECK CONSTRAINT
    - ADD REFERENTIAL CONSTRAINT
    - ADD UNIQUE CONSTRAINT
    - ALTER COLUMN SET NOT NULL
  - DROP TABLE
  - RENAME TABLE
  - REORG TABLE
  - TRUNCATE TABLE

  - Acceso de exploración de tabla de datos de tabla
- Al modificar una tabla para convertirla en una tabla de consultas materializadas, la tabla se coloca en estado pendiente de establecer integridad. Si la tabla se define como REFRESH IMMEDIATE, la tabla se debe sacar del estado pendiente de establecer integridad para poder invocar los mandatos INSERT, DELETE o UPDATE en la tabla a la que hace referencia la selección completa. La tabla se puede sacar del estado pendiente de establecer integridad utilizando REFRESH TABLE o SET INTEGRITY, con la opción IMMEDIATE CHECKED, para renovar completamente los datos de la tabla de acuerdo con la selección completa. Si los datos de la tabla reflejan fielmente el resultado de la selección completa, se puede utilizar la opción IMMEDIATE UNCHECKED de SET INTEGRITY para sacar la tabla del estado pendiente de establecer integridad.
- La modificación de una tabla con el fin de convertirla en una tabla de 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 convertir una tabla de consultas materializadas en 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 una 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 modifica con el fin de convertirla en una tabla normal.
- Las cláusulas ADD se procesan antes que todas las demás cláusulas. Las demás cláusulas se procesan en el orden en que se especifican.
- Las columnas añadidas mediante una operación de modificación de tabla no se añadirán automáticamente a ninguna vista existente de la tabla.
- Al añadir o enlazar una partición de datos a una tabla particionada, o desenlazar una partición de datos de una tabla particionada, los paquetes dependientes de esa tabla se invalidan.
- Para eliminar el particionamiento de una tabla, es necesario eliminar la tabla y volver a crearla.
- Para eliminar la organización de una tabla, es necesario eliminar la tabla y volver a crearla.
- Cuando se crea un índice automáticamente para una restricción de clave primaria o de unicidad, el gestor de bases de datos intentará utilizar el nombre de la restricción especificada como el nombre de índice con un nombre de esquema que coincida con el nombre de la tabla. Si esto coincide con un nombre de índice existente o no se ha especificado ningún nombre para la restricción, el índice se crea en el esquema SYSIBM con un nombre generado por el sistema con el formato "SQL" seguido de una secuencia de 15 caracteres numéricos generados por una función basada en la indicación de fecha y hora.
- Si se crea un índice en una tabla particionada con particiones de datos enlazadas, el índice no incluirá los datos de las particiones de datos enlazadas. Utilice la sentencia SET INTEGRITY para mantener todos los índices de todas las particiones de datos enlazadas.

- Se dice que cualquier tabla que participe en una operación DELETE de la tabla T está *conectada-con-supresión* a T. Así, una tabla está conectada con supresión a T si es dependiente de T o es dependiente de una tabla en la que se realizan supresiones de T en cascada.
- Un paquete tiene una utilización de inserción (actualización/supresión) en la tabla T si los registros se insertan en (actualizan en/suprimen de) T, directamente por una sentencia en el paquete, o indirectamente por las restricciones o los activadores ejecutados por el paquete en nombre de una de sus sentencias. De manera similar, un paquete tiene una utilización de actualización sobre una columna si la columna se modifica directamente por una sentencia del paquete, o indirectamente por las restricciones o los activadores ejecutados por el paquete en nombre de una de sus sentencias.
- En un sistema federado, se puede modificar una tabla base remota que se ha creado utilizando un DDL transparente. Sin embargo, el DDL transparente impone algunos límites en las modificaciones que pueden efectuarse:
  - Una tabla base remota sólo se puede modificar añadiendo nuevas columnas o especificando una clave primaria.
  - Entre las clases específicas que admite el DDL transparente se encuentran las siguientes:
    - ADD COLUMN *definición-columna*
    - NOT NULL y PRIMARY KEY en la cláusula *opciones-columna*
    - ADD *restricción-unicidad* (solo PRIMARY KEY)
  - No se puede especificar un comentario en una columna existente de una tabla base remota.
  - No se puede modificar ni eliminar una clave primaria existente en una tabla base remota.
  - La modificación de una tabla base remota invalida cualquier paquete que dependa del apodo asociado a esa tabla base remota.
  - La fuente de datos remota debe dar soporte a los cambios que se piden con la sentencia ALTER TABLE. Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.
  - El intento de modificar una tabla base remota que no se ha creado utilizando un DDL transparente devuelve un error.
- Los cambios (implícitos o explícitos) en la clave primaria, las claves exclusivas o las claves foráneas pueden tener los efectos siguientes en los paquetes, los índices y las demás claves foráneas.
  - Si se añade una clave primaria o exclusiva:
    - Ello no afecta a los paquetes, a las claves foráneas ni a las claves exclusivas existentes. (Si la clave primaria o exclusiva utiliza un índice exclusivo existente que se ha creado en una versión anterior y que no se ha convertido para dar soporte a la unicidad diferida, el índice se convierte y se invalidan los paquetes sujetos a la utilización de UPDATE en la tabla asociada.)
  - Si se descarta una clave primaria o exclusiva:
    - 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 están sujetos a inserción en la tabla de objetos.
- Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave foránea.
- Se invalidan todos los paquetes que estén sujetos a supresión en la tabla padre.
- Se invalidan todos los paquetes que estén sujetos a actualización en una columna como mínimo de la clave padre.

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

- La adición de una columna a una tabla dará como resultado la invalidación de todos los paquetes sujetos a inserción en la tabla alterada. Si la columna añadida es la primera columna de tipo estructurado definido por el usuario de la tabla, también se invalidarán los paquetes sujetos a DELETE en la tabla modificada.
- La adición de una restricción de comprobación o de referencia a una tabla que ya existe y que no está en estado pendiente de establecer integridad, o la modificación de la restricción de comprobación o de referencia existente de NOT ENFORCED a ENFORCED en una tabla existente que no está en estado pendiente de establecer integridad hará que las filas existentes de la tabla se evalúen inmediatamente para la restricción. Si la verificación falla, se devuelve un error (SQLSTATE 23512). Si una tabla está en estado pendiente de establecer integridad, la adición de una restricción de comprobación o de referencia o la modificación de una restricción de NOT ENFORCED a ENFORCED no dará lugar a la aplicación inmediata de la restricción. Emita la sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED para iniciar la aplicación de la restricción.
- La adición, modificación o eliminación de una restricción de comprobación dará lugar a la invalidación de todos los paquetes sujetos a inserción en la tabla de objetos, sujetos a actualización en como mínimo una de las columnas implicadas en la restricción o sujetos a selección que utilicen la restricción para mejorar el rendimiento.
- La adición de una clave de distribución invalida todos los paquetes sujetos a actualización en, como mínimo, una de las columnas de la clave de distribución.
- Una clave de distribución que esté definida por omisión como la primera columna de la clave primaria no se ve afectada por la eliminación de la clave primaria y la adición de una clave primaria diferente.
- Al eliminar una columna o cambiar su tipo de datos, se elimina toda la información de RUNSTATS de la tabla modificada. Se deberá ejecutar RUNSTATS en la tabla una vez que se pueda acceder a ella nuevamente. El perfil estadístico de la tabla se conserva si la tabla no contiene una columna descartada explícitamente.

- Al modificar una columna (para aumentar su longitud o cambiar su tipo de datos o atributo de capacidad de nulos) o eliminar una columna, se invalidan todos los paquetes que hacen referencia (directa o indirectamente a través de una restricción de referencia o un activador) a su tabla.
- Al modificar una columna (para aumentar su longitud o cambiar su tipo de datos o atributo de capacidad de nulos), se vuelven a generar las vistas (excepto las vistas con tipo) dependientes de su tabla. Si se produce un problema al volver a generar una vista de este tipo, se devuelve un error (SQLSTATE 56098). Las vistas con tipo dependientes de la tabla se marcan como no operativas.
- Al modificar una columna para aumentar su longitud o cambiar su tipo de datos, se marcan todos los activadores y funciones SQL como no válidos; estos se vuelven a compilar implícitamente la próxima vez que se utilizan. Si se produce un problema al volver a generar un objeto de este tipo, se devuelve un error (SQLSTATE 56098).
- Al modificar una columna (para aumentar su longitud o cambiar su tipo de datos o atributo de capacidad de nulos), se pueden producir errores (SQLSTATE 54010) al procesar un activador o una función SQL si una sentencia en la que interviene el activador o la función SQL está preparada o vinculada. Puede producirse este problema si la longitud de fila basada en la suma de las longitudes de las variables de transición y las columnas de tabla de transición es excesiva. Si se descarta este activador o esta función SQL, un posterior intento de volver a crear este objeto devolverá un error (SQLSTATE 54040).
- La modificación de una columna de tipo XML o estructurado con el fin de incrementar la longitud en línea invalidará todos los paquetes que hacen referencia a la tabla, ya sea directa o indirectamente por medio de un activador o de una restricción de referencia.
- La modificación de una columna de tipo XML o estructurado con el fin de incrementar la longitud en línea volverá a generar las vistas que dependen de la tabla.
- Cambiar LOCKSIZE para una tabla dará como resultado una invalidación de todos los paquetes que dependan de la tabla alterada.
- Al cambiar VOLATILE o NOT VOLATILE CARDINALITY se obtendrá como resultado una invalidación de todos los paquetes que tienen una dependencia en la tabla modificada.
- ***Duplicación:*** tenga precaución cuando aumente la longitud o cambie el tipo de datos de una columna. La tabla de datos de cambio asociada a una tabla de aplicación podría estar en el límite de tamaño de fila de DB2 o próximo a él. La tabla de datos de cambio se debe modificar antes de modificar la tabla de aplicación, o se deben modificar las dos tablas dentro de la misma unidad de trabajo, para asegurarse de que la modificación puede completarse para ambas tablas. Debe prestarse atención a las copias, que también pueden estar en el límite de tamaño de fila, o cerca de él, o residir en plataformas que no permiten aumentar la longitud de una columna existente.

  Si la tabla de datos de cambio no se modifica antes de que el programa Capture procese los registros de anotación cronológica con los atributos modificados, es probable que el programa Capture falle. Si una copia que contiene la columna modificada no se modifica antes de que se ejecute la suscripción que mantiene la copia, es probable que la suscripción falle.
- Al desenlazar una partición de una tabla protegida, la tabla de destino creada automáticamente por DB2 quedará protegida del mismo modo que la tabla fuente.

- Cuando se modifica una tabla de modo que queda protegida con granularidad en el nivel de fila, se invalidan las secciones de SQL dinámico de la antememoria que dependen de esa tabla. Del mismo modo, también se invalidan los paquetes que dependen de esa tabla.
- Cuando se modifica una columna de una tabla, T, de modo que se convierte en una columna protegida, se invalidan las secciones de SQL dinámico de la antememoria que dependen de la tabla T. Del mismo modo, también se invalidan los paquetes que dependen de la tabla T.
- Cuando se modifica una columna de una tabla, T, de modo que se convierte en una columna no protegida, se invalidan las secciones de SQL dinámico de la antememoria que dependen de la tabla T. Del mismo modo, también se invalidan los paquetes que dependen de la tabla T.
- Para las filas existentes en la tabla, el valor de la columna de etiqueta de seguridad se establece por omisión en la etiqueta de seguridad correspondiente al acceso de grabación del ID de autorización de sesión en el momento de ejecutarse la sentencia ALTER que añade una columna de etiqueta de seguridad de fila.
- ***Consideraciones para columnas ocultas implícitamente:*** Se puede hacer referencia de manera explícita a una columna definida como oculta implícitamente en una sentencia ALTER TABLE. Por ejemplo, se puede modificar o especificar una columna implícitamente oculta como parte de una restricción de referencia, una restricción de comprobación o una definición de tabla de consultas materializadas.
- ***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)
    - Se puede especificar ADD PARTITIONING KEY en lugar de ADD DISTRIBUTE BY HASH (en este caso también se puede especificar la cláusula USING HASHING opcional)
    - Se puede especificar DROP PARTITIONING KEY en vez de DROP DISTRIBUTION
    - Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC siguen estando soportados pero han quedado obsoletos y no son recomendables, en especial para aplicaciones portátiles
  - Para mantener la compatibilidad con las versiones anteriores de DB2 y por coherencia:

- Puede utilizarse una coma para separar varias opciones en la cláusula *modificación-identidad*

– Para mantener la compatibilidad con DB2 para z/OS:
  - Se puede especificar PART en lugar de PARTITION
  - VALUES puede especificarse en lugar de ENDING AT
– 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 bytes. Los valores de la columna no tienen un juego de caracteres asociado y, por lo tanto, no deben convertirse.

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

Ejemplo 3: Suponga que hay una tabla denominada EQUIPMENT definida con las columnas siguientes:

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

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

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

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

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

Ejemplo 4: Modifique la tabla EMPLOYEE. Añada la restricción de comprobación REVENUE definida para que cada empleado deba alcanzar un total de salario y comisiones superior a 30.000 dólares.

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

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

```
ALTER TABLE CUSTOMERS
  ADD DISTRIBUTE BY HASH (BRANCH_NO)
```

Ejemplo 10: Se ha creado la tabla remota EMPLOYEE en un sistema federado utilizando un DDL transparente. Altere la tabla remota EMPLOYEE para añadir las

columnas PHONE_NO y WORK_DEPT; añada, también, una clave primaria en la columna existente EMP_NO y la nueva columna WORK_DEPT.

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

Ejemplo 11: Modifique la tabla DEPARTMENT para añadir la dependencia funcional FD1 y elimine la dependencia funcional FD1 de la tabla DEPARTMENT.

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

ALTER TABLE DEPARTMENT
  DROP CHECK FD1
```

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

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

Ejemplo 13: Asocie la política de seguridad DATA_ACCESS con la tabla EMPLOYEE.

```
ALTER TABLE EMPLOYEE
  ADD SECURITY POLICY DATA_ACCESS
```

Ejemplo 14: Modifique la tabla EMPLOYEE para proteger la columna SALARY.

```
ALTER TABLE EMPLOYEE
  ALTER COLUMN SALARY
  SECURED WITH EMPLOYEESECLABEL
```

Ejemplo 15: Suponga que tiene una tabla denominada SALARY_DATA que está definida con las columnas siguientes:

```
Nombre columna         Tipo de datos
-----------            ---------
EMP_NAME               VARCHAR(50) NOT NULL
EMP_ID                 SMALLINT NOT NULL
EMP_POSITION           VARCHAR(100) NOT NULL
SALARY                 DECIMAL(5,2)
PROMOTION_DATE         DATE NOT NULL
```

Cambie esta tabla para que se puedan almacenar los salarios en una columna DECIMAL(6,2), establezca PROMOTION_DATE como campo opcional que se puede definir en un valor nulo y elimine la columna EMP_POSITION.

```
ALTER TABLE SALARY_DATA
  ALTER COLUMN SALARY SET DATA TYPE DECIMAL(6,2)
  ALTER COLUMN PROMOTION_DATE DROP NOT NULL
  DROP COLUMN EMP_POSITION
```

Ejemplo 16: Añada una columna denominada DATE_ADDED a la tabla BOOKS. El valor por omisión de esta columna es la indicación de fecha y hora actual.

```
ALTER TABLE BOOKS
  ADD COLUMN DATE_ADDED TIMESTAMP
  WITH DEFAULT CURRENT TIMESTAMP
```

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 de base de datos que actualmente no tiene contenedores.
- Modificar el valor PREFETCHSIZE para un espacio de tablas.
- Modificar el valor BUFFERPOOL utilizado para las tablas del espacio de tablas.
- Modificar el valor OVERHEAD para un espacio de tablas.
- Modificar el valor TRANSFERRATE para un espacio de tablas.
- Modificar la política de almacenamiento en antememoria del sistema de archivos para un espacio de tablas.

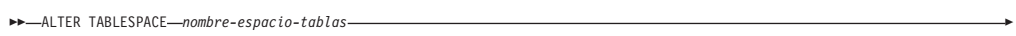
Invocación

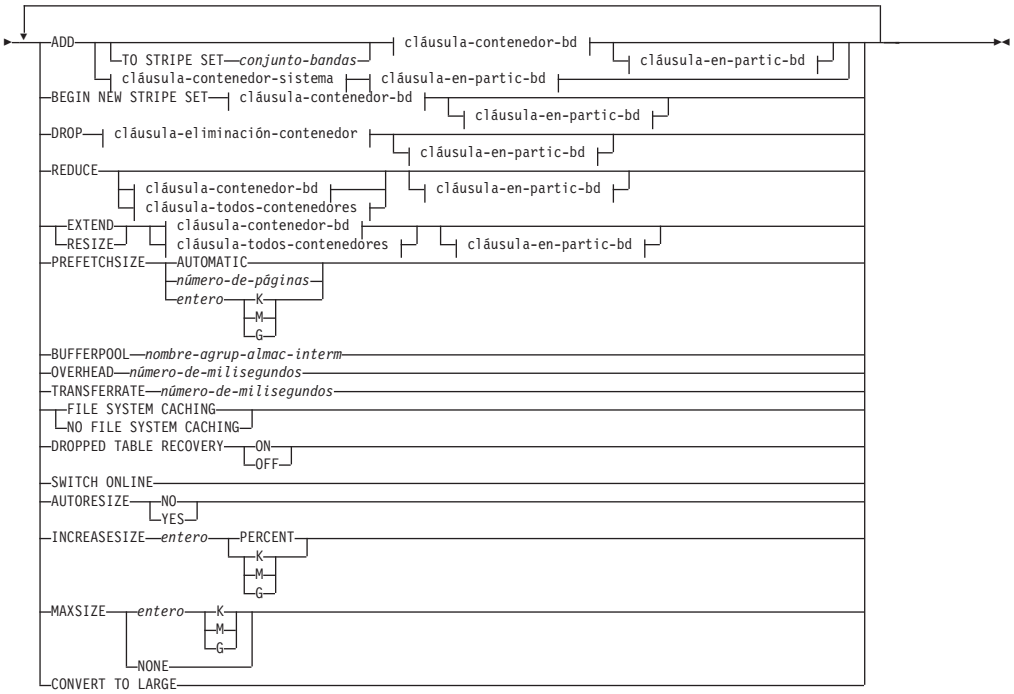
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

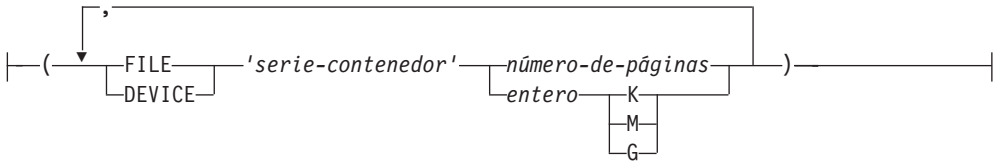
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis

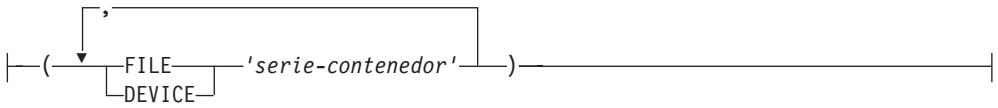




cláusula-contenedor-bd:

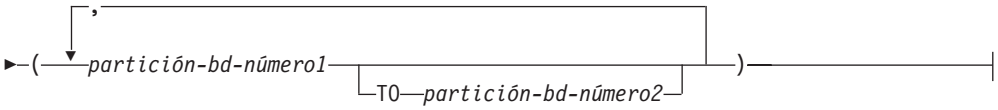


cláusula-eliminación-contenedor:

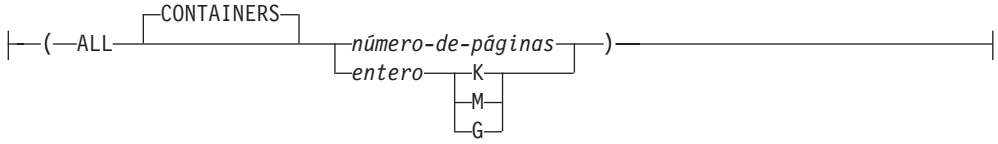


cláusula-contenedor-sistema:

cláusula-en-particiones-bd:



cláusula-todos-contenedores:



Descripción

nombre-espacio-tablas
Nombra el espacio de tablas. Este nombre consta de una sola parte. Es un identificador 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 añaden uno o más contenedores al espacio de tablas y que se colocarán en el conjunto de bandas especificado.

BEGIN NEW STRIPE SET
Especifica que se debe crear un conjunto de bandas nuevo en el espacio de tablas, y que se deben añadir uno o varios contenedores a este conjunto de bandas nuevo. Los contenedores que se añadan posteriormente mediante la opción ADD se añadirán a este nuevo conjunto de bandas a menos que se especifique TO STRIPE SET.

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

REDUCE
Para espacios de tablas de almacenamiento no automático, especifica que va a reducirse el tamaño de los contenedores existentes. El tamaño que se especifica es el tamaño que se reducirá el contenedor existente. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas disminuirán de tamaño. Si la reducción del tamaño da como resultado un tamaño de espacio de tablas que es inferior a la marca de límite superior actual, se intentará reducir la marca de límite superior antes de intentar reducir los contenedores. Para espacios de tablas de almacenamiento no automático, la cláusula REDUCE debe ir seguida de una *cláusula-contenedor-bd* o una *cláusula-todos-contenedores*.

Para los espacios de tablas de almacenamiento automático, especifica que, si se puede, hay que reducir la marca de límite superior actual y que hay que reducir el espacio de tablas según la nueva marca de límite superior. Para espacios de tablas de almacenamiento automático, la cláusula REDUCE no debe ir seguida de una *cláusula-contenedor-bd* o una *cláusula-todos-contenedores*.

EXTEND
Especifica que va a aumentarse el tamaño de los contenedores existentes. El

tamaño especificado es el tamaño en el que se aumenta el contenedor existente. Si se especifica la *cláusula-todos-contenedores*, todos los contenedores del espacio de tablas aumentarán de tamaño.

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

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

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

cláusula-en-partic-bd
Especifica una o más particiones de base de datos para las operaciones del contenedor correspondiente.

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

PREFETCHSIZE
Especifica que deben leerse los datos necesarios para una consulta antes de que la consulta haga referencia a los mismos, de modo que la consulta no deba esperar a que se lleve a cabo la E/S.

AUTOMATIC
Especifica que el tamaño de captación previa de un espacio de tablas debe actualizarse automáticamente; es decir, el tamaño de captación previa se gestionará a través del gestor de bases de datos DB2.

DB2 actualizará el tamaño de captación previa automáticamente siempre que el número de contenedores de un espacio de tablas cambie (seguido por una ejecución satisfactoria de una sentencia ALTER TABLESPACE que añada o desactive uno o más contenedores). El tamaño de captación previa se actualiza al iniciarse 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 de este modo, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor de número de páginas para el tamaño de la captación previa.

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

OVERHEAD *número-de-milisegundos*
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. Tenga en cuenta que el acceso de E/S es demasiado largo o que los datos LOB se almacenan de forma intermedia para los contenedores SMS y DMS.

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

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

DROPPED TABLE RECOVERY
Especifica si las tablas que se han descartado del *nombre-espacio-tablas* pueden recuperarse o no utilizando la opción **RECOVER DROPPED TABLE ON** del mandato ROLLFORWARD DATABASE. Para tablas particionadas, la recuperación de tablas descartadas siempre está activa, incluso cuando dicha recuperación esté desactivada para las tablas sin particiones en uno o más espacios de tablas.

ON
Especifica que las tablas descartadas pueden recuperarse.

OFF
Especifica que las tablas descartadas no pueden recuperarse.

SWITCH ONLINE
Especifica que los espacios de tablas en estado OFFLINE deben ponerse en

línea si sus contenedores han pasado a ser accesibles. Si los contenedores no son accesibles, se devolverá un error (SQLSTATE 57048).

AUTORESIZE
Especifica si debe habilitarse o no la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas gestionado por la base de datos (DMS) o de un espacio de tablas de almacenamiento automático. Los espacios de tablas con dicha posibilidad aumentan de tamaño automáticamente cuando se llenan.

NO
Especifica que debe inhabilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático. Si se inhabilita la posibilidad de cambiar automáticamente el tamaño, no se conservará ningún valor que se haya especificado anteriormente para INCREASESIZE o MAXSIZE.

YES
Especifica que debe habilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

INCREASESIZE *entero* **PERCENT o INCREASESIZE** *entero* **K | M | G**
Especifica la cantidad, por partición de base de datos, en la que aumentará automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente, cuando se llene el espacio de tablas y se haya efectuado una petición de espacio. El valor entero debe ir seguido de:

- PERCENT para especificar la cantidad como porcentaje del tamaño de espacio de tablas en el momento en el que se efectúe una petición de espacio. Cuando se especifique PERCENT, el valor entero debe estar entre 0 y 100 (SQLSTATE 42615).
- K (de kilobytes), M (de megabytes) o G (de gigabytes) para especificar la cantidad en bytes.

Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior o superior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

MAXSIZE *entero* **K | M | G o MAXSIZE NONE**
Especifica el tamaño máximo hasta el cual puede aumentar automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente.

entero
Especifica un límite fijo sobre el tamaño, por partición de base de datos, hasta el cual puede aumentar automáticamente un espacio de tablas DMS o un espacio de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

NONE
Especifica que debe permitirse incrementar el espacio de tablas hasta la capacidad del sistema de archivos o hasta el tamaño máximo del espacio de tablas (descrito en “Límites de SQL y XML”).

CONVERT TO LARGE
Modifica un espacio de tablas DMS normal existente para que sea un espacio de tablas DMS grande. El espacio de tablas y su contenido se bloquean durante la conversión. Esta opción sólo puede utilizarse en espacios de tablas DMS normales. Si se especifica un espacio de tablas SMS, un espacio de tablas

temporal o un espacio de tablas de catálogo del sistema, se devuelve un error (SQLSTATE 560CF). No es posible convertir un espacio de tablas que contiene una partición de datos de una tabla particionada que tiene particiones de datos en otro espacio de tablas (SQLSTATE 560CF). La conversión no puede invertirse una vez que se ha confirmado. Si las tablas del espacio de tablas están definidas con DATA CAPTURE CHANGES, tenga en cuenta los límites de almacenamiento y de capacidad de la tabla de destino y del espacio de tablas.

Normas

- La cláusula BEGIN NEW STRIPE SET no puede especificarse en la misma sentencia que ADD, DROP, EXTEND, REDUCE ni RESIZE, a menos que dichas cláusulas se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- El valor de conjunto de bandas especificado con la cláusula TO STRIPE SET se debe encontrar dentro del rango válido para el espacio de tablas que se está modificando (SQLSTATE 42615).
- Al añadir o eliminar espacio del espacio de tablas, se deben seguir las normas siguientes:
  - EXTEND y RESIZE pueden utilizarse en la misma sentencia, siempre que se aumente el tamaño de cada contenedor (SQLSTATE 429BC).
  - REDUCE y RESIZE pueden utilizarse en la misma sentencia, siempre que se reduzca el tamaño de cada contenedor (SQLSTATE 429BC).
  - EXTEND y REDUCE no pueden utilizarse en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
  - ADD no puede utilizarse con REDUCE ni DROP en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
  - DROP no puede utilizarse con EXTEND ni ADD en la misma sentencia, a menos que se dirijan a particiones de bases de datos distintas (SQLSTATE 429BC).
- La cláusula AUTORESIZE, INCREASESIZE o MAXSIZE no puede especificarse para los espacios de tablas gestionados por el sistema (SMS), los espacios de tablas temporales creados utilizando el almacenamiento automático ni los espacios de tablas DMS definidos para utilizar contenedores de dispositivo sin procesar (SQLSTATE 42601).
- La cláusula INCREASESIZE o MAXSIZE no puede especificarse si el espacio de tablas no permite cambiar el tamaño automáticamente (SQLSTATE 42601).
- Cuando se especifique un nuevo tamaño máximo para un espacio de tablas, el valor debe ser superior al tamaño actual en cada partición de base de datos (SQLSTATE 560B0).
- Las operaciones con contenedores (ADD, EXTEND, RESIZE, DROP o BEGIN NEW STRIPE SET) no pueden realizarse en espacios de tablas de almacenamiento automático porque el gestor de bases de datos controla la gestión del espacio de dichos espacios de tabla (SQLSTATE 42858).
- No pueden añadirse contenedores de dispositivo sin procesar a un espacio de tablas DMS que permita cambiar el tamaño automáticamente (SQLSTATE 42601).
- La cláusula CONVERT TO LARGE no puede especificarse en la misma sentencia que ninguna otra cláusula (SQLSTATE 429BC).
- ***Límites de tamaño de contenedor:*** En espacios de tablas DMS, un contenedor debe tener al menos el doble del tamaño de extensión en páginas de longitud (SQLSTATE 54039). El tamaño máximo de un contenedor depende del sistema operativo.

Notas

- 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 superar los 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 está restringido durante el reequilibrio.
- Si el espacio de tablas se encuentra en estado OFFLINE y los contenedores han pasado a ser accesibles, el usuario puede desconectar todas las aplicaciones y volverlas a conectar a la base de datos para que el espacio de tablas salga del estado OFFLINE. De forma alternativa, la opción SWITCH ONLINE puede activar el espacio de tablas (cambiar su estado OFFLINE) mientras el resto de la base de datos sigue activa y se utiliza.
- Si se añade más de un contenedor a un espacio de tablas, se recomienda añadirlos en la misma sentencia para que el reequilibrio sólo deba efectuarse una vez. Si se intenta añadir contenedores al mismo espacio de tablas en sentencias ALTER TABLESPACE diferentes dentro de una sola transacción, se producirá un error (SQLSTATE 55041).
- Si se intenta ampliar, reducir, cambiar el tamaño o 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.
- La opción REDUCE, RESIZE o DROP intenta liberar, si es necesario, extensiones no utilizadas para espacios de tablas DMS y la opción REDUCE intenta liberar extensiones no utilizadas para espacios de tablas de almacenamiento automático. La eliminación de las extensiones no utilizadas permite que la marca de límite

superior de espacio de tablas se reduzca al valor que fielmente representa la cantidad de espacio utilizado lo que, a su vez, permite mayores reducciones del tamaño del espacio de tablas.

- ***Conversión a espacios de tablas DMS grandes:***
  - Tras la conversión, se recomienda emitir la sentencia COMMIT y luego aumentar la capacidad de almacenamiento del espacio de tablas.
    - Si el espacio de tablas está habilitado para el redimensionamiento automático, el valor del atributo de espacio de tablas MAXSIZE debe aumentarse, a menos que ya esté establecido en NONE.
    - Si el espacio de tablas no está habilitado para el redimensionamiento automático:
      - Habilite el redimensionamiento automático emitiendo una sentencia ALTER TABLESPACE con la opción AUTORESIZE YES, o bien
      - Añada más capacidad de almacenamiento añadiendo conjuntos de bandas, ampliando el tamaño de los contenedores existentes, o realizando ambas acciones.
  - Los índices de las tablas de un espacio de tablas convertido deben reorganizarse o generarse de nuevo para que puedan dar soporte a identificadores de registros de gran tamaño (RID).
    - Los índices pueden reorganizarse mediante el mandato REORG INDEXES ALL (sin la cláusula **CLEANUP ONLY**). Especifique la opción **ALLOW NO ACCESS** para las tablas particionadas.
    - Las tablas también pueden reorganizarse (no INPLACE), lo que dará lugar a que se generen de nuevo todos los índices y se habiliten las tablas para dar soporte a más de 255 filas por tabla.
    - Los índices de tipo 1 regenerados se convierten automáticamente a un índice de tipo 2.
  - Para determinar qué tablas no dan soporte todavía a RID de gran tamaño, utilice la función de tabla ADMIN_GET_TAB_INFO.
- ***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

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 de base de datos 0, 1 y 2. Añada un contenedor nuevo a la partición de base de datos 0. Amplíe todos los contenedores de la partición de base de datos 1. Cambie el tamaño de un contenedor en todas las particiones de bases de datos, excepto en las especificadas de forma explícita (es decir, las particiones de bases de datos 0 y 1).

```
ALTER TABLESPACE 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 en este ejemplo, y se ejecutará en la partición de base de datos 2, ya que otras particiones se envían de forma explícita a las particiones de base de datos 0 y 1. No obstante, si sólo hubieran existido estas dos particiones de base de datos, la sentencia se habría ejecutado correctamente, pero habría devuelto una advertencia (SQL1758W) indicando que los contenedores por omisión se especifican pero no se utilizan.

Ejemplo 8: Habilite la opción de cambiar el tamaño automáticamente para el espacio de tablas DMS_TS1 y establezca su tamaño máximo en 256 megabytes.

```
ALTER TABLESPACE DMS_TS1
  AUTORESIZE YES MAXSIZE 256 M
```

Ejemplo 9: Habilite la opción de cambiar el tamaño automáticamente para el espacio de tablas AUTOSTORE1 y cambie su índice de incremento por el de un 5%.

```
ALTER TABLESPACE AUTOSTORE1
  AUTORESIZE YES INCREASESIZE 5 PERCENT
```

Ejemplo 10: Cambie el índice de incremento por el de 512 kilobytes para un espacio de tablas denominado MY_TS que permite cambiar el tamaño automáticamente, y establezca su tamaño máximo de forma que sea el mayor posible.

```
ALTER TABLESPACE MY_TS
  INCREASESIZE 512 K MAXSIZE NONE
```

ALTER THRESHOLD

La sentencia ALTER THRESHOLD modifica la definición de un umbral.

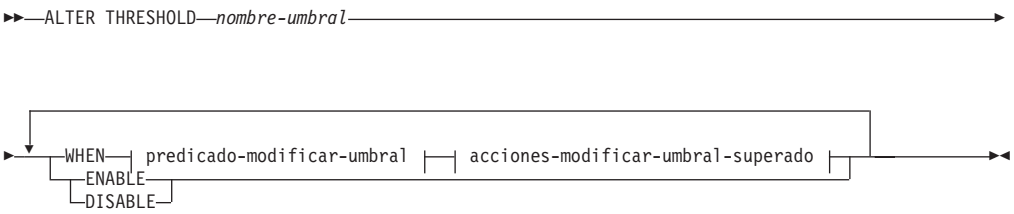
Invocación

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

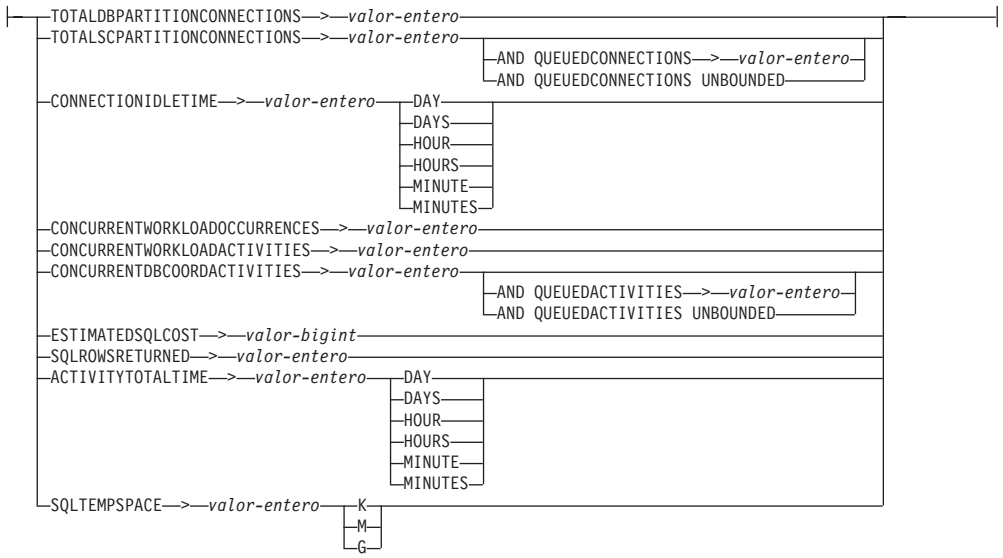
Autorización

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

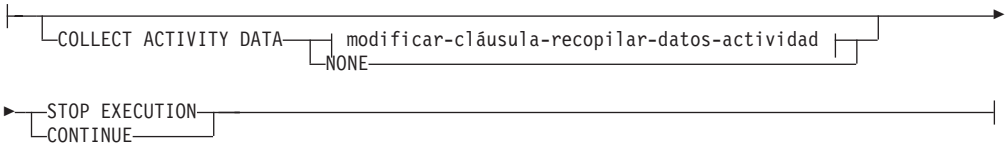
Sintaxis



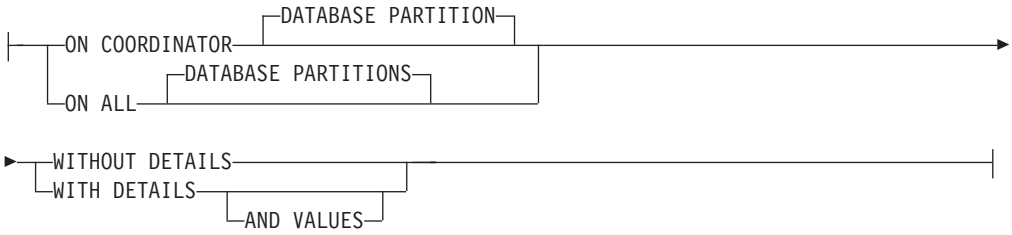
predicado-umbral-superado:



acciones-modificar-umbral-superado:



modificar-cláusula-recopilar-datos-actividad:



Descripción

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

WHEN *predicado-modificar-umbral*
Sustituye el valor de vinculación superior en la condición de umbral por un nuevo valor de vinculación superior. La condición del umbral no puede cambiarse por una condición diferente.

TOTALDBPARTITIONCONNECTIONS > *valor-entero*
Esta condición define una vinculación superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en una partición de base de datos. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá la conexión de cualquier conexión de coordinador nueva. Proseguirán todas las conexiones en cola o en ejecución en la actualidad.

TOTALSCPARTITIONCONNECTIONS > *valor-entero*
Esta condición define una vinculación superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en una partición de base de datos de una superclase de servicio específica. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá que cualquier conexión nueva se una a la clase de servicio. Proseguirán todas las conexiones en cola o en ejecución en la actualidad.

AND QUEUEDCONNECTIONS > *valor-entero* **o AND QUEUEDCONNECTIONS UNBOUNDED**
Especifica un tamaño de cola para el momento en que se supere el número máximo de conexiones de coordinador. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna conexión de coordinador. Especificar UNBOUNDED pondrá en cola todas las conexiones que superen el número máximo especificado de conexiones de coordinador y las *acciones-umbral-superado* nunca se ejecutarán.

CONNECTIONIDLETIME > *valor-entero* **(DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)**
Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá a una conexión permanecer desocupada. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para

especificar una unidad de tiempo apropiada para *valor-entero*. Esta condición se implanta lógicamente en la partición de base de datos del coordinador.

Si se especifica la acción STOP EXECUTION con umbrales CONNECTIONIDLETIME, la conexión para la aplicación se descarta cuando se supera el umbral. Cualquier intento posterior por parte de la aplicación para acceder al servidor de datos no recibirá SQLSTATE 5U026.

Puesto que la granularidad mínima para este umbral es de cinco minutos, los valores especificados se redondean al múltiplo de cinco minutos diferente a cero que más se aproxime a cero. En los casos en los que el redondeo genere un desbordamiento o un subdesbordamiento, se seleccionará el valor más próximo del rango de entero.

El valor máximo par este umbral es de 2 147 400 segundos. Cualquier valor especificado que tenga un número de segundos equivalente superior a 2 147 400 segundos se establecerá en este número de segundos.

Es posible que los cambios en la variable de registro DB2CHECKCLIENTINTERVAL que ocasionan comprobaciones menos frecuentes afecten a la granularidad de este umbral.

CONCURRENTWORKLOADOCCURRENCES > *valor-entero*
Esta condición define una vinculación superior sobre el número de apariciones simultáneas para la carga de trabajo en cada partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820).

CONCURRENTWORKLOADACTIVITIES > *valor-entero*
Esta condición define una vinculación superior sobre el número de actividades de coordinador simultáneas y actividades anidadas para la carga de trabajo en cada partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820).

Cada actividad anidada debe satisfacer las condiciones siguientes:

- Debe ser una actividad de coordinador reconocida. No se contará ninguna actividad de coordinador anidada que no caiga dentro de los tipos de actividades reconocidos. De modo análogo, no se contará ninguna actividad de subagente anidado, como por ejemplo las peticiones de nodo remoto.
- Debe invocarse directamente desde la lógica de usuario, como por ejemplo un procedimiento escrito por el usuario que emita sentencias de SQL.

En consecuencia, las actividades de coordinador anidado que se iniciaron automáticamente bajo la invocación de un programa de utilidad de DB2 o rutinas de los esquemas SYSIBM, SYSFUN o SYSPROC no se cuentan respecto a la vinculación superior especificada por medio de este umbral.

Este umbral no cuenta las actividades SQL internas, como por ejemplo las generadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que son iniciadas por medio del gestor de base de datos y no invocadas directamente mediante la lógica de usuario.

CONCURRENTDBCOORDACTIVITIES > *valor-entero*
Esta condición define una vinculación superior sobre el número de actividades de coordinador de base de datos reconocidas que pueden ejecutarse simultáneamente en todas las particiones de base de datos del dominio especificado. Este valor puede ser cualquier entero positivo,

incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá la ejecución de las actividades de cualquier coordinador de base de datos nueva. Proseguirán todas las actividades de coordinador de base de datos que estén en cola o en ejecución en la actualidad.

Importante: Antes de utilizar los umbrales CONCURRENTDBCOORDACTIVITIES, familiarícese con los efectos que pueden tener sobre el sistema de base de datos. Para obtener más información, consulte el tema "Umbral CONCURRENTDBCOORDACTIVITIES".

AND QUEUEDACTIVITIES > *valor-entero* **o AND QUEUEDACTIVITIES UNBOUNDED**

Especifica un tamaño de cola para el momento en que se supere el número máximo de actividades de coordinador de base de datos. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna actividad de coordinador de base de datos. Especificar UNBOUNDED pondrá en cola todas las actividades de coordinador de base de datos que superen el número máximo especificado de actividades de coordinador de base de datos y las *acciones-umbral-superado* nunca se ejecutarán.

ESTIMATEDSQLCOST > *valor-bigint*

Esta condición define una vinculación superior para el coste asignado de optimizador (en activaciones de temporizador) de una actividad. Este valor puede ser cualquier entero grande positivo diferente a cero (SQLSTATE 42820). Esta condición se implanta en la partición de base de datos del coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo
- Las actividades de DML anidado que se invocan desde la lógica del usuario. En consecuencia, esta condición no hace el seguimiento de las actividades de DML (a menos que su coste se incluya en la estimación del padre, en cuyo caso se hace un seguimiento indirecto de las mismas) que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

SQLROWSRETURNED > *valor-entero*

Esta condición define una vinculación superior para el número de filas devueltas a una aplicación cliente desde el servidor de la aplicación. Este valor puede ser cualquier entero diferente a cero (SQLSTATE 42820). Esta condición se implanta en la partición de base de datos del coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML
- Actividades de DML anidado que derivan de la lógica del usuario. Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

Los conjuntos de resultados devueltos desde un procedimiento se tratan como actividades individuales por separado. Las filas que devuelve el propio procedimiento no se agregan.

ACTIVITYTOTALTIME > *valor-entero* **(DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)**
Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá que se ejecute una actividad, incluyendo el tiempo que la actividad estuvo en cola. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. Esta condición se implanta lógicamente en la partición de base de datos del coordinador.

Puesto que la granularidad mínima para este umbral es de cinco minutos, los valores especificados se redondean al múltiplo de cinco minutos diferente a cero que más se aproxime a cero. En los casos en los que el redondeo genere un desbordamiento o un subdesbordamiento, se seleccionará el valor más próximo del rango de entero.

El valor máximo de este umbral es de 2.147.483.400 segundos. Cualquier valor especificado que tenga un número de segundos equivalente superior a 2.147.483.400 segundos se establecerá en este número de segundos.

Es posible que los cambios en la variable de registro DB2CHECKCLIENTINTERVAL que ocasionan comprobaciones menos frecuentes afecten a la granularidad de este umbral.

SQLTEMPSPACE > *valor-entero* **(K | M | G)**
Esta condición define una vinculación superior para el tamaño de espacio de tablas temporal del sistema en cualquier partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820).

Si se especifica *valor-entero K* (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero M*, el tamaño máximo es *valor entero* multiplicado por 1 048 576. Si se especifica *valor-entero G*, el tamaño máximo es *valor entero* multiplicado por 1 073 741 824.

Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (ejecución de subsección)
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

acciones-modificar-umbral-superado
Especifica la acción que va a adoptarse cuando se supera una condición. Cada vez que se supera una condición, se registra un suceso en el supervisor de sucesos de violaciones, si hay alguno activo.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades que ha superado este umbral se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todas las particiones si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL DATABASE PARTITIONS no tiene efecto y la información sobre actividades siempre se recopila únicamente en la partición coordinadora. Para umbrales de predicción y de reacción, los detalles o valores de actividades únicamente se recopilarán en la partición coordinadora.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades que ha superado el umbral se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. La información del entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que supera el umbral.

STOP EXECUTION
La ejecución de la actividad se detiene y se devuelve un error (SQLSTATE 5U026).

CONTINUE
La ejecución de la actividad no se detiene. Cuando la condición también tiene una cola, esta opción hace que la acción de poner en cola se extienda más allá del tamaño de la cola.

ENABLE o DISABLE
Especifica si el umbral está habilitado o no para que lo utilice el gestor de la base de datos.

ENABLE
El gestor de base de datos utiliza el umbral para restringir la ejecución de actividades de base de datos. Las actividades de base de datos que están en ejecución en la actualidad seguirán ejecutándose sin las restricciones de este umbral.

DISABLE
El gestor de base de datos no utiliza el umbral para restringir la ejecución de actividades de base de datos. Este umbral no restringirá actividades de base de datos nuevas. Los umbrales con una cola, por ejemplo, TOTALSCPARTITIONCONNECTIONS o CONCURRENTDBCOORDACTIVITIES, deben inhabilitarse antes de que puedan descartarse.

Notas

- El valor nuevo para un umbral sólo afecta a las actividades de DB2 que comienzan a ejecutarse después de que se confirme la operación de modificación.
- ***Acción de umbral de CONTINUE para un umbral de puesta en cola:*** si se especifica una acción de umbral de CONTINUE para un umbral de puesta en cola, el tamaño de la cola será ilimitado, independientemente de si se ha especificado un valor máximo para el tamaño de la cola.

Ejemplo

Modifique el umbral MAXBIGQUERIESCONCURRENCY por un máximo de tres actividades en vez de dos.

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY
  WHEN CONCURRENTDBCOORDACTIVITIES > 3
  STOP EXECUTION
```

Debido a que este es un umbral con una cola, el umbral no puede descartarse a menos que se inhabilite, del siguiente modo:

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY DISABLE
```

ALTER TRUSTED CONTEXT

La sentencia ALTER TRUSTED CONTEXT modifica la definición de un contexto fiable en el servidor actual.

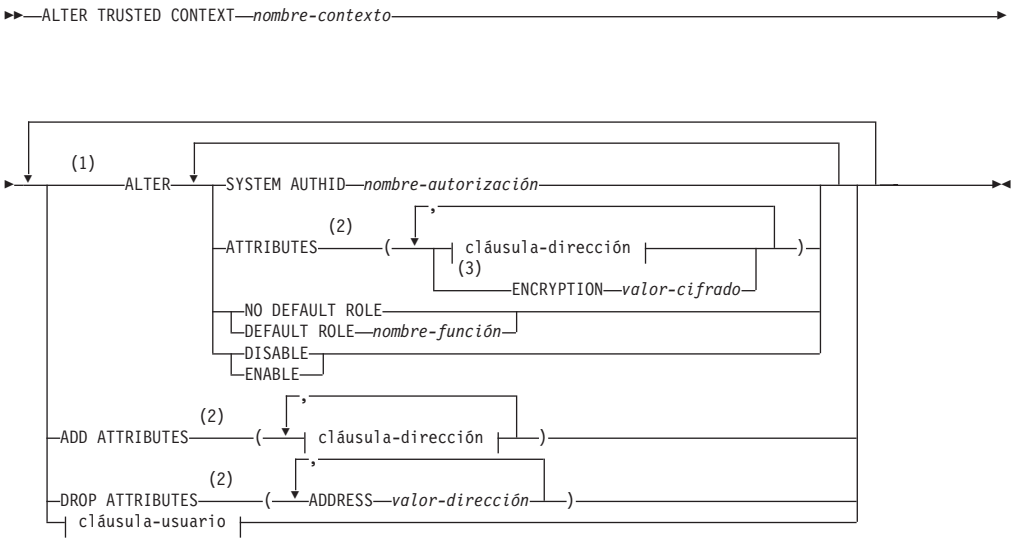
Invocación

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

Autorización

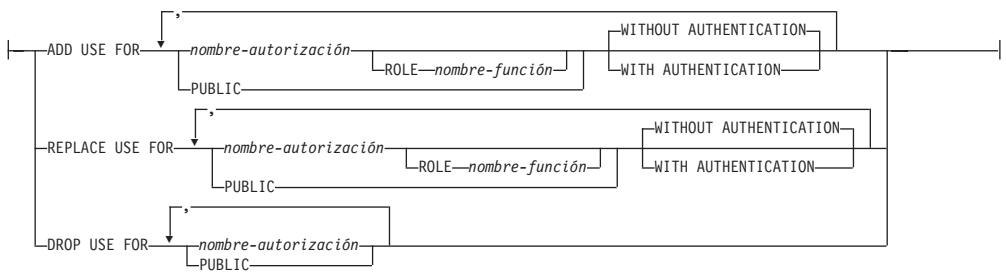
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SECADM.

Sintaxis



cláusula-dirección:

cláusula-usuario:



Notas:

1 Cada una de las cláusulas ATTRIBUTES, DEFAULT ROLE, ENABLE y WITH USE puede especificarse una vez como máximo (SQLSTATE 42614).

2 Cada nombre de atributo y su valor correspondiente deben ser exclusivos (SQLSTATE 4274D).

3 ENCRYPTION no se puede especificar más de una vez (SQLSTATE 42614); sin embargo, WITH ENCRYPTION puede especificarse para cada una de las veces que se especifique ADDRESS.

Descripción

nombre-contexto
Identifica el contexto fiable que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704).

ALTER
Modifica las opciones y atributos de un contexto fiable.

SYSTEM AUTHID *nombre-autorización*
Especifica que el contexto es una conexión establecida por medio del ID de autorización del sistema *nombre-autorización*, que no debe asociarse con un contexto fiable existente (SQLSTATE 428GL). No puede ser el ID de autorización de la sentencia (SQLSTATE 42502).

ATTRIBUTES (...)
Especifica una lista de uno o más atributos de confianza de conexión, sobre la que se define el contexto fiable, que van a modificarse. Los valores existentes para los atributos especificados se sustituyen por los valores nuevos. Si un atributo no forma parte actualmente de la definición de contexto fiable, se devuelve un error (SQLSTATE 4274C). Los atributos que no se especifiquen retendrán sus valores anteriores.

ADDRESS *valor-dirección*
Especifica la dirección de comunicación real que utiliza el cliente para comunicar con el servidor de base de datos. El único protocolo soportado es TCP/IP. Se eliminarán los valores anteriores ADDRESS para el contexto fiable especificado. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su

representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.

- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de contexto fiable si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio. Al crear un objeto de contexto fiable, resulta ventajoso proporcionar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP). Con DHCP, un dispositivo puede disponer de una dirección de IP diferente cada vez que se conecte a la red. Por tanto, si se proporciona una dirección IP estática para el atributo ADDRESS de un objeto de contexto fiable, es posible que algún dispositivo adquiera alguna conexión fiable de modo no intencional. Proporcionar nombres de dominio para el atributo ADDRESS de un objeto de contexto fiable evita este problema en entornos DHCP.

WITH ENCRYPTION *valor-cifrado*
Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser una de las siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autentificación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse el cifrado de Secure Sockets Layer (SSL) para la comunicación de datos entre el cliente de

DB2 y el servidor de DB2 en el caso de que la conexión de entrada vaya a coincidir con el valor de cifrado para esta dirección específica

ENCRYPTION *valor-cifrado*
Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red. El valor por omisión es NONE.

valor-cifrado
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser una de las siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico para una conexión de entrada que coincida con el atributo ENCRYPTION de este objeto de contexto fiable
- LOW, se necesita un cifrado ligero mínimo; el tipo de autentificación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable
- HIGH, debe utilizarse el cifrado de Secure Sockets Layer (SSL) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 en el caso de que una conexión de entrada vaya a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable.

Para obtener más detalles sobre el atributo fiable ENCRYPTION, consulte "CREATE TRUSTED CONTEXT".

NO DEFAULT ROLE o DEFAULT ROLE *nombre-función*
Especifica si se asocia o no una función por omisión con una conexión fiable basada en este contexto fiable. Si está activa una conexión fiable para este contexto, el cambio pasa a estar vigente en la siguiente petición de usuario de conmutador o una petición de conexión nueva.

NO DEFAULT ROLE
Especifica que el contexto fiable no tiene una función por omisión.

DEFAULT ROLE *nombre-función*
Especifica que *nombre-función* es la función por omisión para el contexto fiable. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). Esta función se utiliza con el usuario de una conexión fiable, basada en este contexto fiable, cuando el usuario no tenga definida una función específica de usuario como parte de la definición del contexto fiable.

ENABLE o DISABLE
Especifica si el contexto fiable está habilitado o inhabilitado.

ENABLE
Especifica que el contexto fiable está habilitado.

DISABLE
Especifica que el contexto fiable está inhabilitado. Un contexto fiable que esté inhabilitado no se tomará en consideración cuando se establezca una conexión fiable.

ADD ATTRIBUTES
Especifica una lista de uno o más atributos fiables adicionales sobre la que se define el contexto fiable.

ADDRESS *valor-dirección*
Especifica la dirección de comunicación real que utiliza el cliente para comunicar con el servidor de base de datos. El único protocolo soportado es TCP/IP. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.
- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com.

WITH ENCRYPTION *valor-cifrado*
Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser una de las siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autentificación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse el cifrado de Secure Sockets Layer (SSL) para la comunicación de datos entre el cliente de DB2

y el servidor de DB2 en el caso de que una conexión de entrada vaya a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable.

DROP ATTRIBUTES
Especifica que se van a descartar uno o más atributos de la definición del contexto fiable. Si el atributo y el par de valores de atributo no forman parte actualmente de la definición de contexto fiable, se devuelve un error (SQLSTATE 4274C).

ADDRESS *valor-dirección*
Especifica que se va a eliminar de la definición del contexto fiable la dirección de comunicaciones identificada. El *valor-dirección* especifica una constante de serie que contiene el valor de un atributo fiable de ADDRESS existente.

ADD USE FOR
Especifica usuarios adicionales que pueden utilizar una conexión fiable basada en este contexto fiable. Si la definición de un contexto fiable permite el acceso por parte de PUBLIC y una lista de usuarios, las especificaciones para un usuario alterarán temporalmente las especificaciones para PUBLIC.

nombre-autorización
Especifica que la conexión fiable puede utilizarse por medio del *nombre-autorización* especificado. El *nombre-autorización* no debe identificar un ID de autorización que ya esté definido para la utilización del contexto fiable y no debe especificarse más de una vez en la cláusula ADD USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502).

ROLE *nombre-función*
Especifica que *nombre-función* es la función que ha de utilizarse para el usuario. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). La función explícitamente especificada para el usuario altera temporalmente las funciones por omisión asociadas con el contexto fiable.

PUBLIC
Especifica que cualquier usuario puede utilizar una conexión fiable basada en este contexto fiable. PUBLIC no debe estar previamente definido para la utilización del contexto fiable y no debe especificarse más de una vez en la cláusula ADD USE FOR (SQLSTATE 428GM).

WITHOUT AUTHENTICATION o WITH AUTHENTICATION
Especifica si conmutar el usuario actual en una conexión fiable basada en este contexto fiable requiere autentificación o no.

WITHOUT AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario no requiere autentificación.

WITH AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario requiere autentificación.

REPLACE USE FOR
Especifica la forma en la que va a cambiar un usuario concreto o PUBLIC utiliza el contexto fiable.

nombre-autorización
Especifica el *nombre-autorización* del usuario cuya utilización de la conexión fiable va a cambiarse. El contexto fiable debe estar ya definido para que lo utilice el *nombre-autorización* (SQLSTATE 428GN) y el *nombre-autorización* no debe especificarse más de una vez en la cláusula REPLACE USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502).

ROLE *nombre-función*
Especifica que *nombre-función* es la función para el usuario. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). La función explícitamente especificada para el usuario altera temporalmente las funciones por omisión asociadas con el contexto fiable.

PUBLIC
Especifica que van a cambiar los atributos para utilizar la conexión fiable por medio de PUBLIC. El contexto fiable debe estar ya definido para permitir que lo utilice PUBLIC (SQLSTATE 428GN) y PUBLIC no debe especificarse más de una vez en la cláusula REPLACE USE FOR (SQLSTATE 428GM).

WITHOUT AUTHENTICATION o WITH AUTHENTICATION
Especifica si conmutar el usuario actual en una conexión fiable basada en este contexto fiable requiere autentificación o no.

WITHOUT AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario no requiere autentificación.

WITH AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable basada en este contexto fiable para este usuario requiere autentificación.

DROP USE FOR
Especifica la(s) persona(s) que ya no puede(n) utilizar el contexto fiable. Los usuarios eliminados de la definición del contexto fiable son los usuarios a los que se les permite en la actualidad utilizar el contexto fiable. En el caso de que se pueda eliminar de la definición del contexto fiable uno o más usuarios, pero no la totalidad de los mismos, se eliminarán los usuarios especificados y se devolverá un aviso (SQLSTATE 01682). En el caso de que no se pueda eliminar de la definición del contexto fiable ninguno de los usuarios especificados, se devolverá un aviso (SQLSTATE 428GN).

nombre-autorización
Elimina la posibilidad de que el ID de autorización especificado utilice este contexto fiable.

PUBLIC
Elimina la posibilidad de que todos los usuarios (excepto el ID de autorización del sistema y los ID de autorización individual que se hayan habilitado explícitamente) utilicen este contexto fiable.

Normas

- Una sentencia de SQL exclusiva del contexto fiable debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas del contexto fiable son:

  - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT o DROP (TRUSTED CONTEXT)
- Una sentencia de SQL exclusiva del contexto fiable no puede emitirse en una transacción global; por ejemplo, una transacción XA o una transacción global que se inicie como parte de una confirmación en dos fases para las transacciones federadas (SQLSTATE 51041).

Notas

- Al proporcionar una dirección IP como parte de una definición de contexto fiable, la dirección debe estar en el formato que esté vigente para la red. Por ejemplo, proporcionar una dirección en un formato IPv6 cuando la red sea IPv4 no dará como resultado una coincidencia. En un entorno mixto, resulta ventajoso especificar las representaciones IPv4 e IPv6 de la dirección, o mejor aún, especificar un nombre de dominio seguro (por ejemplo, corona.torolab.ibm.com), que oculta los detalles de formato de dirección.
- Sólo se permite una sentencia de SQL exclusiva del contexto fiable sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva del contexto fiable sin confirmar, las siguientes sentencias de SQL exclusivas del contexto fiable esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva del contexto fiable actual.
- Los cambios se graban en el catálogo del sistema pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- ***Orden de operaciones:*** El orden de las operaciones en una sentencia ALTER TRUSTED CONTEXT es:
  - DROP
  - ALTER
  - ADD ATTRIBUTES
  - ADD USE FOR
  - REPLACE USE FOR
- ***Efecto de los cambios sobre las conexiones fiables existentes:*** Si existen conexiones fiables para el contexto fiable que están modificándose, las conexiones seguirán siendo fiables con la definición vigente antes de la sentencia ALTER TRUSTED CONTEXT hasta que termine la siguiente conexión o petición de usuario de conmutador. Si se inhabilita el contexto fiable mientras están activas las conexiones fiables para este contexto, las conexiones seguirán siendo fiables >hasta que termine la siguiente conexión o petición de usuario de conmutador. Si se cambian los atributos fiables con la sentencia ALTER TRUSTED CONTEXT, se permite que continúen las conexiones fiables que existan mientras la sentencia ALTER TRUSTED CONTEXT utiliza el contexto fiable.
- ***Privilegios de función:*** Si no hay ninguna función asociada al usuario o al contexto fiable, sólo son de aplicación los privilegios asociados con el usuario. Es igual que no estar en un contexto fiable.

Ejemplos

Ejemplo 1: Suponga que existe el contexto fiable APPSERVER y que está habilitado. Emita una sentencia ALTER TRUSTED CONTEXT para permitir a Bill utilizar el contexto fiable APPSERVER, pero coloque el contexto fiable en estado de inhabilitado.

```
ALTER TRUSTED CONTEXT APPSERVER
  DISABLE
  ADD USE FOR BILL
```

Ejemplo 2: Suponga que existe el contexto fiable SECUREROLE. Emita una sentencia ALTER TRUSTED CONTEXT para modificar el usuario existente Joe para que utilice el contexto fiable con la autentificación y para añadir a todos los demás para que utilicen el contexto fiable sin autentificación.

```
ALTER TRUSTED CONTEXT SECUREROLE
  REPLACE USE FOR JOE WITH AUTHENTICATION
  ADD USE FOR PUBLIC WITHOUT AUTHENTICATION
```

Ejemplo 3: Suponga que hay un contexto fiable SECUREROLEENCRYPT con los valores de atributo ADDRESS '9.13.55.100' y '9.12.30.112' y un valor de atributo de ENCRYPTION 'NONE'. Emita una sentencia ALTER para modificar los valores de atributo ADDRESS y el atributo de cifrado a 'LOW'.

```
ALTER TRUSTED CONTEXT SECUREROLEENCRYPT
  ALTER ATTRIBUTES (ADDRESS '9.12.155.200',
    ENCRYPTION 'LOW')
```

ALTER TYPE (estructurado)

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

Autorización

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

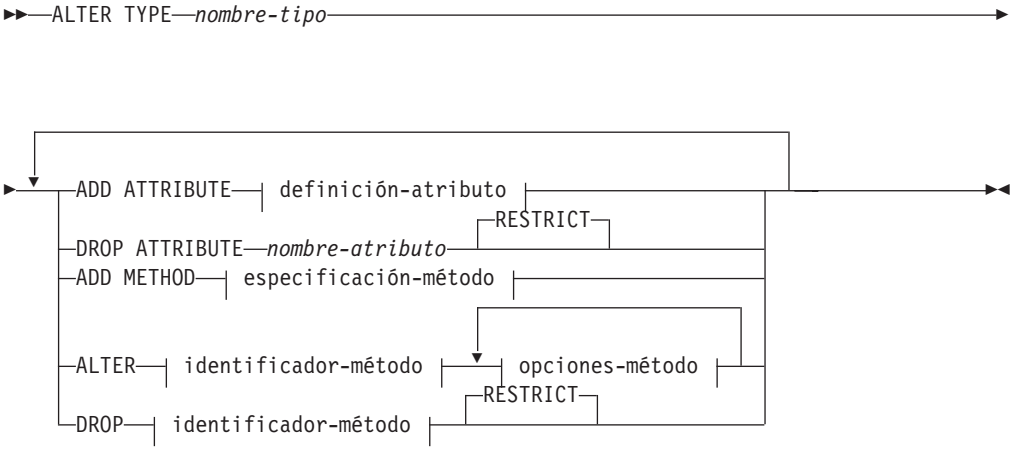
- Privilegio ALTERIN para el esquema del tipo
- Propietario del tipo, según se indica en la columna OWNER de la vista de catálogo SYSCAT.DATATYPES
- Autorización SYSADM o DBADM

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

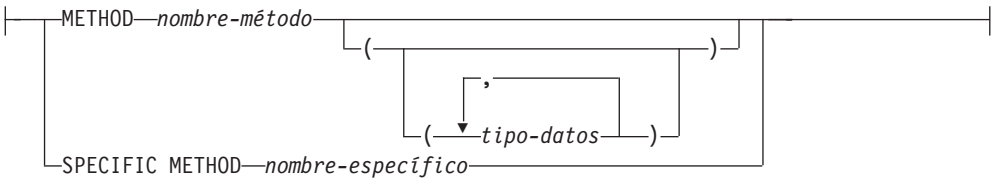
- Autorización CREATE_NOT_FENCED_ROUTINE para la base de datos
- Autorización SYSADM o DBADM

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

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 que se listan en CREATE TABLE, que no sean XML (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". Si el tipo de datos del atributo es un tipo de referencia, el tipo de destino de la referencia debe ser un tipo estructurado que exista (SQLSTATE 42704).

Para evitar las definiciones de tipo que, durante la ejecución, pueden permitir que una instancia del tipo contengan, directa o indirectamente, otra instancia del mismo tipo o uno de sus subtipos, existe una

restricción que establece que un tipo no puede definirse de forma que uno de sus tipos de atributo haga uso de sí mismo directa o indirectamente (SQLSTATE 428EP).

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

DROP ATTRIBUTE
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 desactivarse si se utiliza *nombre-tipo* como tipo de una tabla, una vista o una columna existente, de un atributo anidado en el tipo de una columna o de una extensión de índice.

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

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

identificador-método

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

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

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

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

Si un tipo de datos no está calificado, el nombre del tipo se resuelve realizando una búsqueda en los esquemas de la vía de

acceso de SQL. Esto también se aplica a los nombres de tipo de datos especificados para un tipo REFERENCE.

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

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

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

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

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

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

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

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

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

PRECAUCIÓN:
El uso de métodos NOT FENCED que no se han codificado, revisado ni probado adecuadamente puede comprometer la integridad de DB2. DB2 toma algunas precauciones ante varios de los tipos comunes de anomalías inadvertidas que puedan producirse, pero no puede garantizar la integridad completa si se utilizan métodos NOT FENCED.

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

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

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

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

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

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

DROP *identificador-método*
Identifica de forma exclusiva una instancia de un método que va a 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.
- Un tipo no puede modificarse añadiendo atributos de tal modo que el número total de atributos para el tipo, o cualquiera de sus subtipos, sea superior a 4082 (SQLSTATE 54050).
- Opción ADD ATTRIBUTE:
  - ADD ATTRIBUTE genera métodos de observador y mutador para el nuevo atributo. Estos métodos son similares a los que se generan durante la creación de un tipo estructurado (véase “CREATE TYPE (Estructurado)”). Si estos métodos entran en conflicto con otros métodos u otras funciones existentes, la sentencia ALTER TYPE da un error (SQLSTATE 42745).

  - Si el usuario ha especificado de forma explícita el valor de INLINE LENGTH para el tipo (o cualquiera de sus subtipos) con un valor inferior a 292, y los atributos añadidos dan lugar a que la longitud en línea especificada sea inferior que el tamaño del resultado de la función de construcción para el tipo modificado (32 bytes más 10 bytes por atributo), se genera un error (SQLSTATE 42611).
- Opción DROP ATTRIBUTE:
  - Un atributo que se hereda de un supertipo existente no puede desactivarse (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 propietario del tipo definido por el usuario tiene la capacidad de desactivar la especificación del método utilizando una sentencia ALTER TYPE.

Ejemplos

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

La secuencia siguiente permitiría crear los tipos y las tablas.

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

La secuencia siguiente permitiría 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. Puesto que la longitud en línea no se ha especificado en la sentencia CREATE TYPE original, DB2 calcula de nuevo la longitud en línea añadiendo 13 (10 bytes para el atributo nuevo + longitud del atributo + 2 bytes para un atributo que no es LOB).

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

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

```
ALTER TYPE EMP ...
  ADD METHOD BONUS (RATE DOUBLE)
    RETURNS INTEGER
    LANGUAGE SQL
    CONTAINS SQL
    NO EXTERNAL ACTION
    DETERMINISTIC
```

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

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

ALTER USER MAPPING

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

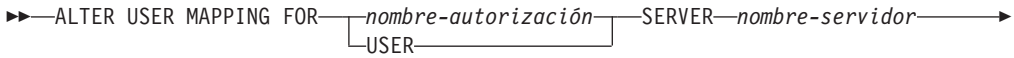
Invocación

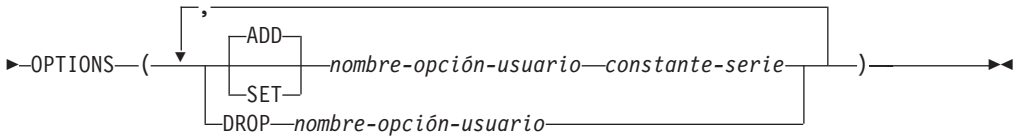
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de autorización correlacionado con la fuente de datos, los privilegios del ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o 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.

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

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

ALTER VIEW

La sentencia ALTER VIEW modifica una vista existente:

- Alterando una columna de tipo de referencia para añadir un ámbito
- Habilitando o inhabilitando una vista para utilizarla en la optimización de la consulta

Invocación

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

Autorización

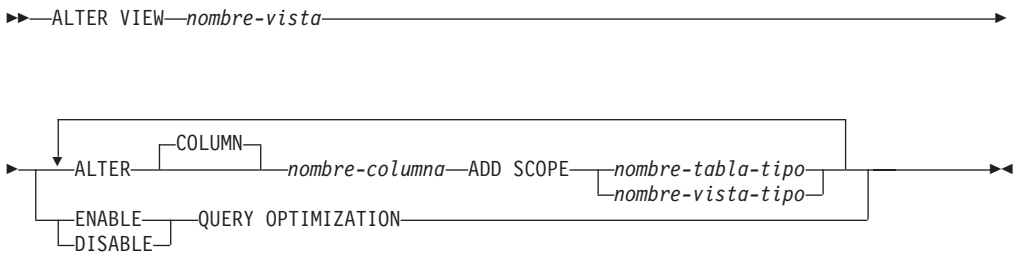
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio ALTERIN para el esquema de la vista
- Propietario de la vista que se debe modificar
- Privilegio CONTROL para la vista a modificar
- Autorización SYSADM o DBADM

Para habilitar o inhabilitar una vista para utilizarla en la optimización de la consulta, los privilegios de ID de autorización de la sentencia también deben incluir como mínimo uno de los privilegios siguientes para cada una de las tablas o de las tablas subyacentes de vistas a las que se hace referencia en la cláusula FROM de la selección completa de la vista:

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

Sintaxis



Descripción

nombre-vista
Especifica la vista que debe cambiarse. Debe ser una vista que esté descrita en el catálogo.

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

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

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

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

ENABLE QUERY OPTIMIZATION o DISABLE QUERY OPTIMIZATION
Especifica si la vista y las estadísticas asociadas se deben utilizar para mejorar la optimización de las consultas. DISABLE QUERY OPTIMIZATION es el valor por omisión cuando se crea una vista.

ENABLE QUERY OPTIMIZATION
Especifica que la vista incluye estadísticas que pueden utilizarse para mejorar la optimización de las consultas que implican esta vista o las consultas que incluyen subconsultas parecidas a la selección completa de esta vista.

DISABLE QUERY OPTIMIZATION
Especifica que la vista y las estadísticas asociadas no deben utilizarse para mejorar la optimización de las consultas.

Normas

- Una vista no puede habilitarse para la optimización de una consulta si:
  - La vista hace referencia directa o indirectamente a una tabla de consultas materializadas (MQT). Debe tenerse en cuenta que una MQT o una vista estadística puede hacer referencia a una vista estadística.
  - Es una vista con tipo

Notas

- Para que puede considerarse para la optimización de una consulta, una vista:
  - No puede contener operaciones de agregación ni diferenciación
  - No puede contener operaciones de unión, excepción ni intersección
  - No puede contener funciones de agregación escalar (OLAP)
- Si una vista se modifica para inhabilitar la optimización de consultas, los planes de consultas en antememoria que utilizaban la vista para la optimización de consultas se invalidarán. Si una vista se modifica para habilitar la optimización de consultas, los planes de consulta en antememoria se invalidarán si hacen referencia a las mismas tablas que las nuevas referencias de vista habilitadas, ya sea directa o indirectamente a través de otras vistas. La invalidación de estos planes de consulta en antememoria da lugar a una revalidación implícita que tiene en cuenta la propiedad de optimización de consulta modificada de la vista.

  La propiedad de optimización de consulta para una vista no tiene impacto alguno en las sentencias de SQL integradas estáticas.

ALTER WORK ACTION SET

La sentencia ALTER WORK ACTION SET modifica un conjunto de acciones de trabajo añadiendo, modificando o descartando acciones de trabajo en el conjunto de acciones de trabajo.

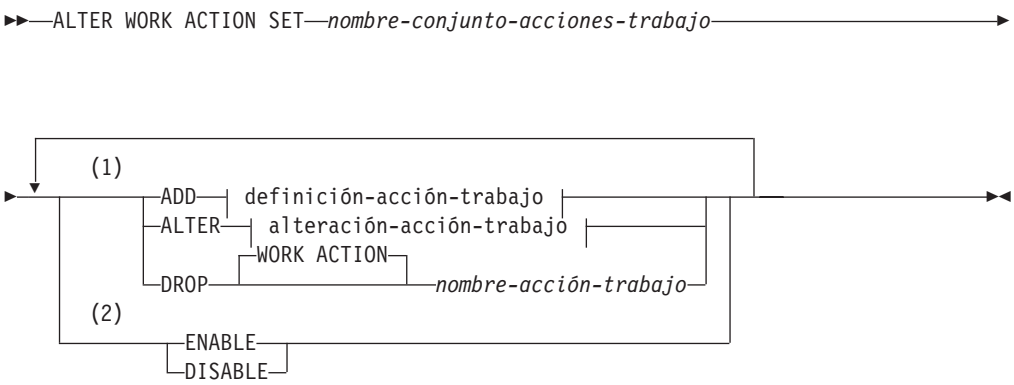
Invocación

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

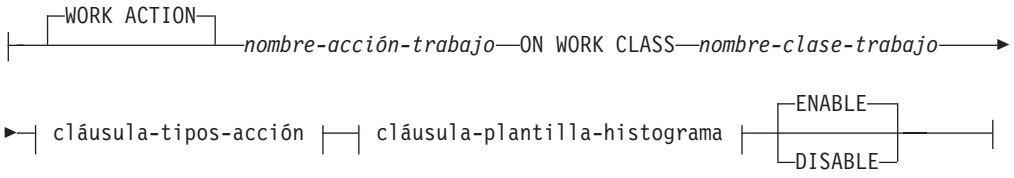
Autorización

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

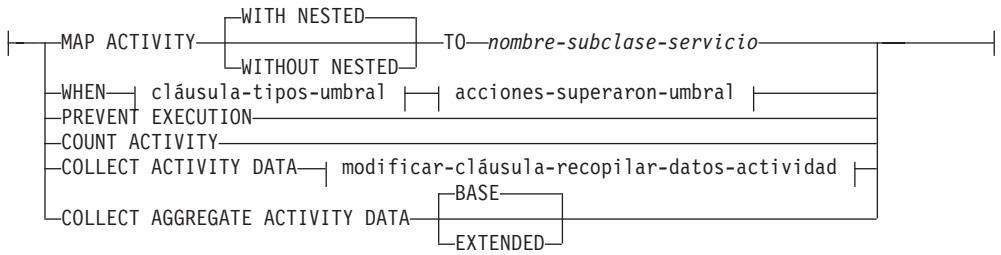
Sintaxis



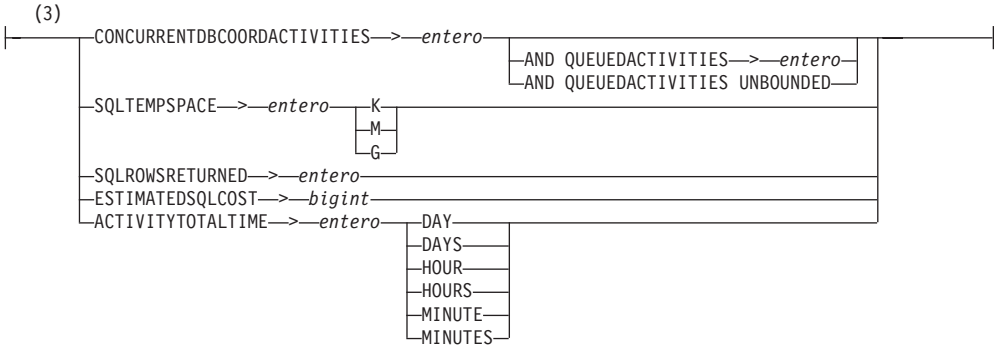
definición-acción-trabajo:



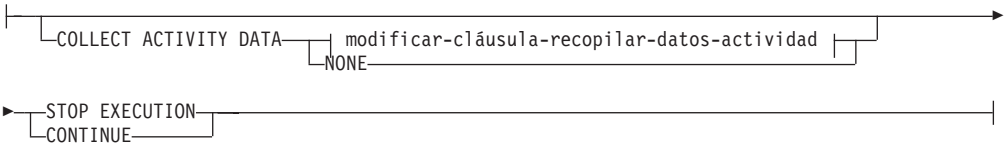
cláusula-tipos-acción:



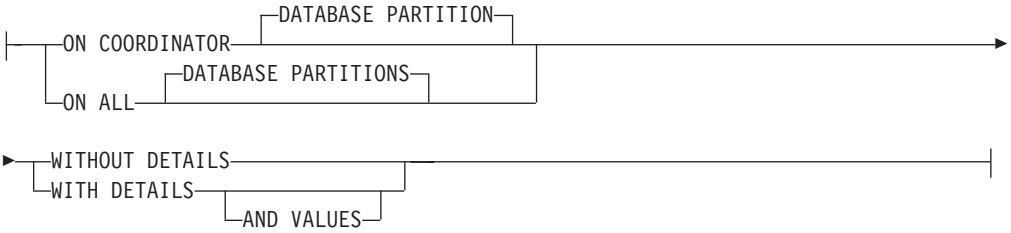
cláusula-tipos-umbral:



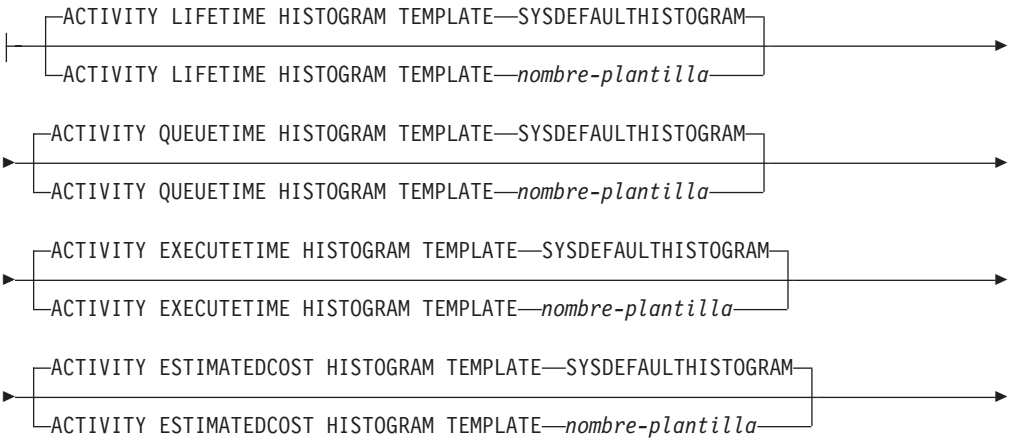
acciones-superaron-umbral:

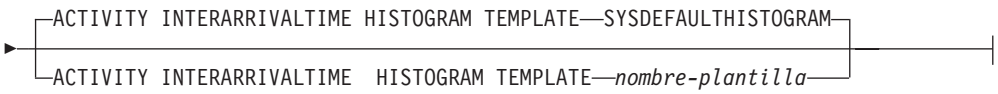


modificar-cláusula-recopilar-datos-actividad:

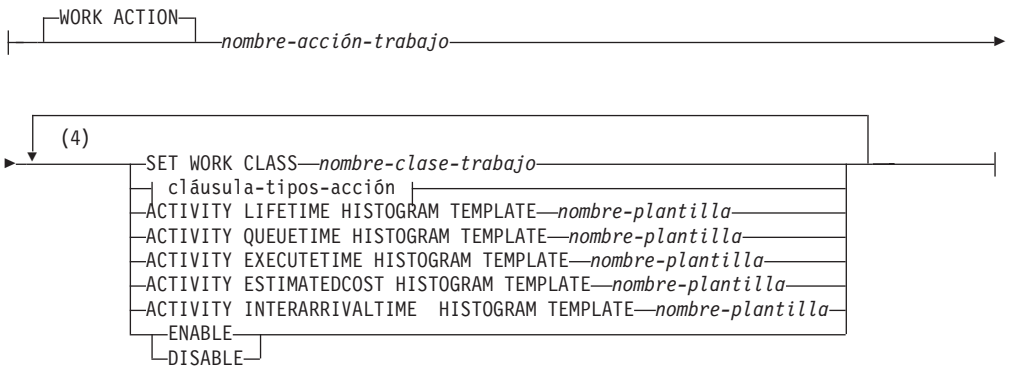


cláusula-plantilla-histograma:





alteración-acción-trabajo:



Notas:

1 Las cláusulas ADD, ALTER y DROP se procesan en el orden en que se especifican.

2 La cláusula ENABLE o DISABLE sólo puede especificarse una vez en la misma sentencia.

3 Sólo puede aplicarse una acción de trabajo del mismo tipo de umbral a una única clase de trabajo a la vez. Cuando se modifica una acción de trabajo de umbral, no se puede cambiar el predicado de umbral.

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

Descripción

nombre-conjunto-acciones-trabajo
Identifica el conjunto de acciones de trabajo que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704).

ADD
Añade una acción de trabajo al conjunto de acciones de trabajo.

WORK ACTION *nombre-acción-trabajo*
Da nombre a la acción de trabajo. El *nombre-acción-trabajo* no debe identificar una acción de trabajo que ya exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42710). El *nombre-acción-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

ON WORK CLASS *nombre-clase-trabajo*
Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY
Especifica una acción de trabajo de correlación de la actividad. Esta acción

sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED
Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED
Todas las actividades de base de datos que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de datos anidadas bajo esta actividad, se correlacionarán a la subclase de servicio; es decir, las actividades con un nivel de anidamiento superior a cero se ejecutarán en la misma clase de servicio que las actividades con un nivel de anidamiento de cero.

WITHOUT NESTED
Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO *nombre-subclase-servicio*
Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN
Especifica el umbral que se aplicará a la actividad de base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo. Un umbral sólo puede especificarse si el objeto de gestor de base de datos para el que se ha definido esta acción de trabajo está definido como base de datos (SQLSTATE 5U034). Ninguno de estos umbrales se aplican a las actividades de base de datos internas iniciadas por medio del gestor de base de datos o a las actividades de base de datos generadas por medio de rutinas SQL administrativas.

cláusula-tipos-umbral
Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia “CREATE THRESHOLD”.

acciones-superaron-umbral
Para obtener una descripción de las acciones de excedido umbral válidas, consulte la sentencia “CREATE THRESHOLD”.

PREVENT EXECUTION
Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY
Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades asociadas con la

clase de trabajo para la que se define esta acción de trabajo van a enviarse al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todas las particiones si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL DATABASE PARTITIONS no tiene efecto y la información sobre actividades siempre se recopila únicamente en la partición coordinadora. Para umbrales de predicción y de reacción, los detalles o valores de actividades únicamente se recopilarán en la partición coordinadora.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. La información del entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

NONE
Especifica que los datos de actividad no se deben recopilar para cada actividad que esté asociada con la clase de trabajo para la que se define esta acción de trabajo.

COLLECT AGGREGATE ACTIVITY DATA
Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE
Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la

que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo de cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED
Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

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

ENABLE o DISABLE
Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

ENABLE
Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE
Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometan actividades de base de datos.

cláusula-plantilla-histograma
Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades asociadas a la clase de trabajo a la que se asigna esta acción de trabajo. Los datos de actividades agregadas sólo se recopilan para la clase de trabajo cuando el tipo de acción de trabajo es COLLECT AGGREGATE ACTIVITY DATA.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos de las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta

información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada una de las particiones de base de datos en las que se ejecuta la actividad. En la partición de base de datos del coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En particiones de base de datos no de coordinador, este es el tiempo que estas particiones pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en una partición de base de datos remota más de una vez y cada una de las veces la partición remota recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en una partición de base de datos. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE
nombre-plantilla
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE
nombre-plantilla
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ALTER
Modifica la definición de la acción de trabajo. Puede cambiar la clase de trabajo a la que se aplica esta acción de trabajo y la acción que va a aplicarse a la actividad de base de trabajo que caiga dentro de la clase de trabajo.

WORK ACTION *nombre-acción-trabajo*
Identifica la acción de trabajo. El *nombre-acción-trabajo* debe identificar una acción de trabajo que exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42704).

SET WORK CLASS *nombre-clase-trabajo*
Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY
Especifica una acción de trabajo de correlación de la actividad. Esta acción sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED
Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED
Todas las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de trabajo anidadas bajo esta actividad se correlacionarán con la subclase de servicio.

WITHOUT NESTED
Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO *nombre-subclase-servicio*
Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN
Especifica el umbral que se aplicará a la actividad de base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo. Un umbral sólo puede especificarse si el objeto de gestor de base de datos para el que se ha definido esta acción de trabajo está definido como base de datos (SQLSTATE 5U034). Ninguno de estos umbrales se aplican a las actividades de base de datos internas iniciadas por medio del gestor de base de datos o a las actividades de base de datos generadas por medio de rutinas SQL administrativas.

cláusula-tipos-umbral
Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia “CREATE THRESHOLD”.

acciones-superaron-umbral
Para obtener una descripción de las acciones de excedido umbral válidas, consulte la sentencia “CREATE THRESHOLD”.

PREVENT EXECUTION
Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY
Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo van a enviarse al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todas las particiones si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL DATABASE PARTITIONS no tiene efecto y la información sobre actividades siempre se recopila únicamente en la partición coordinadora. Para umbrales de predicción y de reacción, los detalles o valores de actividades únicamente se recopilarán en la partición coordinadora.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. La información del entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

NONE
Especifica que los datos de actividad no se deben recopilar para cada actividad que esté asociada con la clase de trabajo para la que se define esta acción de trabajo.

COLLECT AGGREGATE ACTIVITY DATA
Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT

AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE
Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo de cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED
Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

- Histograma de coste estimado de DML de la actividad
- Histograma de tiempo de llegada de DML de actividad

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos de las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada una de las particiones de base de datos en las que se ejecuta la actividad. En la partición de base de datos del coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En particiones de base de datos no de coordinador, este es el

tiempo que estas particiones pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en una partición de base de datos remota más de una vez y cada una de las veces la partición remota recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en una partición de base de datos. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades de lenguaje de manipulación de datos (DML) asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ENABLE o DISABLE
Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos.

ENABLE
Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE
Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometan actividades de base de datos.

DROP *nombre-acción-trabajo*
Descarta una acción de trabajo del conjunto de acciones de trabajo. El *nombre-acción-trabajo* debe identificar una acción de trabajo que exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42704).

ENABLE o DISABLE
Especifica si el conjunto de acciones de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos.

ENABLE
Especifica que el conjunto de acciones de trabajo está habilitado y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE
Especifica que el conjunto de acciones de trabajo está inhabilitado y no va a tomarse en consideración cuando se sometan actividades de base de datos.

Normas

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

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Modifique el conjunto de acciones de trabajo DATABASE_ACTIONS y añada dos acciones de trabajo utilizando la clase de trabajo LARGE_SELECTS. Para la acción de trabajo ONE_CONCURRENT_SELECT, aplique un umbral de simultaneidad de 1 para controlar el número de actividades que puede ejecutarse a la vez y permitir que como máximo haya 3 en cola. Para la acción de trabajo BIG_ROWS_RETURNED, limite el número de filas que puedan devolver las actividades de base de trabajo que caigan en dicha clase a 1 000 000.

```
ALTER WORK ACTION SET DATABASE_ACTIONS
  ADD WORK ACTION ONE_CONCURRENT_SELECT ON WORK CLASS LARGE_SELECTS
     WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 3 STOP EXECUTION
     ADD WORK ACTION BIG_ROWS_RETURNED ON WORK CLASS LARGE_SELECTS
     WHEN SQLROWSRETURNED > 1000000 STOP EXECUTION
```

Ejemplo 2: Modifique el conjunto de acciones de trabajo ADMIN_APPS_ACTIONS al objeto de modificar la acción de trabajo MAP_SELECTS para correlacionar todas las actividades que se ejecuten en la clase de super servicio ADMIN_APPS bajo la clase de trabajo SELECT_CLASS con la subclase de servicio ALL_SELECTS. Añada asimismo una acción de trabajo nueva denominada MAP_UPDATES que correlacione todas las actividades que se ejecuten en la clase de trabajo UPDATE_CLASS con la subclase de servicio ALL_SELECTS.

```
ALTER WORK ACTION SET ADMIN_APPS_ACTIONS
  ALTER WORK ACTION MAP_SELECTS MAP ACTIVITY TO ALL_SELECTS
  ADD WORK ACTION MAP_UPDATES ON WORK CLASS UPDATE_CLASS
    MAP ACTIVITY TO ALL_SELECTS
```

ALTER WORK CLASS SET

La sentencia ALTER WORK CLASS SET añade, modifica o descarta clases de trabajo dentro de un conjunto de clases de trabajo.

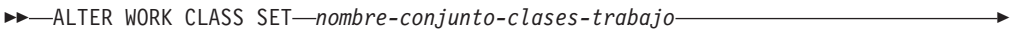
Invocación

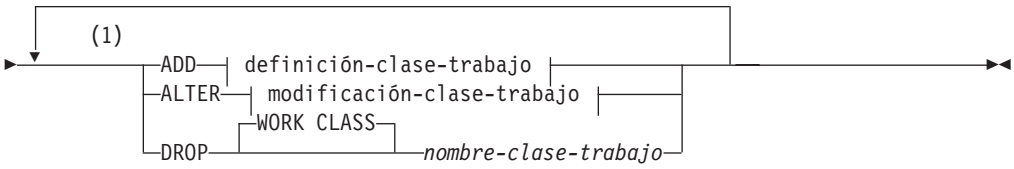
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

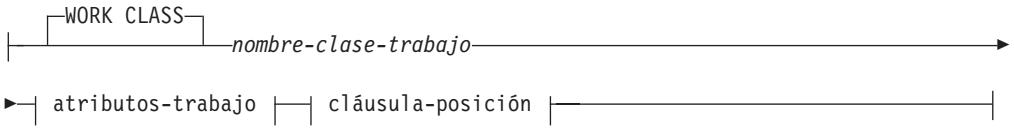
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM.

Sintaxis

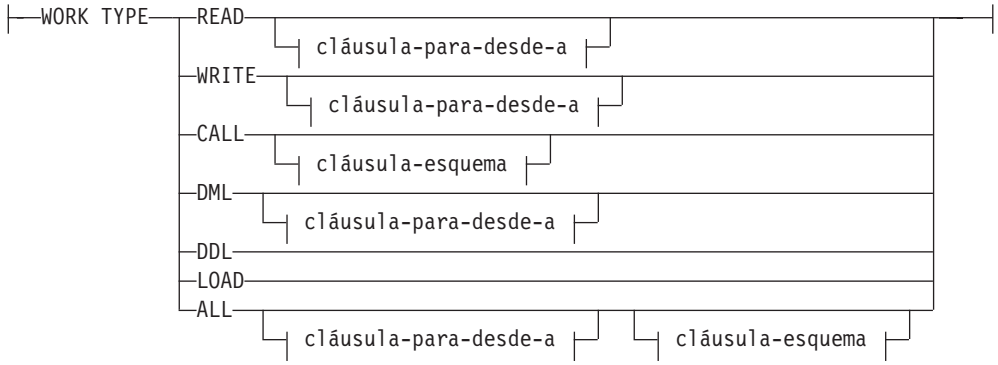




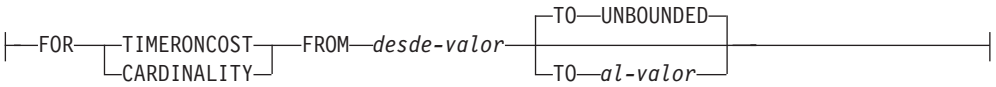
definición-clase-trabajo:



atributos-trabajo:

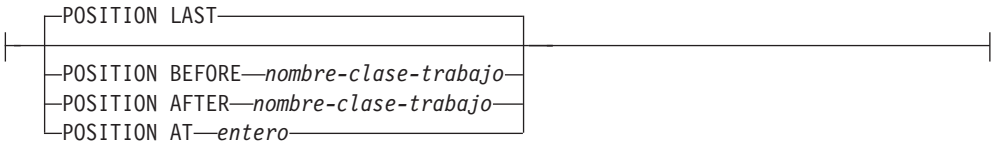


cláusula-para-desde-a:

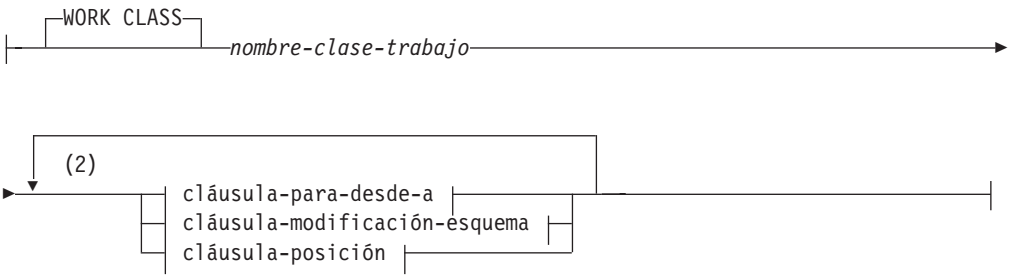


cláusula-esquema:

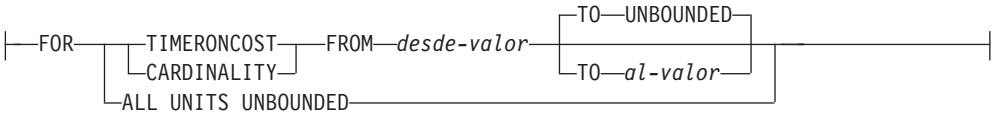
cláusula-posición:



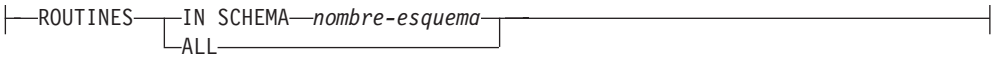
modificación-clase-trabajo:



cláusula-para-desde-a:



cláusula-modificación-esquema:



Notas:

1 Las cláusulas ADD, ALTER y DROP se procesan en el orden en que se especifican.

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

Descripción

nombre-conjunto-clases-trabajo
Identifica el conjunto de clases de trabajo que se va a modificar. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o

delimitado). El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704).

ADD
Añade una clase de trabajo al conjunto de clases de trabajo. Para obtener detalles, consulte "CREATE WORK CLASS SET".

ALTER
Modifica los atributos de actividad de base de datos y la posición de una clase de trabajo específica en el conjunto de clases de trabajo.

WORK CLASS *nombre-clase-trabajo*
Identifica la clase de trabajo se va a modificar. El *nombre-clase-trabajo* debe identificar una clase de trabajo que exista en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42704).

DROP
Descarta la clase de trabajo del conjunto de clases de trabajo.

WORK CLASS *nombre-clase-trabajo*
Identifica la clase de trabajo se va a descartar. El *nombre-clase-trabajo* debe identificar una clase de trabajo que exista en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42704). Una clase de trabajo no se puede descartar si hay una acción de trabajo en cualquiera de los conjuntos de acciones de trabajo asociados con este conjunto de clases de trabajo que dependa de la misma (SQLSTATE 42893).

cláusula-para-desde-a

FOR
Indica el tipo de información que está especificándose en la cláusula FROM *desde-valor* TO *al-valor*. La cláusula FOR sólo se utiliza para los siguientes tipos de trabajo:

- READ
- WRITE
- DML
- ALL

TIMERONCOST
El coste estimado del trabajo, en acciones de temporización. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

CARDINALITY
La cardinalidad estimada del trabajo. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

FROM *desde-valor* **TO UNBOUNDED o FROM** *desde-valor* **TO** *al-valor*
Especifica el rango de valor de activación de temporizador (para el coste estimado) o cardinalidad en la que debe caer la actividad de la base de datos si va a formar parte de esta clase de trabajo. El rango incluye *desde-valor* y *al-valor*. Este rango sólo se utiliza para los siguientes tipos de trabajo:

- READ
- WRITE
- DML
- ALL

FROM *desde-valor* **TO UNBOUNDED**
desde-valor debe ser cero o un valor DOUBLE positivo (SQLSTATE 5U019). El rango no tiene vinculación superior.

FROM *desde-valor* **TO** *al-valor*
desde-valor debe ser cero o un valor DOUBLE positivo y *al-valor* debe ser un valor DOUBLE positivo. *desde-valor* debe ser igual o más pequeño que *al-valor* (SQLSTATE 5U019).

ALL UNITS UNBOUNDED
Indica que no ha de especificarse ningún rango en la cláusula FROM *desde-valor* TO *al-valor* y que ha de incluirse todo el trabajo caiga dentro del tipo de trabajo especificado.

cláusula-modificación-esquema

ROUTINES
Esta cláusula sólo se utiliza si el tipo de trabajo es CALL o ALL y la actividad de base de datos es una sentencia CALL.

IN SCHEMA *nombre-esquema*
Especifica el nombre de esquema del procedimiento que estará llamando la sentencia CALL.

ALL
Especifica que se incluyan todos los esquemas.

cláusula-posición

POSITION
Especifica el lugar en el que va a colocarse esta clase de trabajo dentro del conjunto de clases de trabajo, lo cual determina el orden en el que se evalúan estas clases de trabajo. Al efectuar la asignación de clases de trabajo en tiempo de ejecución, el gestor de base de datos determina en primer lugar el conjunto de clases de trabajo asociado al objeto, la base de datos o una superclase de servicio. Se selecciona la primera clase de trabajo coincidente en dicho conjunto de clases de trabajo. Si no se especifica esta palabra clave, la clase de trabajo se ubica en la última posición.

LAST
Especifica que la clase de trabajo va a colocarse en última posición en la lista ordenada de las clases de trabajo del conjunto de clases de trabajo.

BEFORE *nombre-clase-trabajo*
Especifica que la clase de trabajo va a colocarse antes de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AFTER *nombre-clase-trabajo*
Especifica que la clase de trabajo va a colocarse detrás de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AT *posición*
Especifica la posición absoluta en la que la carga de trabajo va a colocarse en el conjunto de clases de trabajo en la lista ordenada de clases de trabajo. Este valor puede ser cualquier valor entero positivo (SQLSTATE 42615). Si *posición* es mayor que el número de

clases de trabajo existentes más una, la clase de trabajo se sitúa en la última posición del conjunto de clases de trabajo.

Normas

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

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Modifique el conjunto de clases de trabajo LARGE_QUERIES y establezca que las dos clases de trabajo existentes tengan un rango que comience en 100 000, conservando el rango sin vincular. Añada una tercera clase de trabajo para todas las sentencias SELECT que tengan un coste de actividades de temporización estimado que sea igual o superior a 10 000 y sitúe esta clase de trabajo para que tenga prioridad sobre las dos clases de trabajo existentes.

```
ALTER WORK CLASS SET LARGE_QUERIES
  ALTER WORK CLASS LARGE_ESTIMATED_COST
   FOR TIMERONCOST FROM 100000 TO UNBOUNDED
  ALTER WORK CLASS LARGE_CARDINALITY
   FOR CARDINALITY FROM 100000 TO UNBOUNDED
  ADD WORK CLASS LARGE_SELECTS WORK TYPE READ
   FOR TIMERONCOST FROM 10000 TO UNBOUNDED POSITION AT 1
```

Ejemplo 2: Modifique un conjunto de clases de trabajo denominado DML_STATEMENTS para añadir una clase de trabajo que represente todas las sentencias DML SELECT que contengan una sentencia DELETE, INSERT, MERGE o UPDATE.

```
ALTER WORK CLASS SET DML_STATEMENTS
  ADD WORK CLASS UPDATE_CLASS WORK TYPE WRITE
```

ALTER WORKLOAD

La sentencia ALTER WORKLOAD modifica una carga de trabajo.

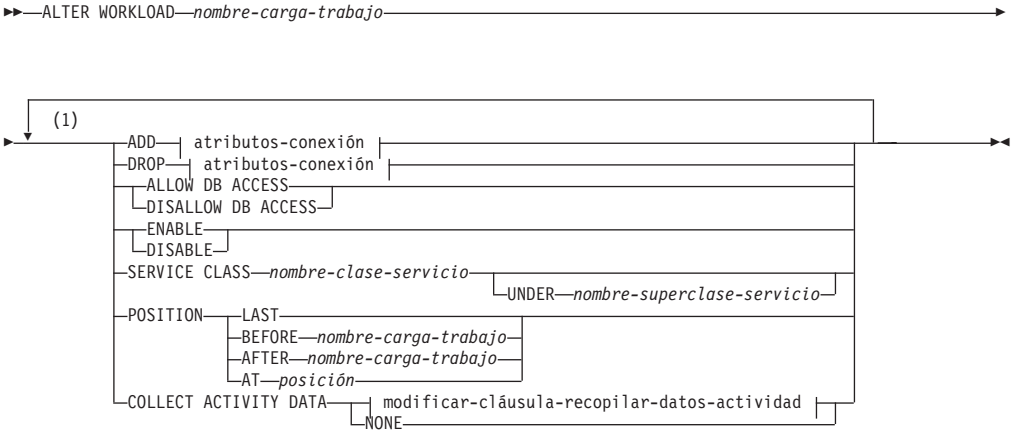
Invocación

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

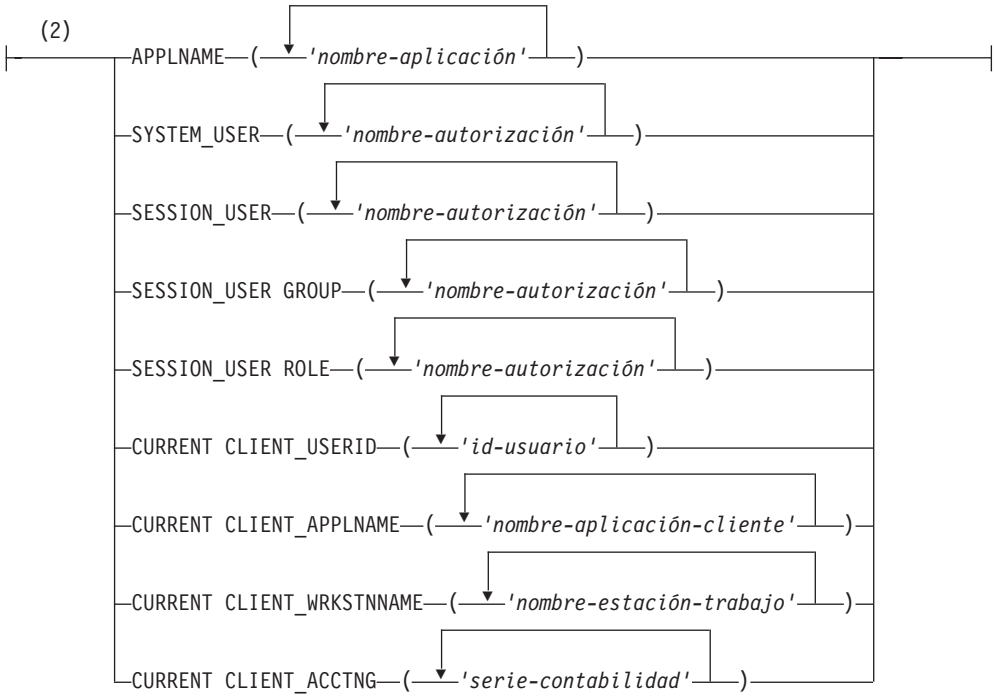
Autorización

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

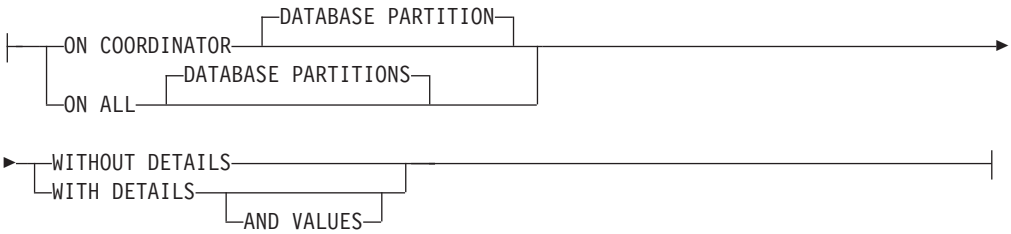
Sintaxis



atributos-conexión:



modificar-cláusula-recopilar-datos-actividad:



Notas:

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

2 Cada cláusula de atributo de conexión sólo puede especificarse una vez.

Descripción

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

ADD *atributos-conexión*
Añade uno o más valores de atributo de conexión a la definición de la carga de trabajo. Los valores de atributo de conexión especificados ya no deben estar definidos para la carga de trabajo (SQLSTATE 5U039). La opción ADD no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

DROP *atributos-conexión*
Descarta uno o más valores de atributo de conexión de la definición de la

carga de trabajo. Los valores de atributo de conexión especificados deben estar definidos para la carga de trabajo (SQLSTATE 5U040). La opción DROP no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832). Debe haber como mínimo un valor de atributo de conexión definida. No puede descartarse el último valor de atributo de conexión (SQLSTATE 5U022).

atributos-conexión
Especifica los valores de atributo de conexión para la carga de trabajo.

Nota: Todos los atributos de conexión son sensibles a las mayúsculas y minúsculas.

APPLNAME ('*nombre-aplicación***', ...)**
Especifica una o más aplicaciones para el atributo de conexión de APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SYSTEM_USER ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SYSTEM USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER GROUP ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER GROUP. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ROLE ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER ROLE. Los roles de un ID de autorización de sesión en este contexto hacen referencia a todas las funciones a disposición del ID de autorización de sesión, sin tener en cuenta el modo en que se obtuvieron las funciones. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_USERID ('*id-usuario***', ...)**
Especifica uno o más ID de usuario cliente para el atributo de conexión CURRENT CLIENT_USERID. Un ID de usuario cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_APPLNAME ('*nombre-aplicación-cliente***', ...)**
Especifica una o más aplicaciones para el atributo de conexión de CURRENT CLIENT_APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_WRKSTNNAME ('*nombre-estación-trabajo***', ...)**
Especifica uno o más nombres de estación de trabajo cliente para el atributo de conexión CURRENT CLIENT_WRKSTNNAME. Un nombre de estación de trabajo cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_ACCTNG ('*serie-contabilidad***', ...)**
Especifica uno o más series de contabilidad cliente para el atributo de conexión CURRENT CLIENT_ACCTNG. Una serie de contabilidad cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

ALLOW DB ACCESS o DISALLOW DB ACCESS
Especifica si se permitirá o no la aparición de carga de trabajo asociada a esta carga de trabajo acceder a la base de datos.

ALLOW DB ACCESS
Especifica que se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos.

DISALLOW DB ACCESS
Especifica que no se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos. La siguiente unidad de trabajo asociada a esta carga de trabajo se rechazará (SQLSTATE 5U020). Se permite que se completen las apariciones de carga de trabajo que ya se estén ejecutando. Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

ENABLE o DISABLE
Especifica si esta carga de trabajo se tomará o no en consideración cuando se seleccione una carga de trabajo.

ENABLE
Especifica que la carga de trabajo se habilitará y se tomará en consideración cuando se seleccione una carga de trabajo.

DISABLE
Especifica que la carga de trabajo se inhabilitará y no se tomará en consideración cuando se seleccione una carga de trabajo. Esta opción no puede especificarse si *nombre-carga-trabajo* es SYSDEFAULTUSERWORKLOAD o SYSDEFAULTADMWORKLOAD (SQLSTATE 42832).

SERVICE CLASS *nombre-clase-servicio*
Especifica que las peticiones asociadas a esta carga de trabajo se ejecuten en la clase de servicio *nombre-clase-servicio*. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-clase-servicio* no puede ser 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032). Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

UNDER *nombre-superclase-servicio*
La cláusula se utiliza al especificar una subclase de servicio. El *nombre-superclase-servicio* identifica la superclase de servicio de *nombre-superclase-servicio*. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no puede ser 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032).

POSITION
Especifica el lugar en el que va a situarse esta carga de trabajo en la lista ordenada de cargas de trabajo. En tiempo de ejecución, se examina esta lista por orden para buscar la primera carga de trabajo que coincida con los atributos de conexión necesarios. Esta opción no puede especificarse si *nombre-carga-trabajo* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

LAST
Especifica que la carga de trabajo va a ser la última de la lista, antes de las cargas de trabajo por omisión SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

BEFORE *nombre-carga-trabajo-relativa*
Especifica que la carga de trabajo va a colocarse antes de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción BEFORE no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AFTER *nombre-carga-trabajo-relativa*
Especifica que la carga de trabajo va a colocarse detrás de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción AFTER no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AT *posición*
Especifica la posición absoluta en la que la carga de trabajo va a colocarse en la lista. Este valor puede ser cualquier valor entero positivo (SQLSTATE 42615). Si *posición* es mayor que el número de cargas de trabajo existentes más una, la carga de trabajo se sitúa en la última posición justo antes de SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades asociadas con esta carga de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza.

modificar-cláusula-recopilar-datos-actividad

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en los que se procesa la actividad; sin embargo los valores o detalles de la actividad sólo se recopilarán en la partición de la base de datos del coordinador de la actividad.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades asociadas con esta carga de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza su ejecución. El entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que esté asociada con esta carga de trabajo.

Normas

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

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia. Para las apariciones de carga de trabajo recién sometidas, los cambios surtirán efecto después de las confirmaciones de sentencia ALTER WORKLOAD. Para las apariciones de carga de trabajo activas, los cambios surtirán efecto al principio de la siguiente unidad de trabajo.
- Si se especifica la opción DISABLE, la carga de de trabajo se inhabilitará después de las confirmaciones de sentencia. La carga de trabajo no se tendrá en cuenta la próxima vez que se seleccione una carga de trabajo. Si hay una aparición de carga de trabajo activa asociada a esta carga de trabajo cuando se confirme la sentencia ALTER WORKLOAD, ésta seguirá ejecutándose hasta el final de la unidad de trabajo actual. Al principio de la siguiente unidad de trabajo, tendrá lugar una reevaluación de carga de trabajo y la conexión pasará a estar asociada a una carga de trabajo diferente.

Ejemplos

Ejemplo 1: La carga de trabajo PAYROLL se ha colocado en la actualidad de modo que la carga de trabajo INVENTORY se tome en consideración en primer lugar cuando DB2 seleccione una carga de trabajo en tiempo de ejecución. Modifique el orden de evaluación para que PAYROLL se tome en consideración en primer lugar.

```
ALTER WORKLOAD PAYROLL
  POSITION BEFORE INVENTORY
```

Ejemplo 2: Modifique el orden de evaluación de modo que la carga de trabajo BENCHMARK se evalúe por medio de DB2 antes de cualquier carga de trabajo del catálogo.

```
ALTER WORKLOAD BENCHMARK
  POSITION AT 1
```

Ejemplo 3: Se ha creado la carga de trabajo REPORTS con APPLNAME establecido en appl1, appl2 y appl3 y SYSTEM_USER establecido en BOB y MARY. Modifique la carga de trabajo para añadir una aplicación nueva, appl4 a la lista de nombres de aplicación y elimine appl2, ya que no debería correlacionarse con REPORTS.

```
ALTER WORKLOAD REPORTS
  ADD APPLNAME ('appl4')
  DROP APPLNAME ('appl2')
```

ALTER WRAPPER

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

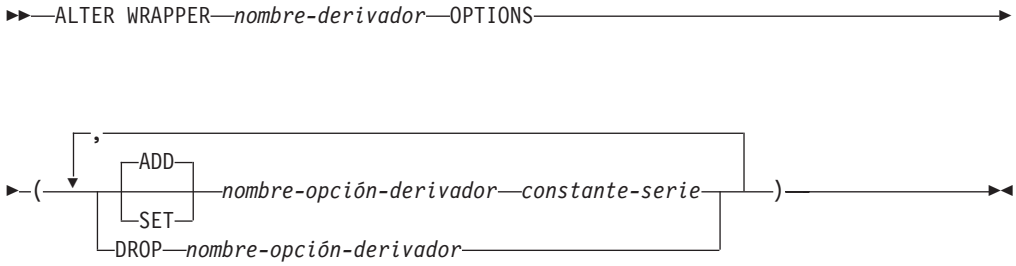
Invocación

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

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

Sintaxis



Descripción

nombre-derivador
Especifica el nombre del derivador.

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

ADD
Habilita una opción de servidor.

SET
Cambia el valor de una opción de derivador.

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

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

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

Notas

- La ejecución de la sentencia ALTER WRAPPER no incluye la comprobación de validez de las opciones específicas de derivador.
- Una sentencia ALTER WRAPPER de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) si la UOW ya incluye uno de los siguientes elementos:
  - Una sentencia SELECT que hace referencia a un apodo que pertenece al derivador.
  - Un cursor abierto en un apodo que pertenece al derivador.
  - Una sentencia INSERT, DELETE o UPDATE emitida para un apodo que pertenece al derivador.

Ejemplos

Ejemplo 1: Active la opción DB2_FENCED para el derivador NET8.

```
ALTER WRAPPER NET8 OPTIONS (SET DB2_FENCED 'Y')
```

ALTER XSROBJECT

Esta sentencia se utiliza para habilitar o inhabilitar el soporte de la descomposición para un esquema XML específico. Los esquemas XML anotados se pueden utilizar para descomponer documentos XML en tablas relacionales, si se ha habilitado la descomposición para dichos esquemas XML.

Invocación

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

Autorización

Se necesita una de las autorizaciones siguientes:

- SYSADM o DBADM
- ALTERIN en el esquema de SQL
- Propiedad del objeto XSR que se va a modificar

Sintaxis

Descripción

nombre-objetoxsr
Identifica el objeto XSR que se va a modificar. El *nombre-objetoxsr* (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este identificador, se devolverá un error (SQLSTATE 42704).

ENABLE DECOMPOSITION o DISABLE DECOMPOSITION
Habilita o inhabilita el uso del objeto XSR para la descomposición. El objeto XSR identificado debe ser un esquema XML (SQLSTATE 42809). Para habilitar la descomposición, es preciso anotar el esquema XML con normas de descomposición (SQLSTATE 225DE) y los objetos a los que dichas normas hacen referencia deben existir en el servidor actual (SQLSTATE 42704).

Notas:

- Cuando se inhabilita la descomposición de un objeto XSR, se eliminan todas las entradas de catálogo relacionadas.
- El soporte de descomposición de un objeto XSR se inhabilitará si se elimina o modifica algún objeto del que dependa el objeto XRS (como por ejemplo tablas) para convertirlo en incompatible con el objeto XSR.

ASSOCIATE LOCATORS

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

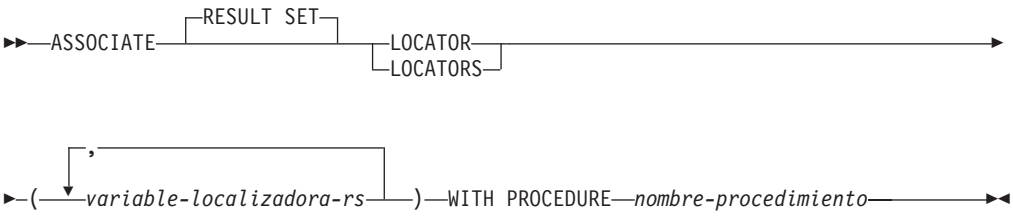
Invocación

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 de SQL compuesto (procedimiento).

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.

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.

Notas

- Si el número de variables localizadoras de conjuntos de resultados que aparecen en la sentencia ASSOCIATE LOCATORS es menor que el número de localizadores devueltos por el procedimiento 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.
- Los valores de localizadora de conjunto de resultados están disponibles para un procedimiento que se llame utilizando una sentencia EXECUTE que ejecuta la sentencia CALL preparada con anterioridad por medio de la sentencia PREPARE. Sin embargo, los valores de localizadora de conjunto de resultados no están disponibles para un procedimiento que se llame utilizando una sentencia EXECUTE IMMEDIATE.

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;
```

AUDIT

La sentencia AUDIT determina la política de comprobación que se va a utilizar para una base de datos concreta u objeto de base de datos en el servidor actual. Cada vez que se está utilizando el objeto, éste se comprobará con arreglo a dicha política.

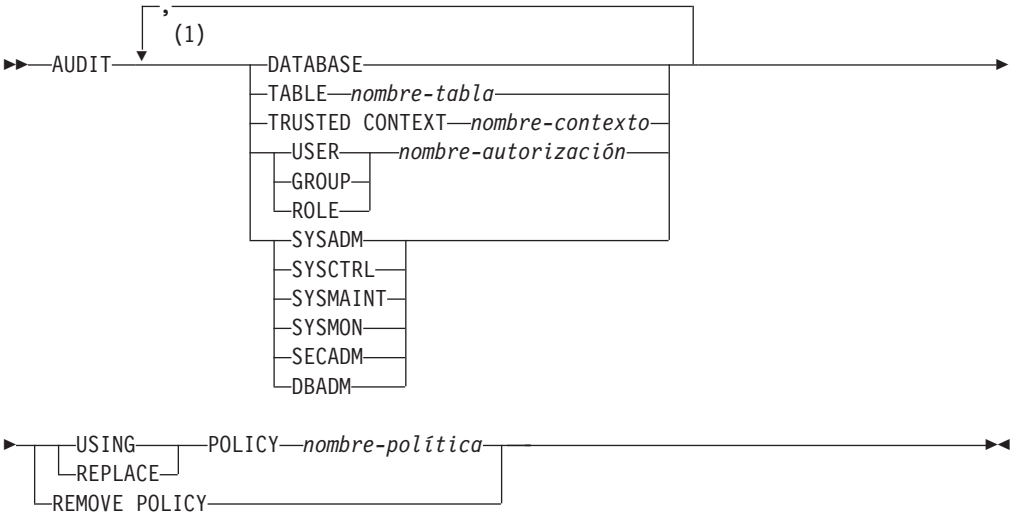
Invocación

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

Autorización

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

Sintaxis



Notas:

1 Cada cláusula (con el mismo nombre de objeto, si es de aplicación) puede especificarse como máximo una vez (SQLSTATE 42713).

Descripción

DATABASE
Especifica que una política de comprobación va a asociarse o eliminarse de la base de datos del servidor actual. Se comprueban todos los sucesos comprobables que se produzcan en la base de datos con arreglo a la política de comprobación asociada.

TABLE *nombre-tabla*
Especifica que una política de comprobación va a asociarse o eliminarse de *nombre-tabla*. El *nombre-tabla* debe identificar una tabla, tabla de consulta

materializada (MQT) o apodo que exista en el servidor actual (SQLSTATE 42704). No puede ser una vista, una tabla de catálogos, una tabla temporal declarada (SQLSTATE 42995) o una tabla escrita (SQLSTATE 42997). Cuando se acceda a la tabla sólo se accederá a los sucesos de comprobación de la categoría EXECUTE con o sin datos, incluso si la política indica que deberían comprobarse otras categorías.

TRUSTED CONTEXT *nombre-contexto*
Especifica que una política de comprobación va a asociarse o eliminarse de *nombre-contexto*. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). Se comprueban todos los sucesos comprobables que se produzcan en la conexión fiable definida por medio del contexto fiable *nombre-contexto* se comprobará con arreglo a la política de comprobación asociada.

USER *nombre-autorización*
Especifica que una política de comprobación va a asociarse o eliminarse del usuario que tiene ID de autorización *nombre-autorización*. Todos los sucesos comprobables que se inician por medio del *nombre-autorización* se comprobará con arreglo a la política de comprobación asociada.

GROUP *nombre-autorización*
Especifica que una política de comprobación va a asociarse o eliminarse del grupo que tiene ID de autorización *nombre-autorización*. Todos los sucesos comprobables que se inician por medio de usuarios que son miembros del *nombre-autorización* se comprobará con arreglo a la política de comprobación asociada. Si la pertenencia de un usuario a un grupo no puede determinarse, la política no se aplicará a dicho usuario.

ROLE *nombre-autorización*
Especifica que una política de comprobación va a asociarse o eliminarse de la función que tiene ID de autorización *nombre-autorización*. El *nombre-autorización* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). Todos los sucesos comprobables que se inician por medio de usuarios que son miembros del *nombre-autorización* se comprobará con arreglo a la política de comprobación asociada. La pertenencia a la función indirecta por medio de otras funciones o grupos es válida.

SYSADM, SYSMAINT, SYSCTRL, SYSMON, DBADM o SECADM
Especifica que una política de comprobación va a asociarse o eliminarse de la autorización especificada. Todos los sucesos comprobables que se inician por medio de un usuario que retiene la autorización especificada, incluso en el caso de que dicha autorización no se necesite para el suceso, se comprobará con arreglo a la política de comprobación asociada.

USING, REMOVE o REPLACE
Especifica si debe utilizarse, eliminarse o sustituirse la política de comprobación para el objeto especificado.

USING
Especifica que la política de comprobación va a utilizarse para el objeto especificado. No debe haberse definido una política de comprobación existente para el objeto (SQLSTATE 5U041). Si ya existe una política de comprobación, ésta debe eliminarse o sustituirse.

REMOVE
Especifica que la política de comprobación va a eliminarse para el objeto especificado. La utilización del objeto ya no se comprobará con arreglo a la política de comprobación. La asociación se suprimirá del catálogo cuando se elimina la política de comprobación del objeto.

REPLACE
Especifica que la política de comprobación va a sustituir una política de comprobación existente para el objeto especificado. Esta acción combina las opciones REMOVE y USING en un paso para asegurar que no hay ningún periodo de tiempo en el que una política de comprobación no se aplique al objeto especificado. Si una política no se estaba utilizando para el objeto especificado, REPLACE es equivalente a USING.

POLICY *nombre-política*
Especifica la política de comprobación que se va a utilizar para determinar valores de comprobación. El *nombre-política* debe identificar una política de comprobación existente en el servidor actual (SQLSTATE 42704).

Normas

- Una sentencia de SQL exclusiva de AUDIT debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas de AUDIT son:
  - AUDIT
  - CREATE AUDIT POLICY, ALTER AUDIT POLICY o DROP (AUDIT POLICY)
  - DROP (ROLE o TRUSTED CONTEXT si está asociada a una política de comprobación)
- Una sentencia de SQL exclusiva de AUDIT no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.
- Un objeto que puede estar asociado con no más de una política (SQLSTATE 5U042).

Notas

- Los cambios se graban en el catálogo, pero no surten efecto hasta después de que se ejecute una sentencia COMMIT.
- Los cambios no surten efecto hasta la siguiente unidad de trabajo que haga referencia al objeto al que se aplica la política de comprobación. Por ejemplo, si se utiliza la política de comprobación para la base de datos, ninguna unidad de trabajo actual comenzará la comprobación con arreglo a la política hasta después de que se complete una sentencia COMMIT o ROLLBACK.
- Las vistas que acceden a una tabla asociada con una política de comprobación se comprueban con arreglo a la política de la tabla subyacente.
- La política de comprobación que se aplica a una tabla no se aplica a una tabla de consulta materializada (MQT) basada en dicha tabla. Resulta recomendable que si se asocia una política de comprobación con una tabla, también se asocie dicha política con cualquier MQT basada en dicha tabla. Es posible que el compilador utilice automáticamente un MQT, aún en el caso de que una sentencia de SQL haga referencia a la tabla base; sin embargo, estará vigente la política de comprobación que se utilice para la tabla base.
- Cuando se realice una operación de usuario de conmutador en un contexto fiable, se vuelven a evaluar todas las políticas de comprobación con arreglo al usuario nuevo y no se utiliza ninguna política del usuario antiguo para la sesión actual. Esta acción se aplica específicamente a las políticas de comprobación asociadas directamente con el usuario, las pertenencias al grupo o función del usuario y las autorizaciones del usuario. Por ejemplo, si se comprobó la sesión actual debido a que el usuario anterior era un miembro de una función comprobada y el usuario conmutado no es un miembro de dicha función, la política ya no se aplicará a la sesión.

- Cuando se ejecuta una sentencia SET SESSION USER, las políticas de comprobación asociadas al usuario original (y las autorizaciones y pertenencias a grupo y función del usuario) se combinan con las políticas asociadas al usuario especificado en la sentencia SET SESSION USER. Las políticas de comprobación asociadas al usuario original siguen vigentes, al igual que las políticas para el usuario especificado en la sentencia SET SESSION USER. Si se emiten diversas sentencias SET SESSION USER en una sesión, sólo se toman en consideración las políticas asociadas al usuario original y al usuario actual.
- Si se descarta el objeto al que está asociada una política de comprobación, la asociación con la política de comprobación se elimina del catálogo y ya no existirá. Si se vuelve a crear dicho objeto con posterioridad, el objeto no se comprobará con arreglo a la política asociada con el mismo cuando se descartó el objeto.

Ejemplos

Ejemplo 1: Utilice la política de comprobación DBAUDPRF para determinar los valores de comprobación para la base de datos en el servidor actual.

```
AUDIT DATABASE USING POLICY DBAUDPRF
```

Ejemplo 2: Elimine la política de comprobación de la tabla EMPLOYEE.

```
AUDIT TABLE EMPLOYEE REMOVE POLICY
```

Ejemplo 3: Utilice la política de comprobación POWERUSERS para determinar los valores de comprobación para las autorizaciones SYSADM, DBADM y SECADM, así como el grupo DBAS.

```
AUDIT SYSADM, DBADM, SECADM, GROUP DBAS USING POLICY POWERUSERS
```

Ejemplo 4: Sustituya la política de comprobación por la función TELLER por la política nueva TELLERPRF.

```
AUDIT ROLE TELLER REPLACE POLICY TELLERPRF
```

BEGIN DECLARE SECTION

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

Invocación

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

Autorización

No se necesita.

Sintaxis

Descripción

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

Normas

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

Ejemplos

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

```
EXEC 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 bytes, debe utilizarse una variable del lenguaje principal CLOB.

Ejemplo 4: Defina las variables de lenguaje principal HVSMINT (smallint), HVBLOB50K (blob(51200)) y HVCLOBLOC (un localizador de CLOB) en un programa REXX.

```
DECLARE :HVCLOBLOC LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'
```

Observe que las variables HVSMINT y HVBLOB50K se han definido de manera implícita al utilizarlas en la sentencia FETCH.

CALL

La sentencia CALL llama a un procedimiento o a un procedimiento externo.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

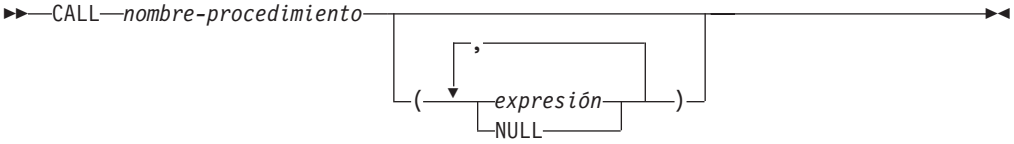
Autorización

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

- ***Asignaciones de parámetros:***

  Cuando se ejecuta la sentencia CALL, el valor de cada uno de sus argumentos se asigna (utilizando la asignación de almacenamiento) al parámetro correspondiente del procedimiento. Se pasa el control al procedimiento de acuerdo con los convenios de llamada del lenguaje principal. Cuando la ejecución del procedimiento se ha completado, el valor de cada parámetro del procedimiento se asigna (utilizando la asignación de almacenamiento) al argumento correspondiente de la sentencia CALL definido como OUT o INOUT. Si el procedimiento devuelve un error, los argumentos OUT no se definen y los argumentos INOUT no cambian. Para obtener detalles sobre las normas de asignación, consulte "Asignaciones y comparaciones".

  Cuando la sentencia CALL se encuentra en un procedimiento de SQL y llama a otro procedimiento de SQL, la asignación de parámetros XML se realiza por referencia. Cuando se pasa un argumento XML por referencia, los árboles de nodos de entrada, si existe alguno, se utilizan directamente del argumento XML, manteniendo todas las propiedades, incluyendo el orden de documentos, las identidades de nodo originales y todas las propiedades padre.
- ***Signaturas de procedimiento:***

  Un procedimiento se identifica por su esquema, un nombre de procedimiento y el número de parámetros. Esto se denomina signatura de procedimiento, que debe ser exclusiva dentro de la base de datos. Puede haber más de un procedimiento con el mismo nombre en un esquema, siempre que el número de parámetros sea distinto para cada procedimiento.
- ***Vía de acceso de SQL:***

  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 vinculación 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:
   - Para las invocaciones en las que se ha especificado un nombre de esquema (es decir, las referencias cualificadas), el nombre del esquema y el nombre del procedimiento coinciden con el nombre de invocación.
   - Para las invocaciones en la que no se ha especificado ningún nombre de esquema (es decir, las referencias no cualificadas), el nombre del procedimiento coincide con el nombre de invocación, y tiene un nombre de esquema que coincide con uno de los esquemas de la vía de acceso de SQL.
   - El número de parámetros definidos coincide con la invocación.
   - El usuario que realiza la invocación tiene el privilegio EXECUTE para el procedimiento.
2. Seleccione el procedimiento cuyo esquema sea anterior en la vía de acceso de SQL.

Si después de paso 1 no queda ningún procedimiento candidato, se devolverá un error (SQLSTATE 42884).

- ***Recuperación de DB2_RETURN_STATUS de un procedimiento de SQL:***

  Si un procedimiento de 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 de SQL, utilice la sentencia GET DIAGNOSTICS para recuperar el valor de DB2_RETURN_STATUS. El valor es -1 cuando SQLSTATE indica un error. El valor es 0 si no se ha devuelto ningún error y si la sentencia RETURN no se ha especificado en el procedimiento.

- ***Devolución de conjuntos de resultados de los procedimientos:***

  Si el programa que llama se ha escrito utilizando CLI, JDBC o SQLJ, o si el que llama es un procedimiento de 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:

  - Para cada cursor que se haya dejado abierto, se devuelve un conjunto de resultados al llamante o (para cursores WITH RETURN TO CLIENT) directamente al cliente.
  - Sólo se devuelven las filas no leídas. Por ejemplo, si el conjunto de resultados de un cursor tiene 500 filas, y el procedimiento ha leído 150 de dichas filas una vez que finaliza el procedimiento, las filas de la 151 a la 500 se devolverán al llamante o a la aplicación (según proceda).

  Si el procedimiento se ha invocado desde la CLI o desde JDBC y se ha dejado abierto más de un cursor, los conjuntos de resultados sólo pueden procesarse en el orden en que se han abierto los cursores.

- ***Mejora del rendimiento:***

  Los valores de todos los argumentos se pasan 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 se pueden llamar desde las rutinas así como desde los programas de aplicación. Cuando se llama a un procedimiento desde una rutina, se considera que la llamada está anidada.

Si un procedimiento devuelve algún conjunto de resultados de la consulta, los conjuntos de resultados se devuelven como se indica a continuación:

– Los conjuntos de resultados RETURN TO CALLER sólo puede verlos el programa que se encuentra en el nivel anterior de anidación.
– Los conjuntos de resultados de RETURN TO CLIENT sólo están visibles si el procedimiento se ha invocado desde un conjunto de procedimientos anidados. Si la función o el método tienen lugar en cualquier punto de la cadena de la llamada, el conjunto de resultados no está visible. Si el conjunto de resultados es visible, sólo lo es para la aplicación cliente que ha realizado la llamada de procedimiento inicial.

Consideremos el ejemplo siguiente:

```
Programa cliente:
EXEC SQL CALL PROCA;

  PROCA:
  EXEC SQL CALL PROCB;

    PROCB:
    EXEC SQL DECLARE B1 CURSOR WITH RETURN TO CLIENT ...;
    EXEC SQL DECLARE B2 CURSOR WITH RETURN TO CALLER ...;
    EXEC SQL DECLARE B3 CURSOR FOR SELECT UDFA FROM T1;

      UDFA:
      EXEC SQL CALL PROCC;

        PROCC:
        EXEC SQL DECLARE C1 CURSOR WITH RETURN TO CLIENT ...;
        EXEC SQL DECLARE C2 CURSOR WITH RETURN TO CALLER ...;
```

En el procedimiento PROCB:

– El cursor B1 está visible en la aplicación cliente, pero no está visible en el procedimiento PROCA.
– El cursor B2 está visible en PROCA, pero no está visible para el cliente.

En el procedimiento PROCC:

– El cursor C1 no está visible para UDFA ni para la aplicación cliente. (Puesto que UDFA aparece en la cadena de llamada entre el cliente y PROCC, el conjunto de resultados no se devuelve al cliente.)
– El cursor C2 está visible en UDFA, pero no está visible para ninguno de los procedimientos superiores.

• ***Anidamiento de procedimientos en activadores, sentencias compuestas 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 una sentencia COMMIT ni ROLLBACK.
  – No se puede acceder a los conjuntos de resultados que devuelve el procedimiento.
  – Si el procedimiento se ha definido como READS SQL DATA o MODIFIES SQL DATA, ninguna sentencia del mismo podrá acceder a la tabla que la sentencia que ha invocado el procedimiento está modificando (SQLSTATE 57053). Si el procedimiento se ha definido como MODIFIES SQL DATA, ninguna sentencia del mismo podrá modificar la tabla que la sentencia que ha invocado el procedimiento esté leyendo o modificando (SQLSTATE 57053).

Cuando se llama a un procedimiento desde una función o un método:
  - El procedimiento tiene las mismas restricciones de acceso a las tablas que la función o el método que realiza la invocación.
  - Los puntos de guardado definidos antes de invocar la función o el método no estarán visibles para el procedimiento, y los puntos de guardado definidos dentro del procedimiento no estarán visibles fuera de la función o del método.
  - Desde el cliente no es posible acceder a los conjuntos de resultados RETURN TO CLIENT que devuelve el procedimiento.
- ***Compilación de sentencias CALL de DB2 para i y DB2 para z/OS:*** La compilación de sentencias CALL de DB2 para i y DB2 para z/OS se comporta implícitamente como si CALL_RESOLUTION DEFERRED se hubiera especificado. Cuando las sentencias CALL se compilan con CALL_RESOLUTION DEFERRED, todos los argumentos deben proporcionarse mediante variables de lenguaje principal, y no se permiten las expresiones.
- ***Compatibilidades:***
  - Existe un formato más antiguo de la sentencia CALL que puede integrarse en una aplicación compilando previamente la aplicación con la opción CALL_RESOLUTION DEFERRED. Esta opción no está disponible para procedimientos SQL ni para procedimientos federados.

Ejemplos

Ejemplo 1:

Un procedimiento Java™ se define en la base de datos utilizando la sentencia siguiente:

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                                OUT COST DECIMAL(7,2),
                                                OUT QUANTITY INTEGER)
               EXTERNAL NAME 'pieza!disponibles'
               LANGUAGE JAVA
               PARAMETER STYLE DB2GENERAL;
```

Una aplicación Java llama a este procedimiento utilizando el fragmento de código siguiente:

```
...
CallableStatement stpCall ;

String sql = "CALL PARTS_ON_HAND (?, ?, ?)";

stpCall = con.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 el método Java `onhand` de la clase `parts`, porque el nombre de procedimiento especificado en la sentencia CALL se encuentra en la base de datos y tiene el nombre externo `parts!onhand`.

Ejemplo 2:

Existen seis procedimientos FOO, en cuatro esquemas distintos, registrados de la forma siguiente (observe que no aparecen todas las palabras clave necesarias):

```
CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...
```

La referencia de procedimiento es la siguiente (donde I1 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.

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.

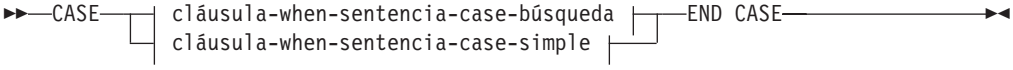
Invocación

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

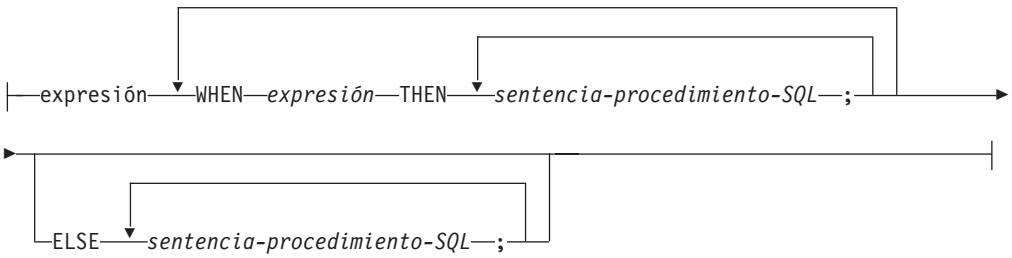
Autorización

No se requieren privilegios para invocar la sentencia CASE. Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL y expresiones incorporadas en la sentencia CASE.

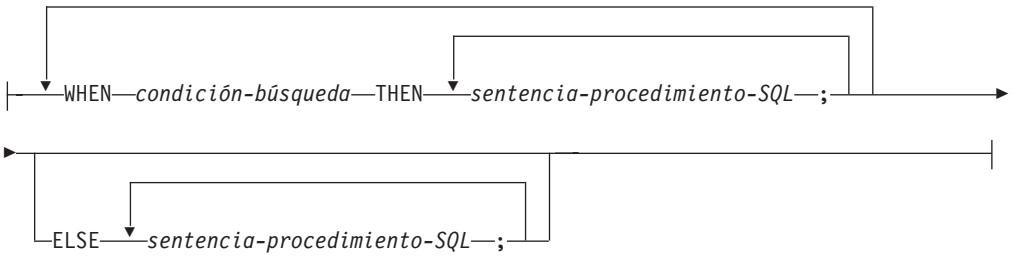
Sintaxis



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 *sentencia-procedimiento-SQL* en "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
```

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 retenido 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 cerrar los cursores que se han definido en los procedimientos llamados desde las funciones o los métodos.
- La cláusula WITH RELEASE no tiene ningún efecto para los cursores que están funcionando bajo los niveles de aislamiento CS o UR. Cuando se especifica para cursores que están funcionando bajo los niveles de aislamiento RS o RR, WITH RELEASE termina algunas de las garantías de dichos niveles de aislamiento. De forma específica, si se vuelve a abrir el cursor, un cursor RS puede experimentar el fenómeno de 'lectura no repetible', y un cursor RR puede experimentar el fenómeno de 'lectura no repetible' y el de 'fantasma'.

  Si un cursor que originalmente era RR o RS se vuelve a abrir tras haberse cerrado utilizando la cláusula WITH RELEASE, adquirirá nuevos bloqueos.
- Se aplican normas especiales a los cursores de un procedimiento que no se hayan cerrado antes de volver al programa que efectúa la llamada.

- Mientras un cursor está abierto (es decir, todavía no se ha cerrado), los cambios realizados en los valores de secuencia como resultado de las sentencias que implican a ese cursor (por ejemplo FETCH o UPDATE utilizando el cursor que incluye una expresión NEXT VALUE para una secuencia) no darán como resultado una actualización en PREVIOUS VALUE para esas secuencias desde el punto de vista del cursor. Los valores PREVIOUS VALUE de esas secuencias afectadas se actualizarán cuando el cursor se cierre explícitamente con la sentencia CLOSE. En un entorno de bases de datos particionadas, si un cursor se cierra implícitamente mediante una operación de confirmar o retrotraer, puede que no se actualice PREVIOUS VALUE con el valor generado más recientemente para la secuencia.

Ejemplo

Se utiliza un cursor para buscar una fila cada vez en las variables del programa C `dnum`, `dname` y `mnum`. Finalmente, se cierra el cursor. Si se vuelve a abrir el cursor, se sitúa de nuevo al principio de las filas en las que se debe buscar.

```
EXEC SQL  DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO
  FROM TDEPT
  WHERE ADMRDEPT = 'A00';

EXEC SQL  OPEN C1;

while (SQLCODE==0) {                                          .
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
     .
     .
}
EXEC SQL  CLOSE C1;
```

COMMENT

La sentencia COMMENT añade o sustituye comentarios en las descripciones del catálogo de diversos objetos.

Invocación

Esta sentencia 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).

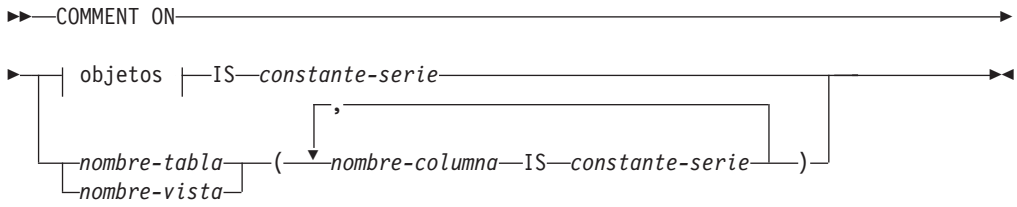
Autorización

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

- Propietario del objeto (tabla subyacente para la columna o restricción) tal como está registrado en la columna OWNER de la vista de catálogo del objeto
- Privilegio ALTERIN para el esquema (sólo aplicable a los objetos que permiten nombres de más de una parte)
- Privilegio CONTROL para el objeto (sólo aplicable a los objetos de índice, paquete, tabla o vista)
- Privilegio ALTER para el objeto (sólo aplicable a los objetos de tabla)
- WITH ADMIN OPTION (aplicable sólo a funciones)
- Autorización SECADM (aplicable sólo a política de comprobación, función, etiqueta de seguridad, componente de etiqueta de seguridad, política de seguridad u objetos de contexto fiable)
- Autorización SYSADM o DBADM (no aplicable a política de comprobación, función, etiqueta de seguridad, componente de etiqueta de seguridad, política de seguridad u objetos de contexto fiable)

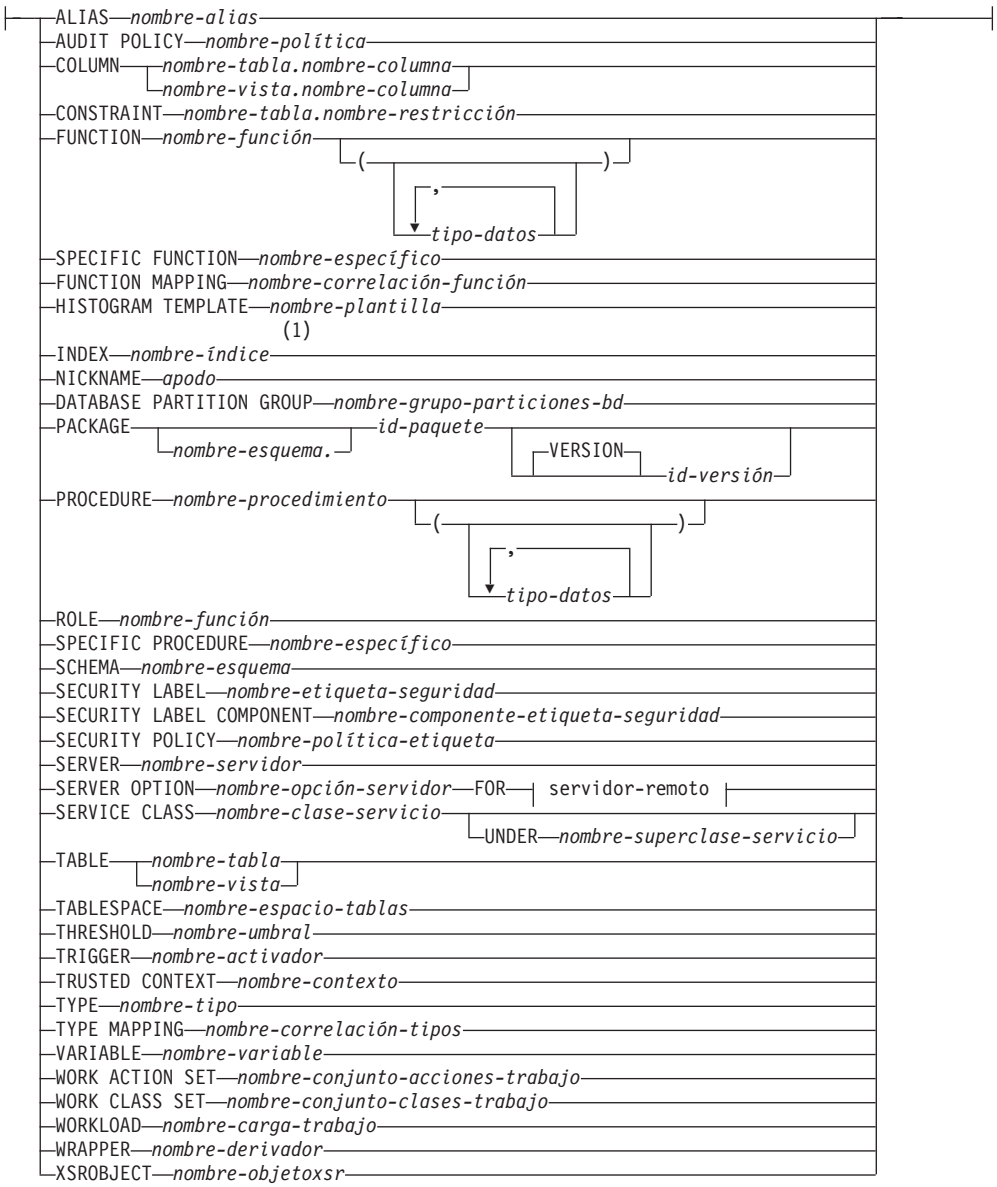
Tenga en cuenta que, para el espacio de tablas o el grupo de particiones de base de datos, el ID de autorización debe tener autorización SYSCTRL o SYSADM.

Sintaxis

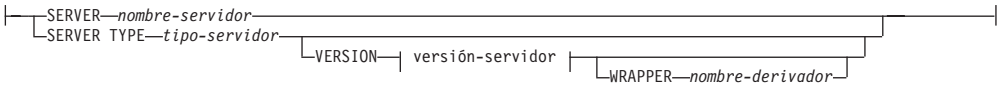


objetos:

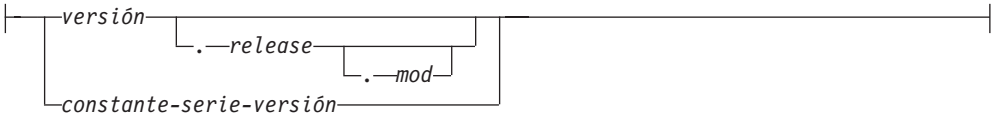
COMMENT



servidor-remoto:



versión-servidor:



Notas:

1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.

Descripción

ALIAS *nombre-alias*
Indica un comentario que se añadirá o sustituirá para un alias. El *nombre-alias* debe identificar un alias que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES para la fila que describe el alias.

AUDIT POLICY *nombre-política*
Indica un comentario que se añadirá o sustituirá para una política de comprobación. El *nombre-política* debe identificar una política de comprobación que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.AUDITPOLICIES para la fila que describe la política de comprobación.

COLUMN *nombre-tabla.nombre-columna* **o** *nombre-vista.nombre-columna*
Indica que se añadirá o se sustituirá un comentario para una columna. La combinación *nombre-tabla.nombre-columna* o *nombre-vista.nombre-columna* debe identificar una combinación de tabla y columna que exista en el servidor actual (SQLSTATE 42704), pero no debe identificar una tabla temporal global (SQLSTATE 42995). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.COLUMNS para la fila que describe la columna.

CONSTRAINT *nombre-tabla.nombre-restricción*
Indica que se añadirá o se sustituirá un comentario para una restricción. La combinación *nombre-tabla.nombre-restricción* debe identificar una restricción y la tabla que restringe; deben existir en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABCONST para la fila que describe la restricción.

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 que exista en el servidor actual.

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 sentencias de SQL estático, la opción de precompilación/vinculación 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/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto 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 sentencias de SQL estático, la opción de precompilación/vinculación 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 exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.FUNCMAPPINGS correspondiente a la fila que describe la correlación de funciones.

HISTOGRAM TEMPLATE *nombre-plantilla*
Indica un comentario que se añadirá o sustituirá para una plantilla de histograma. El *nombre-plantilla* debe identificar una plantilla de histograma que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.HISTOGRAMTEMPLATES para la fila que describe la plantilla de histograma.

INDEX *nombre-índice*
Indica que se añadirá o sustituirá un comentario para un índice o especificación de índice. El *nombre-índice* debe identificar un índice diferenciado o una especificación de índice que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.INDEXES correspondiente a la fila que describe el índice o especificación de índice.

NICKNAME *apodo*
Indica que se añadirá o sustituirá un comentario para un apodo. El *apodo* debe ser un apodo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe el apodo.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*
Indica que se añadirá o se sustituirá un comentario para un grupo de particiones de base de datos. El *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos diferenciado que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.DBPARTITIONGROUPS de la fila que describe al grupo de particiones de base de datos.

PACKAGE *nombre-esquema.id-paquete*
Indica que se añadirá o se sustituirá un comentario para un paquete. Si no se ha especificado un nombre de esquema, el esquema por omisión calificará implícitamente el ID de paquete. El nombre de esquema y el ID de paquete, junto con el ID de versión especificado de forma implícita o explícita, deben identificar un paquete que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.PACKAGES para la fila que describe el paquete.

VERSION *id-versión*
Identifica en qué versión del paquete va a realizarse el comentario. Si no se especifica un valor, la versión tomará por omisión una serie de caracteres vacía. Si existen varios paquetes con el mismo nombre de paquete pero con distintas versiones, sólo podrá comentarse una versión del paquete en una invocación de la sentencia COMMENT. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

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

PROCEDURE
Indica que se añadirá o sustituirá un comentario para un procedimiento. La instancia de procedimiento especificada debe ser un procedimiento que exista en el servidor actual.

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 sentencias de SQL estático, la opción de precompilación/vinculación 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/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

(*tipo-datos***,...)**
Deben coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente. Para procedimientos federados, el tipo de datos debe coincidir con la información del catálogo local. 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 o la escala está codificada, el valor debe coincidir exactamente con el especificado en la sentencia CREATE PROCEDURE o la información de catálogo local, en caso de un procedimiento federado.

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 en particular que debe comentarse, utilizando el nombre específico que se ha indicado o que se 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 sentencias de SQL estático, la opción de precompilación/vinculación 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.

ROLE *nombre-función*
Indica que se añadirá o sustituirá un comentario para la función. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.ROLES para la fila que describe la función.

SCHEMA *nombre-esquema*
Indica que se añadirá o sustituirá un comentario para un esquema. El *nombre-esquema* debe identificar un esquema que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SCHEMATA para la fila que describe el esquema.

SECURITY LABEL *nombre-etiqueta-seguridad*
Indica que se añadirá o sustituirá un comentario para la etiqueta de seguridad denominada *nombre-etiqueta-seguridad*. Se debe calificar el nombre con una política de seguridad y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYLABELS para la fila que describe la etiqueta de seguridad.

SECURITY LABEL COMPONENT *nombre-componente-etiqueta*
Indica que se añadirá o sustituirá un comentario para el componente de etiqueta de seguridad denominado *nombre-componente-etiqueta*. El *nombre-componente-etiqueta* debe identificar un componente de etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704). El comentario

sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYLABELCOMPONENTS para la fila que describe el componente de etiqueta de seguridad.

SECURITY POLICY *nombre-política-seguridad*
Indica que se añadirá o sustituirá un comentario para la política de seguridad denominada *nombre-política-seguridad*. El *nombre-política-seguridad* debe identificar una política de seguridad que exista en el servidor actual (SQLSTATE 42704). El componente sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SECURITYPOLICIES para la fila que describe la política de seguridad.

SERVER *nombre-servidor*
Indica que se añadirá o sustituirá un comentario para una fuente de datos. El *nombre-servidor* debe identificar una fuente de datos que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVERS correspondiente a la fila que describe la fuente de datos.

SERVER OPTION *nombre-opción-servidor* **FOR** *servidor-remoto*
Indica que se añadirá o se sustituirá un comentario para una opción de servidor.

nombre-opción-servidor
Identifica una opción de servidor. Esta opción debe ser una que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.SERVEROPTIONS correspondiente a la fila que describe la opción de servidor.

servidor-remoto
Describe la fuente de datos a la que se aplica la *opción-servidor*.

SERVER *nombre-servidor*
Indica el nombre de la fuente de datos a la que se aplica la *opción-servidor*. El *nombre-servidor* debe identificar una fuente de datos que exista en el servidor actual.

TYPE *tipo-servidor*
Especifica el tipo de fuente de datos—por ejemplo, DB2 para z/OS u Oracle—al que se aplica 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-derivador*
Identifica el derivador que se utiliza para acceder a la fuente de datos referenciada por *nombre-servidor.*

SERVICE CLASS *nombre-clase-servicio*
Indica que se añadirá o sustituirá un comentario para una clase de servicio. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). Para añadir o sustituir un comentario para una subclase de servicio, el *nombre-superclase-servicio* debe especificarse utilizando la cláusula UNDER. El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.SERVICECLASSES para la fila que describe la clase de servicio.

UNDER *nombre-superclase-servicio*
Especifica la superclase de servicio de la subclase de servicio al añadir o sustituir un comentario para la subclase de servicio. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704).

TABLE *nombre-tabla* **o** *nombre-vista*
Indica que se añadirá o se sustituirá un comentario para una tabla o vista. El *nombre-tabla* o *nombre-vista* debe identificar una tabla o vista (no un alias o apodo) que exista en el servidor actual (SQLSTATE 42704) y no debe identificar una tabla temporal declarada (SQLSTATE 42995). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLES correspondiente a la fila que describe la tabla o vista.

TABLESPACE *nombre-espacio-tablas*
Indica que se añadirá o se sustituirá un comentario para un espacio de tablas. El *nombre-espacio-tablas* debe identificar un espacio de tablas diferenciado que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye al valor de la columna REMARKS de la vista de catálogo SYSCAT.TABLESPACES para la fila que describe el espacio de tablas.

THRESHOLD *nombre-umbral*
Indica que se añadirá o suprimirá un comentario para un umbral. El *nombre-umbral* debe identificar un umbral que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.THRESHOLDS para la fila que describe el umbral.

TRIGGER *nombre-activador*
Indica que se añadirá o se sustituirá un comentario para un activador. El *nombre-activador* debe identificar un activador diferenciado que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.TRIGGERS para la fila que describe el activador.

TRUSTED CONTEXT *nombre-contexto*
Indica que se añadirá o sustituirá un comentario para un contexto fiable. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.CONTEXTS para la fila que describe el contexto fiable.

TYPE *nombre-tipo*
Indica que se añadirá o se sustituirá un comentario para un tipo definido por

el usuario. El *nombre-tipo* debe identificar un tipo definido por el usuario que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.DATATYPES para la fila que describe el tipo definido por el usuario.

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

TYPE MAPPING *nombre-correlación-tipos*
Indica que se añadirá o sustituirá un comentario para una correlación de tipos de datos definida por el usuario. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.TYPEMAPPINGS correspondiente a la fila que describe la correlación.

VARIABLE *nombre-variable*
Indica que se añadirá o sustituirá un comentario para una variable global. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.VARIABLES para la fila que describe la variable.

WORK ACTION SET *nombre-conjunto-acciones-trabajo*
Indica que se añadirá o sustituirá un comentario para un conjunto de acciones de trabajo. El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la fila REMARKS de la vista de catálogo SYSCAT.WORKACTIONSETS para la fila que describe el conjunto de acciones de trabajo.

WORK CLASS SET *nombre-conjunto-clases-trabajo*
Indica que se añadirá o sustituirá un comentario para un conjunto de clases de trabajo. El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WORKCLASSSETS para la fila que describe el conjunto de clases de trabajo.

WORKLOAD *nombre-carga-trabajo*
Indica que se añadirá o sustituirá un comentario para una carga de trabajo. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WORKLOADS para la fila que describe la carga de trabajo.

WRAPPER *nombre-derivador*
Indica que se añadirá o sustituirá un comentario para un derivador. El *nombre-activador* debe identificar un activador que exista en el servidor actual (SQLSTATE 42704). El comentario sustituye el valor de la columna REMARKS de la vista de catálogo SYSCAT.WRAPPERS correspondiente a la fila que describe el derivador.

XSROBJECT *nombre-objetoxsr*
Indica un comentario que se añadirá o sustituirá para un objeto XSR. El *nombre-objetoxsr* debe identificar un objeto XSR que exista en el servidor actual

(SQLSTATE 42704). El componente sustituye el valor para la columna REMARKS de la vista de catálogo SYSCAT.XSROBJECTS para que describe el objeto XSR.

IS *constante-serie*
Especifica el comentario que va a añadirse o sustituirse. La *constante-serie* puede ser cualquier constante de serie de caracteres con un máximo de 254 bytes. (El retorno de carro y el salto de línea cuentan como 1 byte cada uno.)

nombre-tabla | *nombre-vista* **(** { *nombre-columna* **IS** *constante-serie* } **...)**
Este formato de la sentencia COMMENT proporciona la posibilidad de especificar comentarios para múltiples columnas de una tabla o vista. Los nombres de columna no deben ser calificados, cada nombre debe identificar una columna de la tabla o vista especificada y la tabla o vista debe existir en el servidor actual. El *nombre-tabla* no puede ser una tabla temporal declarada (SQLSTATE 42995).

No puede realizarse un comentario en una columna de una vista no operativa (SQLSTATE 51024).

Notas

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
    - DISTINCT TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
    - DATA TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*

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 'algoritmo de sustitución y búsqueda masiva de Frank'
```

Ejemplo 9: Macdonald desea realizar un comentario sobre otra función SEARCH, que creó en el esquema EIGLER, utilizando el nombre específico para identificar la instancia de procedimiento que hay que comentar:

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
  'Algoritmo de destrucción y búsqueda masiva de Patrick'
```

Ejemplo 10: Realice un comentario sobre la función OSMOSIS en el esquema BIOLOGY, donde se sabe que sólo hay un procedimiento con dicho nombre:

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
  IS 'Cálculos al modelar ósmosis'
```

Ejemplo 11: Realice un comentario sobre una especificación de índice denominada INDEXSPEC.

```
COMMENT ON INDEX INDEXSPEC
  IS 'Una especificación de índice que indica al optimizador
  que la tabla referenciada por el apodo NICK1 tiene un índice.
```

Ejemplo 12: Realice un comentario sobre el derivador cuyo nombre por omisión es NET8.

```
COMMENT ON WRAPPER NET8
  IS 'El derivador de las fuentes de datos asociadas al
      software cliente Net8 de Oracle.'
```

Ejemplo 13: Cree un comentario sobre el esquema XML HR.EMPLOYEE.

```
COMMENT ON XSROBJECT HR.EMPLOYEE
  IS 'Éste es el esquema XML base para los datos de empleado.'
```

Ejemplo 14: Cree un comentario para el contexto fiable APPSERVER.

```
COMMENT ON TRUSTED CONTEXT APPSERVER
  IS 'WebSphere Server'
```

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 a 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

No se necesita.

Sintaxis

Descripción

Finaliza la unidad de trabajo en la que se ejecuta la sentencia COMMIT y se inicia una nueva unidad de trabajo. Se confirman todos los cambios realizados por las siguientes sentencias ejecutadas durante la unidad de trabajo: ALTER, COMMENT, CREATE, DROP, GRANT, LOCK TABLE, REVOKE, SET INTEGRITY, SET Variable y las sentencias de cambio de datos (INSERT, DELETE, MERGE, UPDATE), incluidas las anidadas en una consulta.

Sin embargo, las sentencias siguientes no están bajo el control de la transacción y los cambios que éstas realizan son independientes de la sentencia COMMIT:

- SET CONNECTION
- SET 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.

Las siguientes sentencias tienen un comportamiento diferente al de otras sentencias de lenguaje de definición de datos (DDL) y de lenguaje de control de datos (DCL). Los cambios realizados por estas sentencias no surten efecto hasta que se confirma la sentencia, incluso para la conexión actual que emite la sentencia. Una aplicación sólo puede emitir una de estas sentencias cada vez y sólo se permite una de estas sentencias dentro de una unidad de trabajo. Cada sentencia debe ir seguida de una sentencia COMMIT o ROLLBACK antes de que pueda emitirse otra de estas sentencias.

- CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
- CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
- CREATE WORK ACTION, ALTER WORK ACTION o DROP (WORK ACTION)
- CREATE WORK CLASS, ALTER WORK CLASS o DROP (WORK CLASS)
- CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
- GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)

Notas

- Se recomienda encarecidamente que cada proceso de aplicación finalice explícitamente su unidad de trabajo antes de terminar. Si el programa de aplicación finaliza normalmente sin una sentencia COMMIT ni ROLLBACK entonces el gestor de bases de datos intenta una confirmación o retrotracción según el entorno de aplicación.
- Para obtener información acerca del efecto de COMMIT en las sentencias de SQL dinámico colocadas en la antememoria, consulte “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
```

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

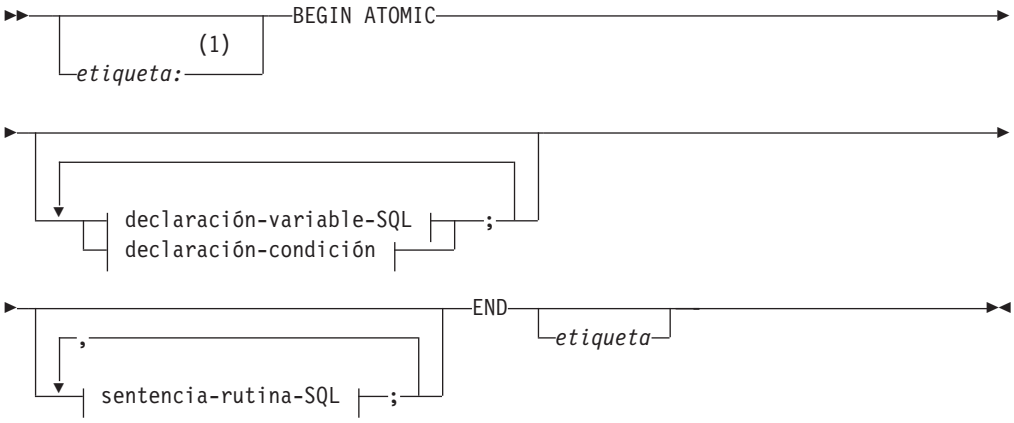
Esta sentencia puede incorporarse a un activador, una función de SQL o un método de SQL o bien emitirse mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

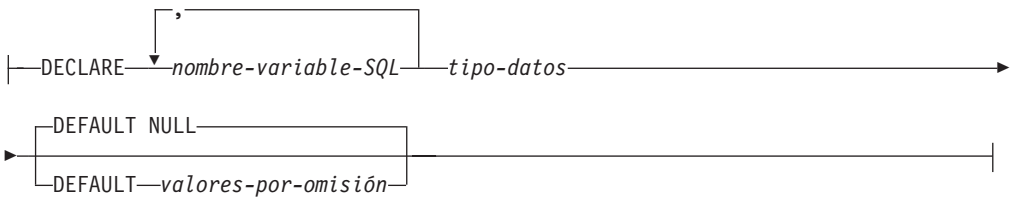
No se necesita ningún privilegio para invocar una sentencia dinámica compuesta. Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta.

Sintaxis

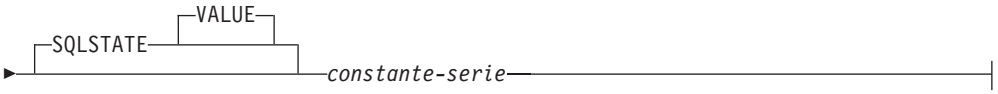
sentencia-dinámica-compuesta



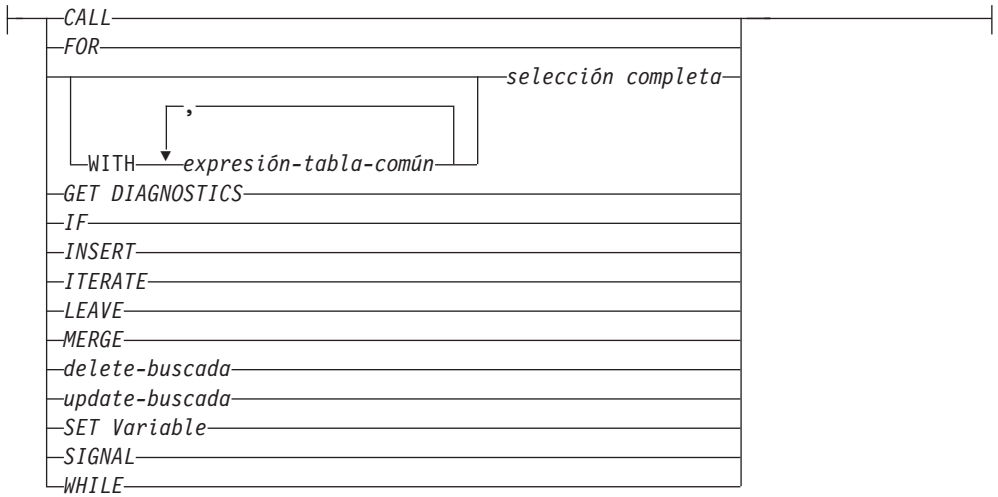
declaración-variable-SQL:



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 una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC
ATOMIC indica que, si se produce un error en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta y no se procesarán las restantes sentencias de SQL de la sentencia.

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 de 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 todos los nombres de variables SQL a mayúsculas. 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. No se da soporte a los tipos XML (SQLSTATE 429BB) y tampoco se da soporte a los tipos array (SQLSTATE 42815).

DEFAULT *valores-por-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 de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una condición dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42737).

FOR SQLSTATE *constante-serie*
Especifica el SQLSTATE asociado con la condición. 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 de 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.

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
```

de SQL compuesto (incorporado)

Combina una o más sentencias de SQL distintas (*subsentencias*) en un bloque ejecutable.

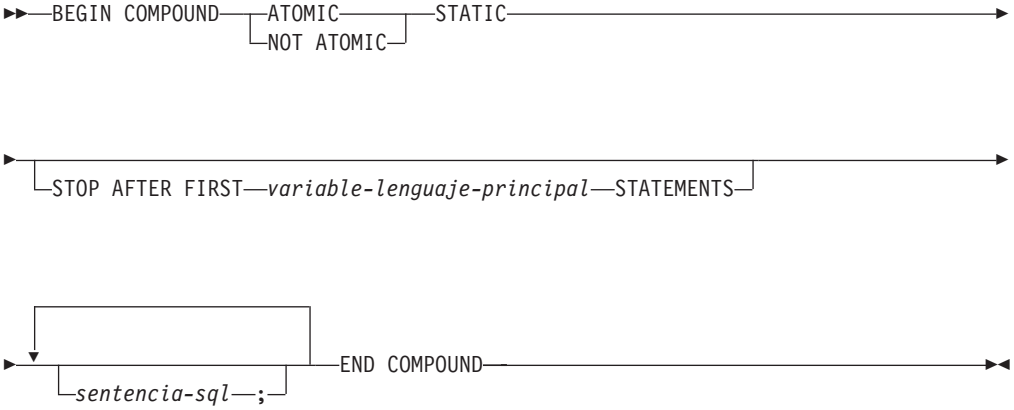
Invocación

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 se soporta en REXX.

Autorización

No se requieren privilegios para invocar una sentencia compuesta incorporada. Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta.

Sintaxis



Descripción

ATOMIC
Especifica que, si falla alguna de las subsentencias de la sentencia de SQL 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-lenguaje-principal
Un entero pequeño que especifica el número de subsentencias que se deben ejecutar.

STATEMENTS
Completa la cláusula STOP AFTER FIRST *variable-lenguaje-principal*.

sentencia-sql-
Todas las sentencias ejecutables, excepto las siguientes, pueden estar contenidas en una sentencia de SQL compuesta estática incorporada:

```
CALL                FETCH
CLOSE               OPEN
CONNECT             PREPARE
SQL compuesto       RELEASE (conexión)
DESCRIBE            ROLLBACK
DISCONNECT          SET CONNECTION
EXECUTE IMMEDIATE   variable SET
```

Nota: INSERT, UPDATE y DELETE no reciben soporte en el SQL 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-lenguaje-principal* STATEMENTS indique que no deben ejecutarse todas las subsentencias. Por ejemplo, suponga que COMMIT es la última subsentencia en un bloque SQL compuesto formado por 100 subsentencias. Si la cláusula STOP AFTER FIRST STATEMENTS indica que sólo deben ejecutarse 50 subsentencias, entonces COMMIT será la subsentencia número 51.

Se devolverá un error si se incluye COMMIT al utilizar CONNECT TYPE 2 o ejecutar en un entorno de proceso de transacciones distribuidas XA (SQLSTATE 25000).

Normas

- DB2 Connect no da soporte a las sentencias SELECT que seleccionan columnas LOB en un bloque SQL compuesto.
- No está permitido ningún código de lenguaje principal en una sentencia de SQL compuesta; es decir, no está permitido ningún código de lenguaje principal entre las subsentencias que componen la sentencia de SQL compuesta.
- DB2 Connect sólo aceptará sentencias de SQL compuestas 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;
```

SQL compuesto (procedimiento)

Una sentencia compuesta de procedimiento agrupa otras sentencias en un procedimiento de SQL. Dentro de una sentencia compuesta se pueden declarar variables SQL, cursores y gestores de condiciones.

Invocación

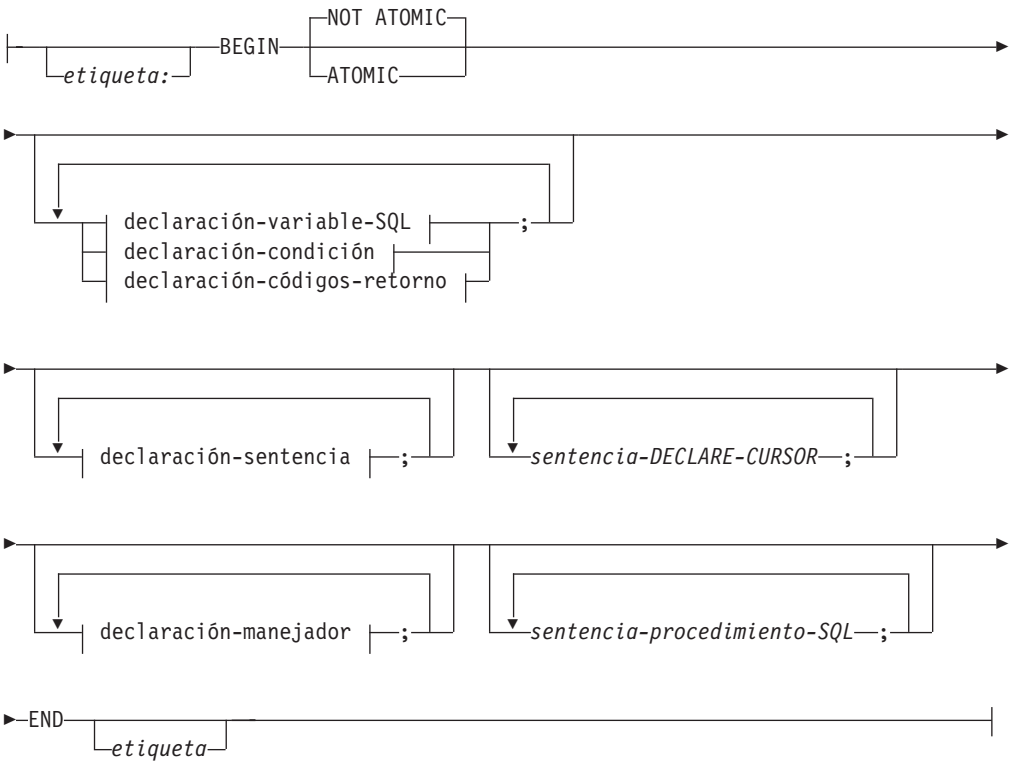
Esta sentencia sólo puede incorporarse en un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización

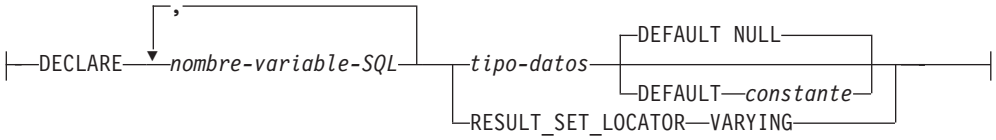
No se requieren privilegios para invocar una sentencia compuesta de procedimiento. Sin embargo, el ID de autorización de la sentencia debe mantener todos los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia compuesta. Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

Sintaxis

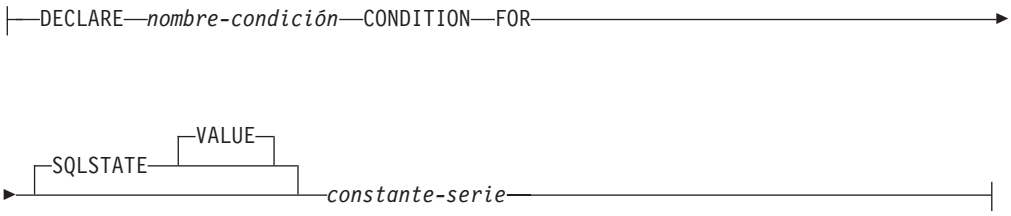
sentencia-compuesta-procedimiento:



declaración-variable-SQL:

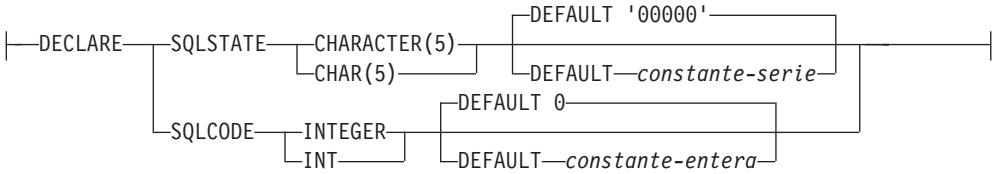


declaración-condición:

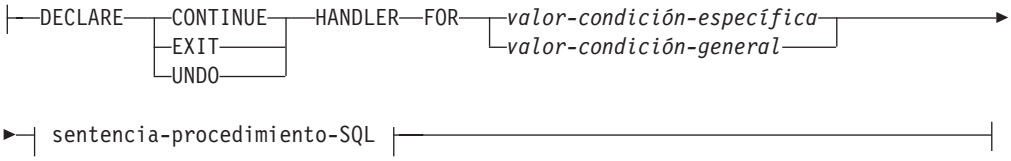


declaración-sentencia:

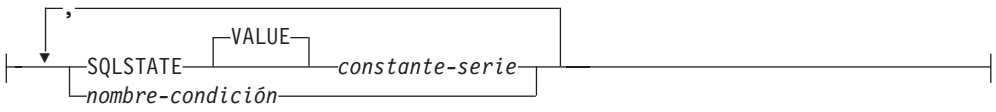
declaración-códigos-retorno:



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 una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

ATOMIC o NOT ATOMIC
ATOMIC indica que si se produce una condición de excepción no manejada en la sentencia compuesta, se retrotraerán todas las sentencias de SQL de la sentencia compuesta. NOT ATOMIC indica que una condición de excepción no manejada dentro de la sentencia compuesta no da lugar a la retrotracción de la sentencia compuesta.

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 todos los nombres de variables SQL a mayúsculas. 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. REFERENCE y los tipos estructurados definidos por el usuario no están soportados (SQLSTATE 429BB).

DEFAULT *constante* **o NULL**
Define el valor por omisión de la variable SQL. La variable se inicializa cuando se invoca el procedimiento de 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 de la sentencia compuesta en la que está declarada, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al nombre de una condición dentro de la sentencia compuesta en la que está declarada, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42737).

FOR SQLSTATE *constante-serie*
Especifica el SQLSTATE 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 de SQL. Estas variables sólo se pueden declarar una vez para cada procedimiento de SQL.

sentencia-declare-cursor
Declara un cursor en el cuerpo del procedimiento. Todos los cursores han de tener un nombre exclusivo dentro de la sentencia compuesta en la que están declarados, salvo en el caso de las declaraciones de las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 42734). Sólo se puede hacer referencia al cursor dentro de la sentencia compuesta en la que está declarado, incluidas las sentencias compuestas anidadas dentro de dicha sentencia compuesta (SQLSTATE 34000).

Utilice una sentencia OPEN para abrir el cursor, y una sentencia FETCH para leer filas utilizando el cursor. Para que el procedimiento de SQL devuelva conjuntos de resultados a la aplicación cliente, el cursor se debe declarar utilizando la cláusula WITH RETURN. El ejemplo siguiente devuelve un conjunto de resultados a la aplicación cliente:

```
CREATE PROCEDURE RESULT_SET()
  LANGUAGE SQL
  RESULT SETS 1
  BEGIN
    DECLARE C1 CURSOR WITH RETURN FOR
      SELECT id, name, dept, job
        FROM staff;
    OPEN C1;
END
```

Nota: Para procesar conjuntos de resultados debe grabar la aplicación cliente utilizando una de las interfaces de programación de aplicaciones siguientes: DB2 Call Level Interface (DB2 Call Level Interface), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC) o SQL incorporado para Java (SQLJ).

Para obtener más información acerca de la declaración de un cursor, consulte "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 de 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 gestor de condiciones debe ser ATOMIC.

Las condiciones que dan lugar a la activación del gestor de condiciones se definen en la declaración-gestor-condiciones, tal como se indica a continuación:

calor-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 debe 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 de SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control de SQL admiten la especificación de una lista de sentencias de procedimiento de SQL.

sentencia-SQL
Todas las sentencias de SQL ejecutables 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 gestor de condiciones no puede contener el mismo *nombre-condición* o valor SQLSTATE más de una vez y no puede contener un valor SQLSTATE y un *nombre-condición* que representen el mismo valor SQLSTATE.
  - Cuando se declaran dos o más 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 gestor de condiciones no pueden especificar la misma condición específica, ya sea como un valor SQLSTATE o como un *nombre-condición* que represente el mismo valor.
  - Un 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 gestor de condiciones para un *valor-condición-específica* o un *valor-condición-general* C declarado en un ámbito interno tiene prioridad respecto a cualquier otro gestor de condiciones para C declarado en un ámbito de inclusión.
    - Cuando, en el mismo ámbito, se declaran un gestor de condiciones específico para una condición *C* y un gestor de condiciones general que también manejaría *C*, el gestor de condiciones específico tiene prioridad respecto al gestor de condiciones 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 finalización para la que no existe un gestor de condiciones adecuado, la ejecución continuará con la siguiente sentencia de SQL.
- La referencia a variables o a parámetros del tipo de datos XML en los procedimientos de SQL después de que se haya producido una operación de confirmación o de retrotracción, sin que primero se hayan asignado valores nuevos a estas variables, no tiene soporte (SQLSTATE 560CE).

Notas

- ***Asignaciones de XML:*** Las asignaciones a parámetros y variables de tipos de datos XML se efectúan según el valor. Cuando los valores XML se asignan según el valor, se realiza una copia ejecutando efectivamente las acciones siguientes para cada elemento de la secuencia:
  - El árbol de nodos, si existe, se copia del valor XML original, manteniendo el orden del documento
  - Se asignan identidades de nodo nuevas a cada nodo
  - La propiedad padre del nodo raíz, si lo hay, se establece en vacío

  Los parámetros del tipo de datos XML de una sentencia CALL se pasan a un procedimiento de SQL por referencia. Cuando los valores XML se pasan por referencia, los árboles de nodos de entrada, (si los hay), se utilizan directamente desde el argumento de XML, con lo que se conservan todas las propiedades, incluidos el orden del documento, las identidades de nodo originales y todas las propiedades padre.

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 este manejador finaliza, la ejecución continúa después
    -- de la sentencia compuesta CS1; el procedimiento A terminará.
  END;

  -- Realizar aquí algún trabajo ...
  CS2: BEGIN
    DECLARE CONTINUE HANDLER FOR D
    H2: BEGIN
      -- Realizar recuperación local y reenviar la condición de
      -- error al descriptor de contexto externo para su proceso
      -- adicional.

      -- ...

      RESIGNAL C; -- activará el descriptor de contexto UNDO H1; la ejecución
                  -- NO VOLVERÁ aquí. Los cursores locales
-- declarados en H2 y CS2 se cerrarán.
    END;

    -- Realizar aquí algún trabajo adicional ...

    -- Simular la generación de la condición D por parte de alguna
    -- sentencia de SQL en la sentencia compuesta CS2:
    SIGNAL D; -- activará H2
  END;
END
```

CONNECT (Tipo 1)

La sentencia CONNECT (Tipo 1) conecta un proceso de aplicación con el servidor de aplicaciones identificado según las normas para una unidad de trabajo remota.

Un proceso de aplicación sólo puede estar conectado a un único servidor de aplicaciones en un momento dado. Se denomina *servidor actual*. Puede establecerse un servidor de aplicaciones por omisión cuando se inicializa el peticionario de aplicaciones. Si la conexión implícita está disponible y se inicia un proceso de aplicación, se conecta de manera implícita al servidor de aplicaciones por omisión. El proceso de la aplicación puede conectarse explícitamente a otro servidor de aplicaciones ejecutando una sentencia CONNECT. Una conexión dura hasta que se ejecuta una sentencia CONNECT RESET o una sentencia DISCONNECT o hasta que otra sentencia CONNECT cambia el servidor de aplicaciones.

Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

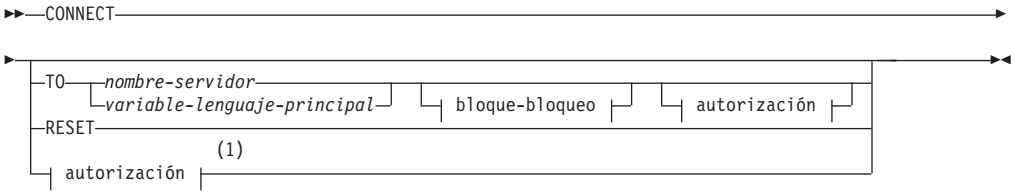
El proceso de CONNECT pasa por dos niveles de control de acceso. Ambos niveles deben cumplirse para que la conexión sea correcta.

El primer nivel de control de acceso es la autentificación, en la que el ID de usuario asociado a la conexión debe autentificarse correctamente en función del método de autentificación configurado para el servidor. Cuando se produce la autentificación correcta, se calcula un ID de autorización de DB2 a partir del ID de usuario de conexión, en función del conector de autentificación en vigor en el servidor. A continuación, este ID de autorización de DB2 debe pasar el segundo nivel de control de acceso de la conexión, es decir, la autorización. Para ello, este ID de autorización debe tener como mínimo una de las autorizaciones siguientes:

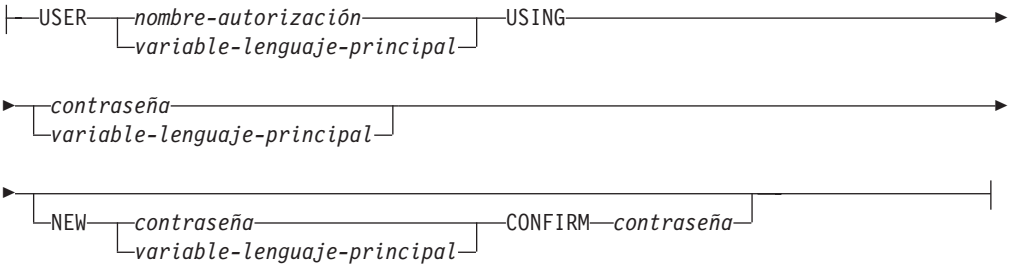
- Autorización CONNECT
- Autorización DBADM
- Autorización SYSADM
- Autorización SYSCTRL
- Autorización SYSMAINT

Nota: en el caso de una base de datos dividida en particiones, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones de la base de datos.

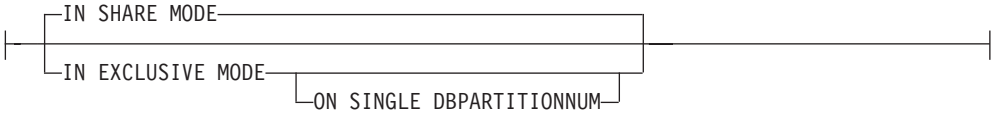
Sintaxis



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 encuentran en el campo SQLERRMC (ya que están con una sentencia CONNECT correcta).

Esta forma de CONNECT:

- No necesita que el proceso de aplicación esté en estado conectable.
- Si está conectado, no cambia el estado de conexión.
- Si está desconectado y está disponible la conexión implícita, se realiza una conexión con el servidor de aplicaciones por omisión. En este caso, el código de país o de región y la página de códigos del servidor de aplicaciones se ponen en el campo SQLERRMC, como una sentencia CONNECT satisfactoria.
- Si está desconectado y no está disponible la conexión implícita, el proceso de aplicación permanece desconectado.

- No cierra los cursores.

TO *nombre-servidor***o***variable-lenguaje-principal*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o una *variable-lenguaje-principal* que contenga el nombre-servidor.

Si se especifica una *variable-lenguaje-principal*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable de indicador. El *nombre-servidor* contenido en la *variable-lenguaje-principal* 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.

Cuando se ejecuta la sentencia CONNECT, el proceso de aplicación debe estar en un estado conectable.

Conexión correcta:

Si la sentencia CONNECT es satisfactoria:

- Se cierran todos los cursores abiertos, se destruyen todas las sentencias preparadas y se liberan todos los bloqueos del servidor de aplicaciones anterior.
- El proceso de aplicación se desconecta de su servidor de aplicaciones anterior, si lo hay, y se conecta al servidor de aplicaciones identificado.
- El nombre real del servidor de aplicaciones (no un alias) se coloca en el registro especial CURRENT SERVER.
- Se coloca información acerca del servidor de aplicaciones en el campo SQLERRP de SQLCA. Si el servidor de aplicaciones es un producto IBM, la información tiene el formato *pppvvrrm*, donde:
  - *ppp* identifica el producto de la manera siguiente:
    - DSN para DB2 para z/OS
    - ARI para DB2 Server para VSE & VM
    - QSQ para DB2 para i5/OS
    - SQL para DB2 Database para Linux, UNIX y Windows
  - *vv* es un identificador de versión de dos dígitos como, por ejemplo, '08'
  - *rr* es un identificador de release de dos dígitos como, por ejemplo, '01'
  - *m* es un identificador de nivel de modificación de un único carácter como, por ejemplo, '0'.

  Este release (Versión 9.5) de DB2 Database para Linux, UNIX y Windows se identifica como 'SQL09050'.
- El campo SQLERRMC de la SQLCA se establece en los valores siguientes (separados por X'FF')
  1. el código de país o región del servidor de aplicaciones (o espacios en blanco si se utiliza DB2 Connect)
  2. la página de códigos del servidor de aplicaciones (o CCSID si se utiliza DB2 Connect)
  3. el ID de autorización (sólo los primeros 8 bytes como máximo),
  4. el alias de base de datos,
  5. el tipo de plataforma del servidor de aplicaciones. Los valores reconocidos actualmente son:

Símbolo
Servidor

QAS DB2 para System i

QDB2 DB2 para z/OS

QDB2/6000
DB2 Database para AIX

QDB2/HPUX
DB2 Database para HP-UX

QDB2/LINUX
DB2 Database para Linux

QDB2/NT
DB2 Database para Windows

QDB2/SUN
DB2 Database para el sistema operativo Solaris

QSQLDS/VM
DB2 Server para VM

QSQLDS/VSE
DB2 Server para VSE

6. El ID de agente. Identifica al agente que se ejecuta en el gestor de bases de datos en nombre de la aplicación. Este campo es el mismo que el elemento **agent_id** que devuelve el supervisor de bases de datos.
7. El índice de agente. Identifica el índice del agente y se utiliza para el servicio.
8. Número de partición de base de datos. 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 base de datos en una base de datos particionada. Si la base de datos no puede distribuirse, 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 Autenticado mediante 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.

  9 Autentificado utilizando un mecanismo de seguridad de plugin de API GSS externo.

  11 Autenticado en el servidor, que acepta datos cifrados.

  255 Autentificación no especificada.
- El campo SQLERRD(6) de SQLCA devuelve el número de la partición de base de datos con la que se ha establecido la conexión si se ha distribuido la base de datos. 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 no es satisfactoria:

- El campo SQLERRP de la SQLCA se establece en el nombre del módulo del peticionario de aplicaciones que ha detectado el error. Los tres primeros caracteres del nombre del módulo identifican el producto.
- Si la sentencia CONNECT da un resultado anómalo porque el proceso de aplicación no está en estado conectable, no se cambiará el estado de conexión del proceso de aplicación.
- Si la sentencia CONNECT no se ejecuta satisfactoriamente porque el *nombre-servidor* no se encuentra en el directorio local, se emitirá un mensaje de error (SQLSTATE 08001) y el estado de conexión del proceso de la aplicación no cambiará:

  - Si el peticionario de aplicación no estaba conectado a un servidor de aplicaciones, el proceso de aplicación sigue sin estar conectado.
  - Si el peticionario de aplicaciones ya estaba conectado a un servidor de aplicaciones, el proceso de aplicación permanece conectado a ese servidor de aplicaciones. Las sentencias posteriores se ejecutan en dicho servidor de aplicaciones.
- Si la sentencia CONNECT no se ejecuta satisfactoriamente por cualquier otro motivo, el proceso de la aplicación pasará a estado de desconexión.

IN SHARE MODE
Permite otras conexiones simultáneas con la base de datos e impide que otros usuarios se conecten a la base de datos en modalidad exclusiva.

IN EXCLUSIVE MODE
Evita que los procesos de aplicación simultáneos ejecuten cualquier operación en el servidor de aplicaciones, a menos que tengan el mismo ID de autorización que el usuario que mantiene el bloqueo exclusivo. DB2 Connect no da soporte a esta opción.

ON SINGLE 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* **o** *variable-lenguaje-principal*
Identifica el ID de usuario que intenta conectarse con el servidor de aplicaciones. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres. que no incluye una variable de indicador. El ID de usuario que está dentro de la *variable-lenguaje-principal* debe justificarse por la izquierda y no debe delimitarse utilizando comillas.

USING *contraseña* **o** *variable-lenguaje-principal*
Identifica la contraseña del ID de usuario que intenta conectarse con el servidor de aplicaciones. La *contraseña* o la *variable-lenguaje-principal* pueden tener una longitud de hasta 14 bytes. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 14 y no debe incluir una variable de indicador.

NEW *contraseña/variable-lenguaje-principal* **CONFIRM** *contraseña*
Identifica la nueva contraseña que debe asignarse al ID de usuario que se identifica por medio de la opción USER. La *contraseña* o la *variable-lenguaje-principal* pueden tener una longitud de hasta 14 bytes. Si se especifica una variable del lenguaje principal, debe ser una variable de serie de caracteres con un atributo de longitud no superior a 14 y no debe incluir una variable de indicador. El sistema en que se cambiará la contraseña depende de cómo esté configurada la autentificación de usuario. No se pueden asignar contraseñas nuevas utilizando esta cláusula en los servidores siguientes para los releases indicados (y posteriores): DB2 Universal Database Versión 8 en sistemas operativos AIX y Windows, DB2 Versión 9.1 Fix Pack 3 o posterior en sistemas operativos Linux, DB2 para z/OS Versión 7 y DB2 para i5/OS V6R1. A fin de dar soporte al cambio de contraseñas para los productos de base de

datos DB2 en Linux, la instancia de DB2 debe estar configurada para utilizar los conectores de seguridad IBMOSchgpwdclient e IBMOSchgpwdserver.

Notas

- Es conveniente que la primera sentencia de SQL que ejecute un proceso de la aplicación sea la sentencia CONNECT.
- Si se emite una sentencia CONNECT al servidor de aplicaciones actual con otro ID de usuario y contraseña, la conversación se desasignará y se reasignará. El gestor de bases de datos cierra todos los cursores (con pérdida de la posición del cursor si se ha utilizado la opción WITH HOLD).
- Si se emite una sentencia CONNECT al servidor de aplicaciones actual con el mismo ID de usuario y contraseña, la conversación no se desasignará ni se reasignará. En este caso, los cursores no se cierran.
- Para utilizar un entorno de bases de datos particionadas con varias particiones, el usuario o la aplicación debe conectarse a una de las particiones de base de datos indicadas en el archivo `db2nodes.cfg`. Debe intentar garantizar que todos los usuarios no utilicen la misma partición de base de datos como partición de coordinador.
- El *nombre-autorización* SYSTEM no se puede especificar explícitamente en la sentencia CONNECT. No obstante, en sistemas operativos Windows, las aplicaciones locales que se ejecutan con la cuenta de sistema local pueden conectarse implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Cuando se establece conexión con Windows Server de forma explícita, el *nombre-autorización* o *variable-lenguaje-principal* de usuario puede especificarse utilizando el nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft® Windows. El calificador debe ser un nombre de estilo NetBIOS, que tiene una longitud máxima de 15 bytes. Por ejemplo, 'Dominio\Usuario'.
- Puede que no pueda accederse a la base de datos si ésta no se ha activado explícitamente, si una aplicación de cliente realiza reconexiones frecuentes o si el intervalo de tiempo entre emisiones de los mandatos DEACTIVATE DATABASE y ACTIVATE DATABASE se ha establecido en un valor muy breve. Active la base de datos mediante la emisión del mandato ACTIVATE DATABASE y, a continuación, intente conectarse con la base de datos.
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM

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);
```

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 proceso de CONNECT pasa por dos niveles de control de acceso. Ambos niveles deben cumplirse para que la conexión sea correcta.

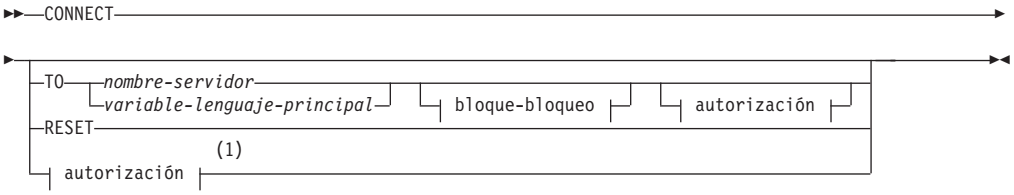
El primer nivel de control de acceso es la autentificación, en la que el ID de usuario asociado a la conexión debe autentificarse correctamente en función del método de autentificación configurado para el servidor. Cuando se produce la autentificación correcta, se calcula un ID de autorización de DB2 a partir del ID de usuario de conexión, en función del conector de autentificación en vigor en el servidor. A continuación, este ID de autorización de DB2 debe pasar el segundo nivel de control de acceso de la conexión, es decir, la autorización. Para ello, este ID de autorización debe tener como mínimo una de las autorizaciones siguientes:

- Autorización CONNECT
- Autorización DBADM
- Autorización SYSADM
- Autorización SYSCTRL
- Autorización SYSMAINT

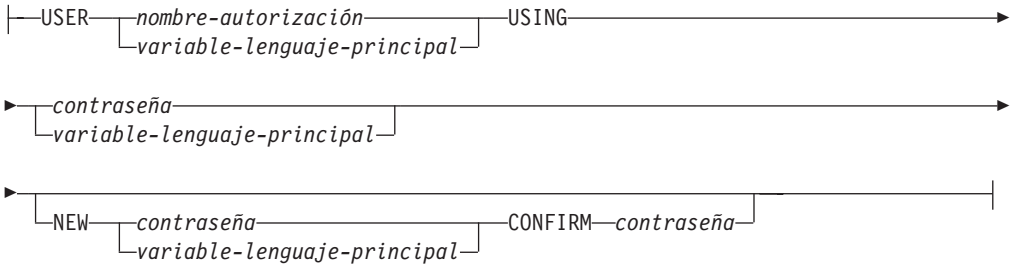
Nota: En el caso de una base de datos dividida en particiones, las definiciones de usuario y de grupo deben ser idénticas en todas las particiones de la base de datos.

Sintaxis

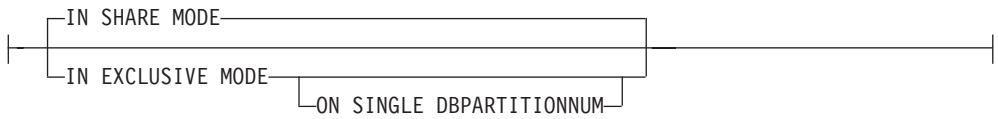
La selección entre el Tipo 1 y el Tipo 2 se determina según las opciones de precompilador. Si desea obtener una visión general de estas opciones, consulte “Conexión a bases de datos relacionales distribuidas”.



autorización:



bloque-bloqueo:



Notas:

1 Este formato sólo es válido si se ha habilitado la conexión implícita.

Descripción

TO *nombre-servidor/variable-lenguaje-principal*
Las normas para codificar el nombre del servidor son las mismas que para el Tipo 1.

Si la opción SQLRULES(STD) está en vigor, el *nombre-servidor* no debe identificar una conexión existente del proceso de aplicación; de lo contrario, se produce un error (SQLSTATE 08002).

Si la opción SQLRULES(DB2) está en vigor y el *nombre-servidor* identifica una conexión existente del proceso de aplicación, esa conexión se convierte en la actual y la conexión anterior se coloca en estado inactivo. Es decir, el efecto de la sentencia CONNECT en esta situación es el mismo que el de la sentencia SET CONNECTION.

Para obtener información acerca de la especificación de SQLRULES, consulte el apartado "Opciones que rigen la semántica de la unidad de trabajo distribuida".

Conexión correcta

Si la sentencia CONNECT es satisfactoria:

- Se crea (o deja de estar inactiva) una conexión con el servidor de aplicaciones y se coloca en estado actual y mantenido.
- Si CONNECT TO se dirige a un servidor diferente del actual, la conexión actual se coloca en estado inactivo.
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma forma que con CONNECT (Tipo 1).

Conexión no satisfactoria

Si la sentencia CONNECT no es satisfactoria:

- No importa cuál haya sido la razón de la anomalía, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.

- Al igual que para un CONNECT de Tipo 1 no satisfactorio, el campo SQLERRP de la SQLCA se establece en el nombre del módulo en el peticionario o servidor de aplicaciones que ha detectado el error.

CONNECT (sin ningún operando), IN SHARE/EXCLUSIVE MODE, USER y USING

Si existe una conexión, el Tipo 2 se comporta como un Tipo 1. El ID de autorización y el alias de base de datos se colocan en el campo SQLERRMC de la SQLCA. Si no existe una conexión, no se realiza ningún intento de establecer una conexión implícita y los campos SQLERRP y SQLERRMC devuelven un blanco. (Las aplicaciones pueden comprobarse si existe una conexión actual comprobando estos campos.)

CONNECT sin operandos que incluya USER y USING todavía puede conectar un proceso de aplicación a una base de datos mediante la variable de entorno DB2DBDFT. Este método es equivalente a CONNECT RESET Tipo 2, pero está permitido utilizar un ID de usuario y una contraseña.

RESET

Equivale a una conexión explícita con la base de datos por omisión, si está disponible. Si no hay una base de datos por omisión que esté disponible, el estado de conexión del proceso de aplicación y los estados de sus conexiones permanecen invariables.

La disponibilidad de una base de datos por omisión se determina de acuerdo con las opciones de instalación, las variables de entorno y los valores de autentificación.

Normas

- Como se describe en “Opciones que rigen la semántica de la unidad de trabajo distribuida”, un conjunto de opciones de conexión controla la semántica de la gestión de conexiones. Se asignan valores por omisión a todos los archivos fuente preprocesados. Una aplicación puede constar de múltiples archivos fuente precompilados con diferentes opciones de conexión.

  A menos que primero se haya ejecutado un mandato SET CLIENT o una API, las opciones de conexión que se utilizan al procesar previamente el archivo fuente que contiene la primera sentencia de SQL que se ejecuta en tiempo de ejecución se convierten en las opciones de conexión que están en vigor.

  Si posteriormente se ejecuta una sentencia CONNECT desde un archivo fuente previamente procesado con opciones de conexión distintas sin que intervenga en la ejecución ningún mandato SET CLIENT o la API, se devuelve un error (SQLSTATE 08001). Observe que, una vez que se ha ejecutado un mandato SET CLIENT o una API, se pasan por alto las opciones de conexión utilizadas al preprocesar todos los archivos fuente de la aplicación.

  En el Ejemplo 1 del apartado “Ejemplos” de esta sentencia se muestran estas normas.
- Aunque la sentencia CONNECT se puede utilizar para establecer o conmutar conexiones, CONNECT con la cláusula USER/USING sólo se aceptará cuando no haya una conexión actual o inactiva al servidor indicado. La conexión debe liberarse antes de emitir una conexión con el mismo servidor con la cláusula 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. No obstante, en sistemas operativos Windows, las aplicaciones locales que se ejecutan con la cuenta de sistema local pueden conectarse implícitamente a la base de datos, con el ID de usuario SYSTEM.
- Cuando se establece conexión con Windows Server de forma explícita, el *nombre-autorización* o *variable-lenguaje-principal* de usuario puede especificarse utilizando el nombre compatible con el Administrador de cuentas de seguridad (SAM) de Microsoft Windows. El calificador debe ser un nombre de estilo NetBIOS, que tiene una longitud máxima de 15 bytes. Por ejemplo, 'Dominio\Usuario'.
- ***Terminación de una conexión***: Cuando se termina una conexión y una transacción aún no se ha confirmado o retrotraído, consulte el siguiente apartado "Utilización de CONNECT, CONNECT RESET implícitas y Desconexión" para obtener más información acerca de lo que les sucede a estas transacciones. Para garantizar un comportamiento coherente, codifique una sentencia COMMIT o ROLLBACK en lugar de depender de la sentencia CONNECT.

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**:

Tipo 1	Tipo 2
Cada unidad de trabajo sólo puede establecer conexión con un servidor de aplicaciones.	Cada unidad de trabajo puede establecer conexión con múltiples servidores de aplicaciones.
Se debe confirmar o retrotraer la unidad de trabajo actual antes de permitir una conexión con otro servidor de aplicaciones.	No es necesario confirmar ni retrotraer la unidad de trabajo actual antes de conectar con otro servidor de aplicaciones.
La sentencia CONNECT establece la conexión actual. Las peticiones SQL posteriores se reenvían a esta conexión hasta que otra CONNECT la modifique.	Igual que CONNECT Tipo 1 si se establece la primera conexión. Si se conmuta a una conexión inactiva y SQLRULES está establecido en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.
Conectar con la conexión actual es válido y no modifica la conexión actual.	Igual que CONNECT de tipo 1 si se la opción de precompilador SQLRULES se establece en DB2. Si SQLRULES está establecida en STD, debe utilizarse la sentencia SET CONNECTION en su lugar.

Tipo 1	Tipo 2
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.

Tipo 1	Tipo 2
Tras utilizar CONNECT RESET para desconectar la conexión actual, si la siguiente sentencia de SQL no es una sentencia CONNECT, realizará una conexión implícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.	CONNECT RESET equivale a una conexión explícita con el servidor de aplicaciones por omisión si hay uno definido en el sistema.
Es un error emitir sentencias CONNECT RESET consecutivas.	SÓLO es un error emitir sentencias CONNECT RESET consecutivas si se ha especificado SQLRULES(STD), porque esta opción inhabilita el uso de CONNECT para la conexión existente.
CONNECT RESET retrotrae implícitamente la unidad de trabajo actual.	CONNECT RESET retrotrae implícitamente la unidad de trabajo actual.
Si el sistema desconecta una conexión existente por cualquier motivo, las sentencias de SQL posteriores que no sean CONNECT para esta base de datos recibirán un SQLSTATE de 08003.	Si el sistema desconecta una conexión existente, siguen estando permitidas las sentencias COMMIT, ROLLBACK y SET CONNECTION.
La unidad de trabajo se confirmará implícitamente cuando el proceso de aplicación termine de manera satisfactoria.	Igual que el Tipo 1.
Todas las conexiones (sólo una) se desconectan cuando el proceso de aplicación termina.	Todas las conexiones (actuales, inactivas y las marcadas como pendientes de liberación) se desconectan cuando el proceso de aplicación termina.

Anomalías de CONNECT:

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	• Sin sentencia	• Ninguno	• Ninguno	• Ninguno

Secuencia	Sentencia	Servidor actual	Conexiones inactivas	Pendiente de liberación
1	• SELECT * FROM TBLA	• S0 (valor por omisión)	• Ninguno	• Ninguno
2	• CONNECT TO S1 • SELECT * FROM TBLB	• S1 • S1	• S0 • S0	• Ninguno • Ninguno
3	• CONNECT TO S2 • UPDATE TBLC SET ...	• S2 • S2	• S0, S1 • S0, S1	• Ninguno • Ninguno
4	• CONNECT TO S3 • SELECT * FROM TBLD	• S3 • S3	• S0, S1, S2 • S0, S1, S2	• Ninguno • Ninguno
5	• SET CONNECTION S2	• S2	• S0, S1, S3	• Ninguno
6	• RELEASE S3	• S2	• S0, S1	• S3
7	• COMMIT	• S2	• S0, S1	• Ninguno
8	• SELECT * FROM TBLE	• S2	• S0, S1	• Ninguno
9	• DISCONNECT S1 • SELECT * FROM TBLF	• S2 • S2	• S0 • S0	• Ninguno • Ninguno

CREATE ALIAS

La sentencia CREATE ALIAS define un alias para una tabla, una vista, un apodo u otro alias.

Invocación

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

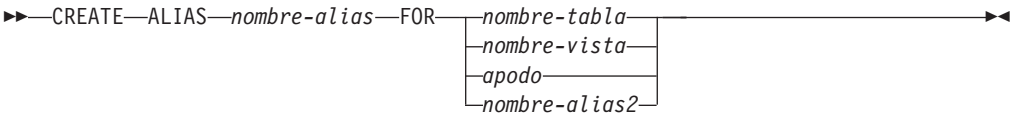
Autorización

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

- 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
- Autorización SYSADM o DBADM

Los privilegios requeridos para utilizar el objeto al que se hace referencia mediante su alias son idénticos a los privilegios requeridos para utilizar el objeto directamente.

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

- 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.
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para z/OS:
    - SYNONYM puede especificarse en lugar de ALIAS

Ejemplos

Ejemplo 1: HEDGES intenta crear un alias para una tabla T1 (ambos sin calificar).

```
CREATE ALIAS A1 FOR T1
```

Se crea el alias HEDGES.A1 para HEDGES.T1.

Ejemplo 2: HEDGES intenta crear un alias para una tabla (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

Se crea el alias HEDGES.A1 para MCKNIGHT.T1.

Ejemplo 3: HEDGES intenta crear un alias para una tabla (el alias en un esquema diferente; HEDGES no es DBADM; HEDGES no tiene el privilegio CREATEIN para el esquema MCKNIGHT).

```
CREATE ALIAS  MCKNIGHT.A1 FOR MCKNIGHT.T1
```

Este ejemplo falla (SQLSTATE 42501).

Ejemplo 4: HEDGES intenta crear un alias para una tabla no definida (ambos calificados; FUZZY.WUZZY no existe).

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

Esta sentencia resulta satisfactoria pero emite un mensaje de aviso (SQLSTATE 01522).

Ejemplo 5: HEDGES intenta crear un alias para un alias (ambos calificados).

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

La primera sentencia resulta satisfactoria (como en el ejemplo 2).

La segunda sentencia es satisfactoria y crea una cadena de alias, según la cual HEDGES.A2 hace referencia a HEDGES.A1, que a su vez hace referencia a MCKNIGHT.T1. Observe que no importa si HEDGES tiene privilegios o no sobre MCKNIGHT.T1. El alias se crea sean cuales sean los privilegios de las tablas.

Ejemplo 6: Designar A1 como alias para el apodo FUZZYBEAR.

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

Ejemplo 7: Una gran organización tiene un departamento financiero con el número D108 y un departamento de personal con el número D577. D108 mantiene determinada información en una tabla que reside en un RDBMS de DB2. D577 conserva determinados registros en una tabla que reside en Oracle RDBMS. Un DBA define los dos RDBMS como fuentes de datos de un sistema federado y asigna a las tablas los apodos de DEPTD108 y DEPTD577, respectivamente. El usuario de un sistema federado necesita crear vínculos entre estas tablas, pero preferiría hacer referencia a ellas por los nombres que tienen más sentido que sus apodos alfanuméricos. Por lo tanto, el usuario define FINANCE como un alias para DEPTD108 y PERSONNEL como un alias para DEPTD577.

```
CREATE ALIAS FINANCE FOR DEPTD108
CREATE ALIAS PERSONNEL FOR DEPTD577
```

CREATE AUDIT POLICY

La sentencia CREATE AUDIT POLICY define una política de comprobación en el servidor actual. La política determina las categorías que han de comprobarse; después puede aplicarse a otros objetos de base de datos para determinar el modo en que van a comprobarse dichos objetos.

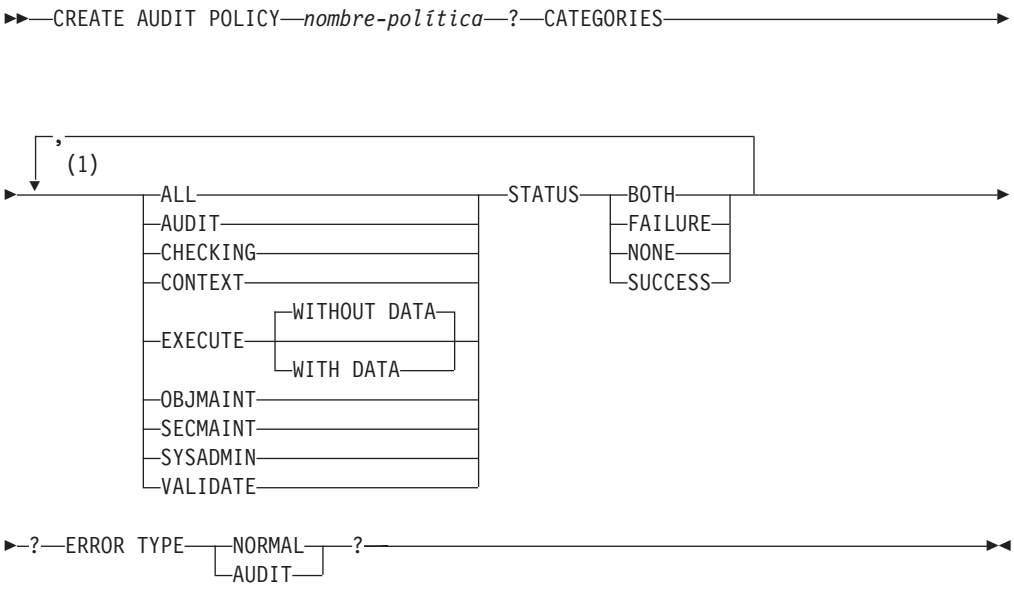
Invocación

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

Autorización

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

Sintaxis



Notas:

1 Cada categoría puede especificarse una vez como máximo (SQLSTATE 42614) y no puede especificarse ninguna otra categoría si se especifica ALL (SQLSTATE 42601).

Descripción

nombre-política

Asigna un nombre a la política de comprobación. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-política* no debe identificar una política de comprobación que ya esté descrita en el catálogo (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

CATEGORIES
Una lista de una o más categorías de comprobación para las que se especifica un estado. Si no se especifica ALL, el STATUS de cualquier categoría que no se especifique explícitamente se establece en NONE.

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

AUDIT
Genera registros cuando se han cambiado valores de comprobación o cuando se accede al archivo de anotaciones cronológicas de auditoría

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

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

EXECUTE
Genera registros para mostrar la ejecución de sentencias de SQL.

WITHOUT DATA o WITH DATA
Especifica si se proporcionan valores de datos de entrada para las variables de lenguaje principal y deben registrarse marcadores de parámetro como parte de la categoría EXECUTE.

WITHOUT DATA
Se proporcionan valores de datos de entrada para las variables de lenguaje principal y no se registran marcadores de parámetro como parte de la categoría EXECUTE. WITHOUT DATA es el valor por omisión.

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

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

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

SYSADMIN
Genera registros cuando se realicen operaciones que necesitan autorización SYSADM, SYSMAINT o SYSCTRL.

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

STATUS
Especifica un estado para la categoría especificada.

BOTH
Se comprobarán los sucesos satisfactorios y anómalos.

FAILURE
Sólo se comprobarán los sucesos anómalos.

SUCCESS
Sólo se comprobarán los sucesos satisfactorios.

NONE
No se comprobarán los sucesos de esta categoría.

ERROR TYPE
Especifica si van a devolverse o ignorarse los errores de comprobación.

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

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

Normas

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

Notas

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

Ejemplo

Cree una política de comprobación para comprobar éxitos y anomalías para las categorías AUDIT y OBJMAINT; sólo las anomalías para las categorías SECMAINT, CHECKING y VALIDATE y ningún suceso para las demás categorías.

CREATE AUDIT POLICY

```
CREATE AUDIT POLICY DBAUDPRF
  CATEGORIES AUDIT    STATUS BOTH,
             SECMAINT STATUS FAILURE,
             OBJMAINT STATUS BOTH,
             CHECKING STATUS FAILURE,
             VALIDATE STATUS FAILURE
  ERROR TYPE NORMAL
```

CREATE BUFFERPOOL

La sentencia CREATE BUFFERPOOL define una nueva agrupación de almacenamientos intermedios 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 de base de datos, existiendo la posibilidad de alterar temporalmente el tamaño de particiones de base de datos específicas. Asimismo, en una base de datos particionada, la agrupación de almacenamientos intermedios se define en todas las particiones de base de datos 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 de base de datos que se encuentran en esos grupos de particiones de base de datos.

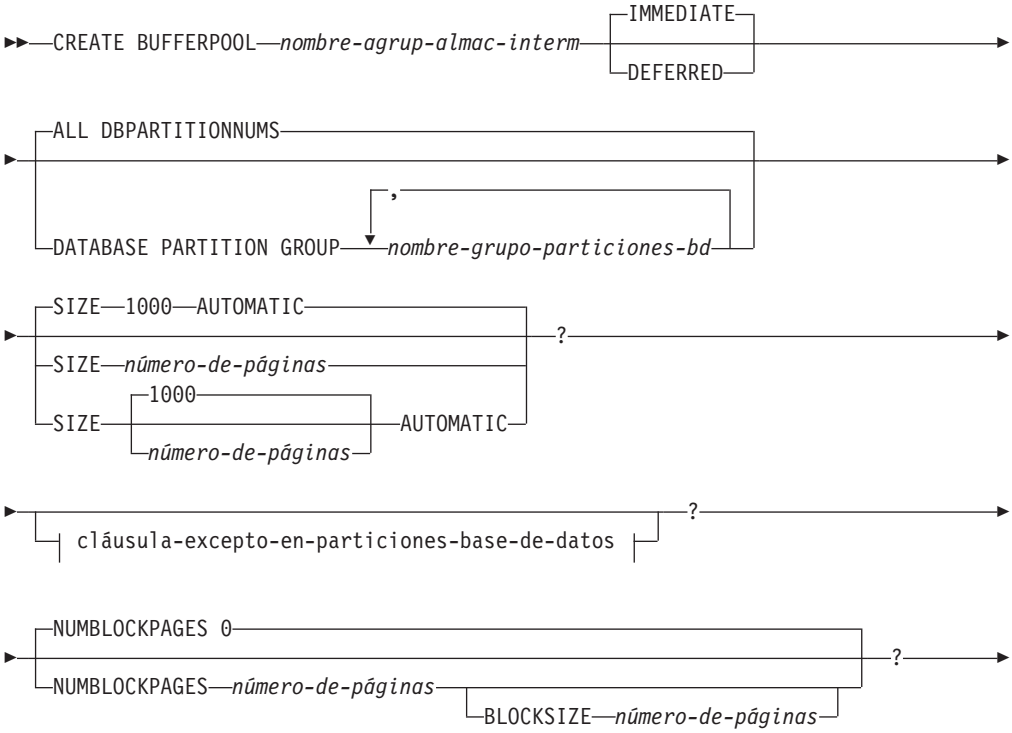
Invocación

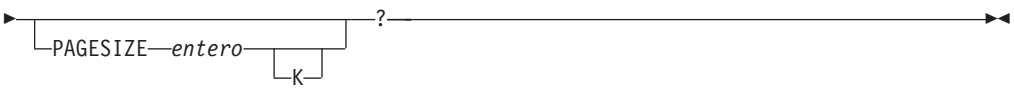
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

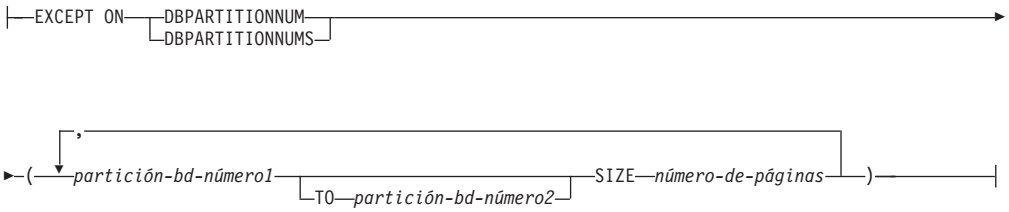
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSCTRL o SYSADM.

Sintaxis





cláusula-excepto-en-particiones-base-de-datos:



Descripción

nombre-agrup-almac-interm
Indica el nombre de la agrupación de almacenamientos intermedios. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-agrup-almac-interm* no debe identificar ninguna agrupación de almacenamientos intermedios que ya exista en el catálogo (SQLSTATE 42710). El *nombre-agrup-almac-interm* no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

IMMEDIATE o DEFERRED
Indica si la agrupación de almacenamientos intermedios se creará de forma inmediata.

IMMEDIATE
La agrupación de almacenamientos intermedios se creará inmediatamente. Si no hay espacio reservado suficiente en la memoria compartida de la base de datos para asignar la nueva agrupación de almacenamientos intermedios (SQLSTATE 01657), la sentencia se ejecutará como DEFERRED.

DEFERRED
La agrupación de almacenamientos intermedios se creará cuando se desactive la base de datos (es necesario desconectar todas las aplicaciones de la base de datos). No es necesario espacio de memoria reservado; DB2 asignará la memoria necesaria del sistema.

ALL DBPARTITIONNUMS
Esta agrupación de almacenamientos intermedios se creará en todas las particiones de base de datos de la base de datos.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd, ...*
Identifica el grupo o grupos de particiones de base de datos a los que se aplica la definición de agrupación de almacenamientos intermedios. Si se especifica este parámetro, la agrupación de almacenamientos intermedios sólo se creará en las particiones de base de datos de estos 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 especifica la cláusula DATABASE PARTITION GROUP, esta agrupación de almacenamientos intermedios se creará en todas las particiones de bases de datos (y en todas las particiones de bases de datos que se añadan posteriormente a la base de datos).

SIZE
Especifica el tamaño de la agrupación de almacenamientos intermedios. Para

una base de datos particionada, éste será el tamaño por omisión para todas las particiones de base de datos en las que exista la agrupación de almacenamientos intermedios. El valor por omisión es 1.000 páginas.

número-de-páginas
Número de páginas de la nueva agrupación de almacenamientos intermedios.

AUTOMATIC
Habilita el ajuste automático para la agrupación de almacenamientos intermedios. El gestor de bases de datos ajusta el tamaño de la agrupación de almacenamientos intermedios como respuesta a los requisitos de carga de trabajo. El número de páginas implícito o explícito especificado se utiliza como tamaño inicial de la agrupación de almacenamientos intermedios.

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

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 de base de datos para las que el tamaño de la agrupación de almacenamientos intermedios será diferente del valor por omisión. Si no se especifica esta cláusula, todas las particiones de base de datos 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 de base de datos concretas. DBPARTITIONNUM es sinónimo de DBPARTITIONNUMS.

partición-bd-núm-1
Especifica un número de partición de base de datos que se incluye en las particiones de base de datos para las que se ha creado la agrupación de almacenamientos intermedios.

TO *partición-bd-número2*
Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9). Todas las particiones de base de datos que se encuentren entre los números de partición de base de datos especificados (ambos inclusive, deben incluirse en las particiones de base de datos 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 o 32.768. Los valores válidos de *entero* con el sufijo K son 4, 8, 16 ó 32. Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio. Si el tamaño de página no es uno de estos valores, se devuelve un error (SQLSTATE 428DE).

El valor por omisión se proporciona mediante el parámetro de configuración de base de datos **tamaño-página**, que se establece cuando se crea la base de datos.

Notas

- Si la agrupación de almacenamientos intermedios se crea mediante la utilización de la opción DEFERRED, cualquier espacio de tablas que se cree en esta agrupación de almacenamientos intermedios utilizará una pequeña agrupación de almacenamientos intermedios del sistema con el mismo tamaño de página hasta la próxima vez que se active la base de datos. La base de datos deberá reiniciarse para que la agrupación de almacenamientos intermedios se active y para que entren en vigor las asignaciones de espacio de tablas para la nueva agrupación de almacenamientos intermedios. La opción por omisión es IMMEDIATE.
- Debe haber suficiente memoria real en la máquina para el total de las agrupaciones de almacenamientos intermedios, así como para el resto de necesidades de espacio del gestor de bases de datos y de las aplicaciones. Si DB2 no puede obtener memoria para las agrupaciones de almacenamientos intermedios normales, intentará iniciarse con 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 tablas utilizarán las agrupaciones de almacenamientos intermedios del sistema.
- ***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

CREATE DATABASE PARTITION GROUP

La sentencia CREATE DATABASE PARTITION GROUP define un grupo nuevo de particiones de base de datos, asigna particiones de base de datos en el grupo de particiones de base de datos y registra la definición del grupo de particiones de base de datos en el catálogo del sistema.

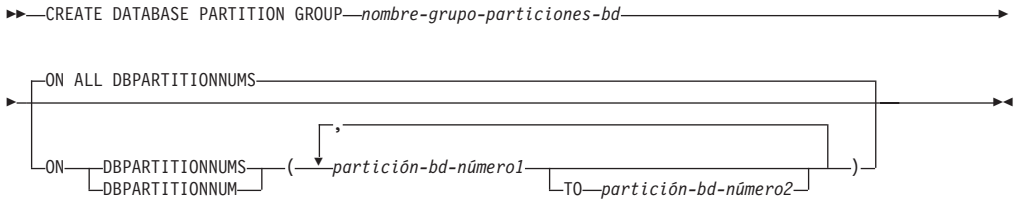
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-grupo-particiones-bd
Indica el nombre del grupo de particiones de base de datos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo-particiones-bd* no debe identificar un grupo de particiones de base de datos que ya exista en el catálogo (SQLSTATE 42710). El *nombre-grupo-particiones-bd* no debe empezar por los caracteres 'SYS' o 'IBM' (SQLSTATE 42939).

ON ALL DBPARTITIONNUMS
Especifica que el grupo de particiones de base de datos está definido en todas las particiones de base de datos definidas para la base de datos (archivo `db2nodes.cfg`) cuando se crea el grupo de particiones de base de datos.

Si se añade una partición de base de datos al sistema de la base de datos, debe emitirse la sentencia ALTER DATABASE PARTITION GROUP para que incluya esta nueva partición de base de datos en un grupo de particiones de base de datos (incluido IBMDEFAULTGROUP). Además, debe emitirse el mandato REDISTRIBUTE DATABASE PARTITION GROUP para mover datos a la partición de base de datos.

ON DBPARTITIONNUMS
Especifica las particiones de base de datos que se encuentran en el grupo de particiones de base de datos. DBPARTITIONNUM es sinónimo de DBPARTITIONNUMS.

partición-bd-núm-1
Especifique un número de partición de base de datos. (Para mantener la compatibilidad con la versión anterior, puede especificarse un *nombre-nodo* en el formato NODE*nnnnn*.)

TO *partición-bd-número2*
Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9). Todas las particiones de base de datos entre los números de partición de base de datos especificados e incluidos éstos se encuentran en el grupo de particiones de base de datos.

Normas

- Cada partición de base de datos especificada por número debe estar definida en el archivo db2nodes.cfg (SQLSTATE 42729).
- Cada *número-partición-base-de-datos* de la lista de la cláusula ON DBPARTITIONNUMS deberá aparecer una vez como máximo (SQLSTATE 42728).
- Un *número-partición-base-de-datos* válido es un número comprendido entre 0 y 999, ambos inclusive (SQLSTATE 42729).

Notas

- Esta sentencia crea una correlación de distribución para el grupo de particiones de base de datos. Para cada correlación de distribución, se genera un identificador de correlaciones de distribución (PMAP_ID). Esta información se graba en el catálogo y puede recuperarse desde SYSCAT.DBPARTITIONGROUPS y SYSCAT.PARTITIONMAPS. Cada entrada de la correlación de distribución especifica la partición de base de datos de destino en la que residen todas las filas generadas aleatoriamente. Para un grupo de particiones de base de datos de una sola partición, la correlación de distribución correspondiente sólo tiene una entrada. Para un grupo de particiones de base de datos de varias particiones, la correlación de distribución correspondiente tiene 4096 entradas, donde los números de partición de base de datos se asignan a las entradas de correlación de forma rotativa por omisión.
- ***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

Ejemplos

Supongamos que dispone de una base de datos particionada con seis particiones de base de datos 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 de base de datos. 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 denominada MEDGROUP en las particiones de base de datos 0, 1, 2, 5 y 8. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- Supongamos que desea crear un grupo de particiones de base de datos de una sola partición MINGROUP en la partición 7 de la base de datos. La sentencia es la siguiente:

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

CREATE EVENT MONITOR

La sentencia CREATE EVENT MONITOR define un supervisor que registrará ciertos sucesos que se produzcan cuando se utilice la base de datos. La definición de cada supervisor de sucesos especifica también el lugar donde la base de datos debe registrar los sucesos.

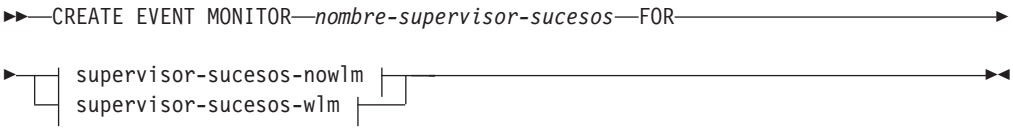
Invocación

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

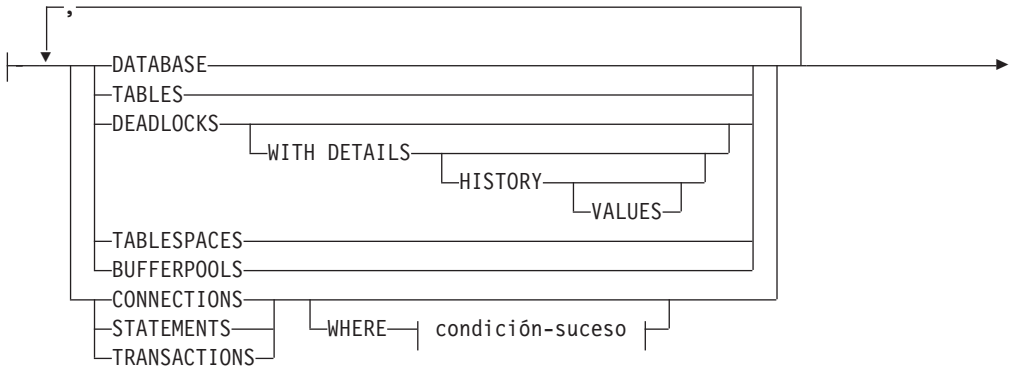
Autorización

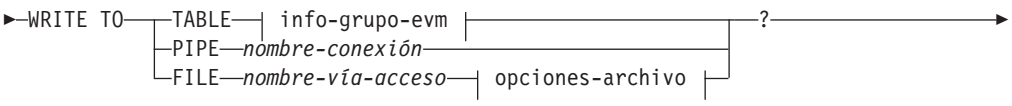
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM.

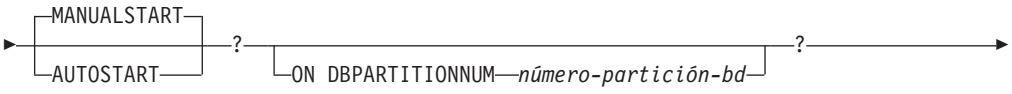
Sintaxis



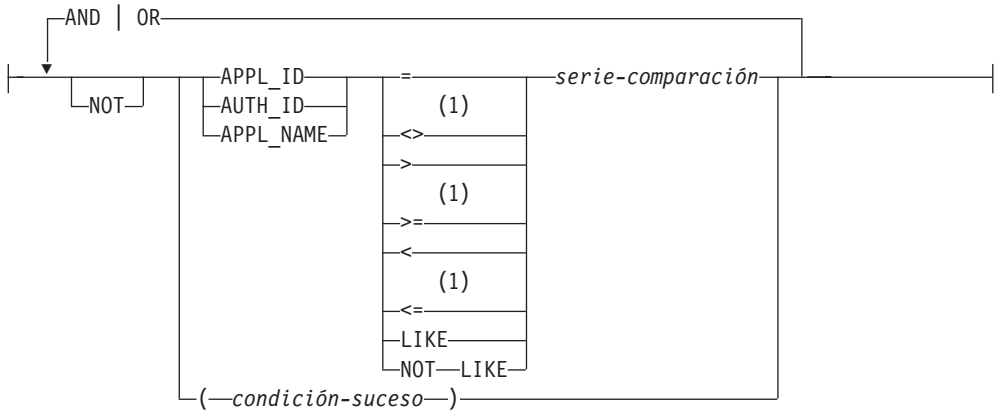
supervisor-sucesos-nowlm:



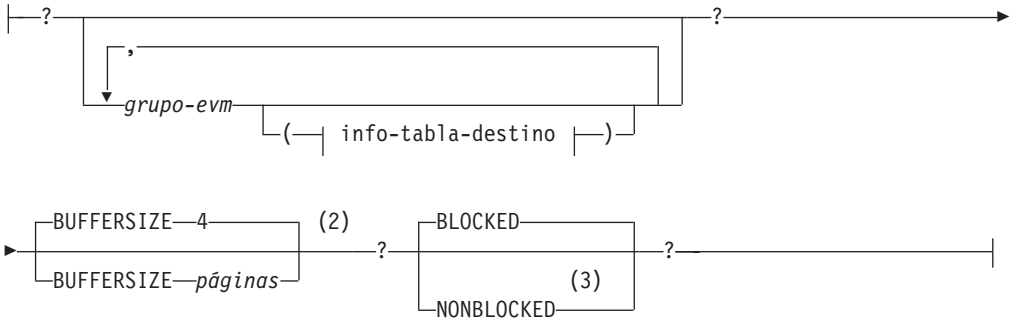




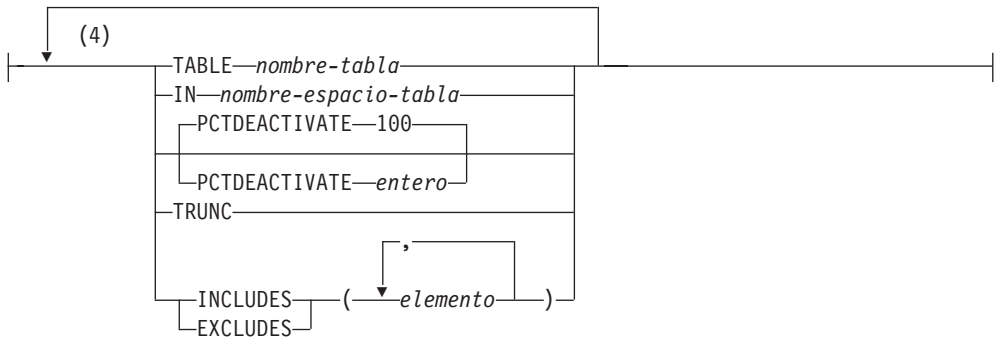
condición-suceso:



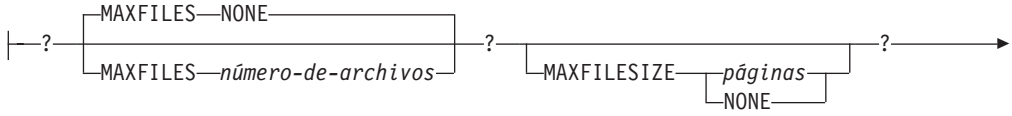
info-grupo-evm:

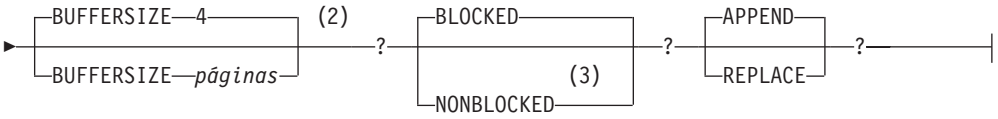


info-tabla-destino:

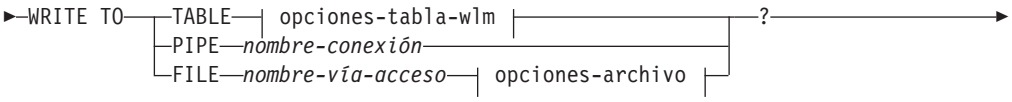


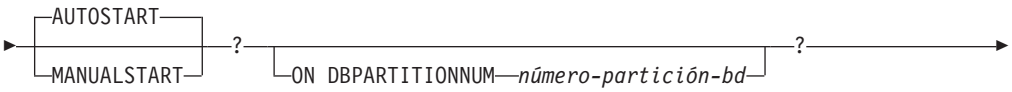
opciones-archivo:



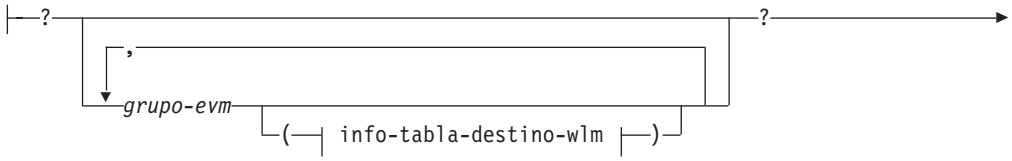


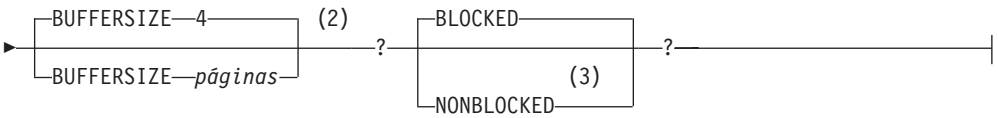
supervisor-sucesos-wlm:



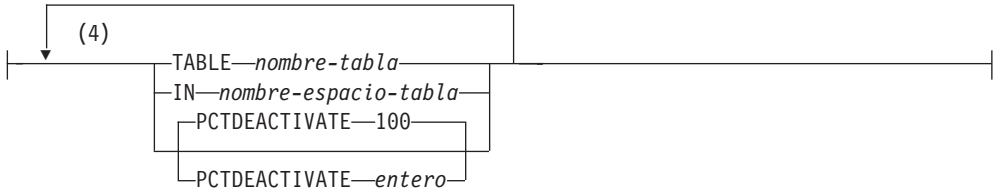


opciones-tabla-wlm:





info-tabla-destino-wlm:



Notas:

1 También se da soporte a otras formas de estos operadores.

2 La cláusula BUFFERSIZE sólo es aplicable al supervisor de sucesos de estadísticas, al supervisor de sucesos de violaciones de umbral y a supervisores de sucesos de gestión que no son de carga de trabajo (WLM) (SQLSTATE 42613).

3 La palabra clave NONBLOCKED sólo es aplicable al supervisor de sucesos de estadísticas, al supervisor de sucesos de violaciones de umbral y a supervisores de sucesos que no son WLM (SQLSTATE 42613).

4 Cada cláusula sólo puede especificarse una vez.

Descripción

nombre-supervisor-sucesos
Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-supervisor-sucesos* no debe identificar ningún supervisor de sucesos que ya exista en el catálogo (SQLSTATE 42710).

FOR
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. Para tablas particionadas, se registra un suceso de tabla para cada partición de datos de cada tabla activa. Una tabla activa es una tabla que ha cambiado desde la primera conexión con la base de datos.

DEADLOCKS
Especifica que el supervisor de sucesos registre un suceso de punto muerto siempre que se produzca un punto muerto.

WITH DETAILS
Especifica que el supervisor de sucesos debe generar un suceso de conexión de punto muerto más detallado para cada aplicación que esté implicada en un punto muerto. Estos detalles adicionales incluyen:

- Información acerca de la sentencia que la aplicación estaba ejecutando cuando se ha producido el punto muerto, como, por ejemplo, el texto de la sentencia.
- Los bloqueos que mantenía la aplicación al producirse el punto muerto. En un entorno de bases de datos particionadas, sólo se incluyen los bloqueos mantenidos en la partición de base de datos en la que la aplicación estaba esperando su bloqueo cuando se ha producido el punto muerto. Para las tablas particionadas, ello incluye el identificador de la partición de datos.

HISTORY
Especifica que los datos del supervisor de sucesos también incluirán:

- El histórico de todas las sentencias de la unidad de trabajo actual en el nodo participante (incluidos los cursores WITH HOLD abiertos en unidades de trabajo anteriores). Las sentencias SELECT emitidas en el nivel de aislamiento UR (lectura no confirmada) no se incluyen en el histórico de sentencias.
- El entorno de compilación de sentencias para cada sentencia de SQL en formato binario (si está disponible).

VALUES
Especifica que los datos del supervisor de sucesos también incluirán:

- Los valores de datos utilizados como variables de entrada para cada sentencia de SQL. Estos valores no incluirán datos LOB, datos largos, datos de tipo estructurado ni datos XML.

En una única sentencia CREATE EVENT MONITOR, puede especificar sólo uno de estos elementos: DEADLOCKS, DEADLOCKS WITH DETAILS, DEADLOCKS WITH DETAILS HISTORY o DEADLOCKS WITH DETAILS HISTORY VALUES (SQLSTATE 42613).

TABLESPACES
Especifica que el supervisor de sucesos registre un suceso de espacio de tablas para cada espacio de tablas cuando se desconecte la última aplicación de la base de datos.

BUFFERPOOLS
Especifica que el supervisor de sucesos registre un suceso de agrupación de almacenamientos intermedios cuando se desconecte la última aplicación de la base de datos.

ACTIVITIES
Especifica que el supervisor de sucesos registra un suceso de actividad cuando una actividad termina de ejecutarse o antes de completarse la ejecución si el procedimiento CAPTURE_ACTIVITY_IN_PROGRESS activa el suceso. La actividad debe:

- Pertenecer a una clase de servicio o carga de trabajo que tenga el conjunto COLLECT ACTIVITY DATA, o
- Pertenecer a una clase de trabajo cuya acción de trabajo asociada sea COLLECT ACTIVITY DATA, o
- Identificarse como la actividad que ha violado el umbral cuya cláusula COLLECT ACTIVITY DATA estaba especificada, o
- Haberse identificado en una llamada al procedimiento CAPTURE_ACTIVITY_IN_PROGRESS antes de completarse

THRESHOLD VIOLATIONS
Especifica que el supervisor de sucesos registra un suceso de violación de umbral cuando se viola un umbral. Dichos sucesos pueden registrarse en cualquier punto de la vida de una actividad y no sólo al completarse.

STATISTICS
Especifica que el supervisor de sucesos registra un suceso de clase de trabajo, carga de trabajo o clase de servicio:

- Cada *periodo* minutos, donde *periodo* es el valor del parámetro de configuración de base de datos **wlm_collect_int**
- Cuando se llama al procedimiento wlm_collect_stats

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-suceso
Define un filtro que determina qué conexiones hacen que se produzca un suceso de CONNECTION, STATEMENT o TRANSACTION. Si el resultado de la condición de suceso es TRUE para una conexión en particular, dicha conexión generará los sucesos pedidos.

Esta cláusula es una forma especial de la cláusula WHERE que no debe confundirse con una condición de búsqueda estándar.

Para determinar si una aplicación generará los sucesos para un supervisor de sucesos en particular, se evalúa la cláusula WHERE:

- Para cada conexión activa cuando se activa un supervisor de sucesos por primera vez
- Posteriormente para cada conexión nueva con la base de datos en el tiempo de conexión

La cláusula WHERE no se evalúa para cada suceso.

Si no se ha especificado ninguna cláusula WHERE, se supervisarán todos los sucesos del tipo de suceso especificado.

La condición-suceso no debe exceder una longitud de 32.678 bytes en la página de códigos de base de datos (SQLSTATE 22001).

APPL_ID
Especifica que el ID de aplicación para cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

AUTH_ID
Especifica que el ID de autorización de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

APPL_NAME
Especifica que el nombre de programa de aplicación de cada conexión debe compararse con la *serie-comparación* para determinar si la conexión debe generar sucesos de CONNECTION, STATEMENT o TRANSACTION (lo que se haya especificado).

El nombre de programa de aplicación son los 20 primeros bytes del nombre de archivo del programa de aplicación, después del último separador de la vía de acceso.

serie-comparación
Es una serie de caracteres que debe compararse con el APPL_ID, AUTH_ID o APPL_NAME de cada aplicación que se conecta con la base de datos. *serie-comparación* debe ser una constante de serie (es decir, las variables del lenguaje principal y otras expresiones de serie no están permitidas).

WRITE TO
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.

info-grupo-evm
Define la tabla de destino de un grupo de datos lógicos. Esta cláusula debe especificarse para cada agrupación que deba registrarse. Sin embargo, si no se especifica ninguna cláusula info-grupo-evm, se registran todos los grupos del tipo de supervisor de sucesos.

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

Tipo de supervisor de sucesos	Valor 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]
Puntos muertos con detalles e histórico	• CONNHEADER • DEADLOCK • DLCONN[2] • DLLOCK[3] • STMTHIST • CONTROL[1]
Puntos muertos con detalles, histórico y valores	• CONNHEADER • DEADLOCK • DLCONN[2] • DLLOCK[3] • STMTHIST • STMTVALS • CONTROL[1]

Tipo de supervisor de sucesos	Valor grupo-evm
Espacios de tablas	• TABLESPACE • CONTROL[1]
Agrupaciones de almacenamientos intermedios	• BUFFERPOOL • CONTROL[1]
Conexiones	• CONNHEADER • CONN • CONTROL[1] • CONNMEMUSE
Sentencias	• CONNHEADER • STMT • SUBSECTION[4] • CONTROL[1]
Transacciones	• CONNHEADER • XACT • CONTROL[1]
Actividades	• ACTIVITY • ACTIVITYSTMT • ACTIVITYVALS • CONTROL
Estadísticas	• QSTATS • SCSTATS • WCSTATS • WLSTATS • HISTOGRAMBIN • CONTROL
Violaciones de umbral	• THRESHOLDVIOLATIONS • CONTROL

[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 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. La tabla de destino debe ser una tabla no particionada. Si el nombre es un nombre no calificado, el esquema de tabla toma por omisión el 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 CONCAT "_"
  CONCAT nombre-supervisor-sucesos,1,128)
```

IN *nombre-espacio-tablas*
Define el espacio de tablas en el que va a crearse la tabla. Si no se proporciona ningún nombre de espacio de tablas, 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
 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 tablas SMS.

Se recomienda que cuando un espacio de tablas de destino tenga el redimensionamiento automático habilitado el valor de PCTDEACTIVATE debe establecerse en 100.

TRUNC
Especifica que las columnas STMT_TEXT y STMT_VALUE_DATA están definidas como VARCHAR(*n*), donde *n* es el tamaño mayor que puede caber en la fila de la tabla. En este caso, los datos que tengan más de *n* bytes se truncarán. En el ejemplo siguiente se muestra cómo se calcula el valor de *n*. Supongamos que:

- La tabla se ha creado en un espacio de tablas que utiliza páginas de 32K.
- La longitud total de todas las demás columnas de la tabla es de 357 bytes.

En este caso, el tamaño de fila máximo para la tabla es de 32677 bytes. Por lo tanto, el elemento se definirá como VARCHAR(32316); es decir, 32677 - 357 - 4. Si no se especifica TRUNC, la columna se definirá como CLOB(2M).

Tenga en cuenta que STMT_TEXT se encuentra en el grupo de sucesos STMT, en el grupo de sucesos STMT_HISTORY y en el grupo de sucesos DLCONN (para los supervisores de sucesos de puntos muertos con detalles). STMT_VALUE_DATA se encuentra en el grupo de sucesos DATA_VALUE.

INCLUDES
Especifica que los elementos siguientes van a incluirse en la tabla.

EXCLUDES
Especifica que los elementos siguientes *no* van a incluirse en la tabla.

elemento
Identifica un elemento del supervisor. La información de los elementos puede proporcionarse de una de las formas siguientes:

- No especificar información de los elementos. En este caso, todos los elementos se incluyen en la sentencia CREATE TABLE.
- Especificar los elementos que deben incluirse en el formato: INCLUDES (elemento1, elemento2, ..., elemento*n*). Para estos elementos, sólo se crean columnas de tabla.
- Especificar los elementos que deben excluirse en el formato: EXCLUDES (elemento1, elemento2, ..., elemento*n*). Sólo se crean columnas de tabla para todos los elementos que no se han indicado aquí.

Utilice el mandato db2evtbl para crear una sentencia CREATE EVENT MONITOR que incluya una lista completa de elementos para un grupo.

BUFFERSIZE *páginas*
Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Los supervisores de sucesos de tabla insertan todos los datos de un almacenamiento intermedio y emiten COMMIT cuando se ha procesado el almacenamiento intermedio. Cuanto más grandes sean los almacenamientos intermedios, mayor será el ámbito de confirmación utilizado por el supervisor de sucesos. Los supervisores de sucesos altamente activos deben tener almacenamientos intermedios mayores que los supervisores de sucesos relativamente inactivos. Cuando se inicia un supervisor, se asignan dos almacenamientos intermedios del tamaño especificado. Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esa pila. Si se utilizan muchos supervisores de

sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

BLOCKED
Especifica que cada agente que genera un suceso debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Debe seleccionarse BLOCKED para garantizar que no se van a perder datos. Es la opción por omisión.

NONBLOCKED
Especifica que cada agente que genera un suceso no debe esperar a que se grabe en disco un almacenamiento intermedio de sucesos si el agente determina que ambos almacenamientos intermedios de sucesos están llenos. Los supervisores de sucesos NONBLOCKED no hacen que las operaciones vayan más despacio como los supervisores de sucesos BLOCKED. Sin embargo, los supervisores de sucesos NONBLOCKED están sujetos a la pérdida de datos en sistemas muy activos.

PIPE
Especifica que el destino para los datos del supervisor de sucesos es una conexión con nombre. El supervisor de sucesos graba los datos en la conexión en una sola corriente (es decir, como si fuera un solo archivo infinitamente largo). Cuando se graban los datos en una conexión, el supervisor de sucesos no realiza grabaciones por bloques. Si no hay espacio en el almacenamiento intermedio de conexión, el supervisor de sucesos desechará los datos. Es responsabilidad de la aplicación supervisora el leer los datos pronto si desea asegurar que no se pierdan datos.

nombre-conexión
El nombre de la conexión (FIFO en AIX) en la que el supervisor de sucesos grabará los datos.

Las normas de denominación para las conexiones son específicas de las plataformas. En sistemas operativos UNIX, los nombres de conexiones se tratan como nombres de archivo. Como resultado, están permitidos los nombres de conexiones relativos y se tratan como nombres-vía relativas (consulte *nombre-vía* más abajo). En Windows, no obstante, existe una sintaxis especial para un nombre de conexión y, como consecuencia de ello, se necesitan nombres de conexión absolutos.

La existencia de una conexión no se comprobará en el tiempo de creación del supervisor de sucesos. Es responsabilidad de la aplicación supervisora haber creado y abierto la conexión para lectura en el momento en que se activa el supervisor de sucesos. Si la conexión no está disponible en ese momento, el supervisor de sucesos se desactivará por sí solo y anotará cronológicamente un error. (Es decir, si se ha activado el supervisor de sucesos en el momento del inicio de la base de datos como resultado de la opción AUTOSTART, el supervisor de sucesos anotará un error en el registro cronológico de errores del sistema.) Si se activa el supervisor de sucesos mediante la sentencia SET EVENT MONITOR STATE SQL, dicha sentencia fallará (SQLSTATE 58030).

FILE
Indica que el destino para los datos del supervisor de sucesos es un

archivo (o conjunto de archivos). El supervisor de sucesos graba la corriente de datos como una serie de archivos numerados de 8 caracteres, con la extensión "evt". (por ejemplo, 00000000.evt, 00000001.evt y 00000002.evt). Los datos deben considerarse un archivo lógico incluso cuando se dividen los datos en fragmentos más pequeños (es decir, el inicio de la corriente de datos es el primer byte del archivo 00000000.evt; el final de la corriente de datos es el último byte del archivo nnnnnnnn.evt).

El tamaño máximo de cada archivo se puede definir también como el número máximo de archivos. Un supervisor de sucesos no dividirá nunca un solo registro de sucesos entre dos archivos. Sin embargo, el supervisor de sucesos puede grabar registros relacionados en dos archivos distintos. Es responsabilidad de la aplicación que utiliza estos datos realizar un seguimiento de dicha información relacionada cuando procese los archivos de sucesos.

nombre-vía-acceso
El nombre del directorio en el que el supervisor de sucesos debe grabar los datos de los archivos de sucesos. La vía de acceso debe ser conocida en el servidor; no obstante, la vía de acceso propiamente dicha puede residir en otra partición de base de datos (por ejemplo, en un sistema UNIX, puede ser un archivo montado NFS). Se debe utilizar una constante de serie cuando se especifica el *nombre-vía*.

El directorio no tiene que existir en el momento de CREATE EVENT MONITOR. Sin embargo, se realiza una comprobación de la existencia de la vía de acceso de destino cuando se activa el supervisor de sucesos. En este momento, si la vía de acceso de destino no existe, se genera un error (SQLSTATE 428A3).

Si se especifica una vía de acceso absoluta (una vía de acceso que empieza por el directorio raíz en AIX, o por un identificador de disco en Windows), se utilizará la vía de acceso especificada. 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.

Es posible especificar dos o más supervisores de sucesos que tengan la misma vía de acceso de destino. Sin embargo, cuando uno de los supervisores de sucesos se ha activado por primera vez, y siempre que el directorio de destino no esté vacío, será imposible activar cualquiera de los supervisores de sucesos.

opciones-archivo
Especifica las opciones para el formato del archivo.

MAXFILES NONE
Especifica que no hay ningún límite en el número de archivos de sucesos que vaya a crear el supervisor de sucesos. Es el valor por omisión.

MAXFILES *número-de-archivos*
Especifica que hay un límite en el número de archivos del supervisor de sucesos que vayan a existir para un supervisor de sucesos en particular en cualquier momento. Siempre que un supervisor de sucesos tenga que crear otro archivo, comprobará que el número de archivos .evt del directorio sea inferior al *número-de-archivos*. Si ya se ha alcanzado este límite, el supervisor de sucesos se desactivará por sí solo.

Si una aplicación elimina los archivos de sucesos del directorio después de haberlos grabado, el número total de archivos que un supervisor de sucesos puede producir puede exceder el *número-de-archivos*. Esta opción se ha proporcionado para permitir a un usuario garantizar que los datos de sucesos no consumirán más de una cantidad de espacio de disco especificada.

MAXFILESIZE *páginas*
Especifica que hay un límite en el tamaño de cada archivo del supervisor de sucesos. Siempre que un supervisor de sucesos grabe un nuevo registro de suceso en un archivo, comprueba que el archivo no vaya a crecer de manera que sobrepase las *páginas* (en unidades de páginas de 4K). Si el archivo resultante fuese a ser demasiado grande, entonces el supervisor de sucesos conmutará al siguiente archivo. El valor por omisión para esta opción es:

- Windows - 200 páginas de 4 K
- UNIX - 1000 páginas de 4K

El número de páginas debe ser mayor que el tamaño del almacenamiento intermedio de sucesos en páginas, como mínimo. Si no se cumple este requisito, se generará un error (SQLSTATE 428A4).

MAXFILESIZE NONE
Especifica que no hay ningún límite establecido en el tamaño de un archivo. Si se especifica MAXFILESIZE NONE, también debe especificarse MAXFILES 1. Esta opción significa que un archivo contendrá todos los datos de sucesos para un supervisor de sucesos en particular. En este caso el único archivo de sucesos será 00000000.evt.

BUFFERSIZE *páginas*
Especifica el tamaño de los almacenamientos intermedios del supervisor de sucesos (en unidades de páginas de 4K). Todas las E/S del archivo del supervisor de sucesos se almacenan temporalmente para mejorar el rendimiento de los supervisores de sucesos. Cuanto más grandes sean los almacenamientos intermedios, menos E/S realizará el supervisor de sucesos. Los supervisores de sucesos altamente activos deben tener almacenamientos intermedios mayores que los supervisores de sucesos relativamente inactivos. Cuando se inicia el supervisor, se asignan dos almacenamientos intermedios del tamaño especificado. Los supervisores de sucesos utilizan el doble de almacenamiento intermedio para permitir E/S asíncronas.

El tamaño por omisión de cada almacenamiento intermedio es de 4 páginas (se asignan dos almacenamientos intermedios de 16K). El tamaño mínimo es de 1 página. El tamaño máximo de los almacenamientos intermedios está limitado por el valor del parámetro MAXFILESIZE, así como por el tamaño de la pila del supervisor, ya que los almacenamientos intermedios se asignan desde esta pila. Si se utilizan muchos supervisores de sucesos a la vez, incremente el tamaño del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.

Los supervisores de sucesos que 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 debe activarse manualmente utilizando la sentencia SET EVENT MONITOR STATE. Después de que un supervisor de sucesos MANUALSTART se haya activado, sólo se puede desactivar utilizando la sentencia SET EVENT MONITOR STATE o mediante la detención de la instancia. Éste es el comportamiento por omisión de los supervisores de sucesos que no son WLM, a excepción del supervisor de sucesos DB2DETAILDEADLOCK.

AUTOSTART
Especifica que el supervisor de sucesos debe activarse automáticamente siempre que la partición de base de datos en la que se ejecuta el supervisor esté activada. Éste es el comportamiento por omisión de los supervisores de sucesos WLM y del supervisor de sucesos DB2DETAILDEADLOCK.

ON DBPARTITIONNUM *núm-partición-bd*
Especifica la partición de la base de datos en la que se debe ejecutar un supervisor de sucesos de conexión o archivo. Cuando el ámbito de supervisión se define como LOCAL, sólo se recopilan los datos en la partición especificada. Cuando el ámbito de supervisión se define como GLOBAL, todas las

particiones de base de datos recopilan datos e informan a la partición de base de datos que tiene el número 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.

Esta cláusula no es válida para supervisores de sucesos de 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 de bases de datos en las que se hayan definido espacios de tablas para tablas de destino.

Si no se especifica esta cláusula, se utiliza el número de partición (para la aplicación) de la base de datos conectada actualmente.

LOCAL
El supervisor de sucesos sólo informa acerca de la partición de base de datos que está en ejecución. Ofrece un rastreo parcial de la actividad de la base de datos. Éste es el valor por omisión.

Esta cláusula no es válida para supervisores de sucesos de tabla.

GLOBAL
El supervisor de sucesos informa acerca de todas las particiones de base de datos. Para una base de datos particionada, sólo los supervisores de sucesos DEADLOCKS pueden definirse como GLOBAL.

Esta cláusula no es válida para supervisores de sucesos de tabla.

Normas

- Cada uno de los tipos de sucesos (DATABASE, TABLES, DEADLOCK,...) sólo pueden especificarse una vez en la definición de un supervisor de sucesos en particular.

Notas

- Las definiciones del supervisor de sucesos se registran en la vista de catálogo SYSCAT.EVENTMONITORS. Los sucesos en sí se registran en la vista de catálogo SYSCAT.EVENTS. Los nombres de las tablas de destino se graban en la vista de catálogo SYSCAT.EVENTTABLES.
- Cuando se utiliza DEADLOCKS WITH DETAILS en lugar de DEADLOCKS, ello afecta al rendimiento. Cuando se produce un punto muerto, el gestor de bases de datos necesita más tiempo para grabar la información adicional del punto muerto.
- Normalmente, siempre que se establece una conexión se graba un suceso CONNHEADER. Sin embargo, si se ha creado un supervisor de sucesos sólo para DEADLOCKS WITH, sólo se grabará un suceso CONNHEADER la primera vez que la conexión participe en un punto muerto.
- En una base de datos con múltiples particiones de base de datos, la cláusula ON DBPARTITIONNUM puede utilizarse con los supervisores de sucesos FILE y PIPE que tengan un tipo de suceso DEADLOCKS para indicar dónde debe residir el supervisor de sucesos propiamente dicho; la información de las otras particiones de base de datos, si es relevante, se enviará a aquella ubicación para el proceso.
- En una base de datos con múltiples particiones de base de datos, un supervisor de sucesos de punto muerto recibirá información sobre las aplicaciones que tengan bloqueos que participen en el punto muerto desde todas las particiones de base de datos en las que existían dichos bloqueos participantes. Si la partición de base de datos a la que está conectada la aplicación (la partición coordinadora

de la aplicación) no es una de las particiones de base de datos participantes, no se recibirá información sobre ningún suceso de punto muerto desde esa partición de base de datos.

- El parámetro BUFFERSIZE restringe el tamaño de los sucesos STMT, STMT_HISTORY, DATA_VALUE y DETAILED_DLCONN. Si un suceso STMT o STMT_HISTORY no cabe en un almacenamiento intermedio, se trunca mediante el truncamiento del texto de la sentencia. Si un suceso DETAILED_DLCONN no puede caber dentro de un almacenamiento intermedio, éste se trunca eliminando bloqueos. Si todavía no tiene cabida, el texto de la sentencia se trunca. Si un suceso DATA_VAL no cabe en un almacenamiento intermedio, se trunca el valor de datos.

  Los supervisores de sucesos WITH DETAILS HISTORY VALUES (y, en menor grado, WITH DETAILS HISTORY) utilizan una cantidad significativa de espacio de pila de supervisor para efectuar el seguimiento de las sentencias y sus valores de datos. Si desea más información, consulte la descripción del parámetro de configuración del gestor de bases de datos **tamaño_pila_supervisor**.
- Si la partición de base de datos en la que se debe ejecutar el supervisor de sucesos no está activa, la activación de éste se produce la próxima vez que se activa la partición de base de datos.
- Una vez activado un supervisor de sucesos, éste se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva una partición de base de datos y ésta se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- ***Supervisores de sucesos de grabación en tabla:***
  - Notas generales:
    - Todas las tablas de destino se crean cuando se ejecuta la sentencia CREATE EVENT MONITOR.
    - Si la creación de una tabla no se realiza satisfactoriamente por cualquier razón, se envía un error al programa de aplicación y la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente.
    - Una tabla de destino sólo puede utilizarla un supervisor de sucesos. Durante el proceso de CREATE EVENT MONITOR, si se encuentra una tabla de destino que ya se había definido para que la utilizara otro supervisor de sucesos, la sentencia CREATE EVENT MONITOR no se ejecuta satisfactoriamente y se envía un error al programa de aplicación. Una tabla se habrá definido para que la utilice otro supervisor de sucesos si el nombre de la tabla coincide con un valor que se encuentra en la vista de catálogo SYSCAT.EVENTTABLES.
    - Durante el proceso de CREATE EVENT MONITOR, si ya existe una tabla, pero *no* se ha definido para que la utilice otro supervisor de sucesos, no se creará ninguna tabla y el proceso continuará. Se enviará un aviso al programa de aplicación.
    - Deberán existir espacios de tablas para que pueda ejecutarse la sentencia CREATE EVENT MONITOR. La sentencia CREATE EVENT MONITOR no crea espacios de tablas.
    - Si se han especificado, las palabras clave LOCAL y GLOBAL se pasan por alto. Con los supervisores de sucesos WRITE TO TABLE, se inicia una hebra o un proceso de salida del supervisor de sucesos en cada partición de base de datos de la instancia, y cada uno de estos procesos sólo informa datos a la partición de base de datos en la que está ejecutándose.

- Los supervisores de sucesos de grabación en tabla no graban los tipos de sucesos siguientes del archivo de anotaciones cronológicas plano del supervisor o del formato de conexión:
  - LOG_STREAM_HEADER
  - LOG_HEADER
  - DB_HEADER (Los elementos db_name y db_path no se graban. El elemento conn_time se graba en CONTROL.)
- En un entorno de bases de datos particionadas, los datos sólo se graban en las tablas de destino de las particiones de base de datos en las que existan sus espacios de tablas. Si no existe un espacio de tablas para una tabla de destino en ninguna partición de base de datos, se pasarán por alto los datos para esa tabla de destino. Este comportamiento permite a los usuarios elegir un subconjunto de particiones de base datos con el fin de supervisarlas, creando un espacio de tablas que sólo exista en determinadas particiones de base de datos.

  En un entorno de bases de datos particionadas, si algunas tablas de destino no residen en una partición de base de datos, pero otras tablas de destino sí que residen en esa misma partición de base de datos, sólo se registrarán los datos para las tablas de destino que residan en esa partición de base de datos.
- Los usuarios pueden podar manualmente todas las tablas de destino.

– Columnas de tabla:
- Los nombres de columna de una tabla coinciden con un identificador de elemento del supervisor de sucesos. 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.
- Dado que el rendimiento de las tablas con columnas CLOB es inferior al de las tablas que tienen columnas VARCHAR, considere la posibilidad de utilizar la palabra clave TRUNC al especificar el valor *grupo-evm* de STMT (o el valor *grupo-evm* de DLCONN, si se utiliza el tipo de suceso DEADLOCKS WITH DETAILS).

- A diferencia de otras tablas de destino, las columnas de la tabla CONTROL no coinciden con identificadores de elementos del supervisor. Las columnas se definen de la forma siguiente:

Nombre columna	Tipo datos	Nulos	Descripción
PARTITION_KEY	INTEGER	N	Clave de distribución (sólo base de datos particionada)
PARTITION_NUMBER	INTEGER	N	Número de partición de base de datos (sólo base de datos particionada)
EVMONNAME	VARCHAR(128)	N	Nombre de supervisor de sucesos
MESSAGE	VARCHAR(128)	N	Describe la naturaleza de la columna MESSAGE_TIME. Puede ser uno de los 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) - LAST_DROPPED_RECORD (última vez en que se ha producido un desbordamiento)
MESSAGE_TIME	TIMESTAMP	N	Indicación de fecha y hora

  - En un entorno de 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 distribución para la tabla. El valor de esta columna se elige de forma que cada proceso del supervisor de sucesos inserte datos en la partición de base de datos en la que está ejecutándose el proceso; es decir, se realizan operaciones de inserción localmente en la partición de base de datos en la que está ejecutándose el proceso del supervisor de sucesos. En cualquier partición de base de datos, el campo PARTITION_KEY contendrá el mismo valor. Esto significa que si se 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 de base de datos en la que se han insertado los datos. A diferencia de la columna PARTITION_KEY, la columna PARTITION_NUMBER no es obligatoria. La columna PARTITION_NUMBER no está permitida en un entorno de bases de datos no particionadas.

– Atributos de tabla:

  - Se utilizan atributos de tabla por omisión. Además de la clave de distribución (sólo bases de datos particionadas), durante la creación de las tablas no se especifica ninguna opción adicional.
  - En la tabla puede crearse índices.

- Pueden añadirse atributos de tabla adicionales (como, por ejemplo, si es volátil, RI, activadores, restricciones, etc.), pero el proceso del supervisor de sucesos (o la hebra) los pasará por alto.
- Si se añade "not logged initially" como atributo de tabla, se desactivará al ejecutarse COMMIT por primera vez y no volverá a activarse.

– Activación del supervisor de sucesos:
- Cuando se activa un supervisor de sucesos, todos los nombres de tabla de destino se recuperan de la vista de catálogo SYSCAT.EVENTTABLES.
- En un entorno de bases de datos particionadas, el proceso de activación se produce en cada partición de base de datos de la instancia. En una partición de base de datos en particular, el proceso de activación determina los espacios de tablas y los grupos de particiones de base de datos para cada tabla de destino. El supervisor de sucesos sólo se activa en una partición de base de datos si existe, como mínimo, una tabla de destino en esa partición de base de datos. Además, si no se encuentra alguna tabla de destino en una partición de base de datos, se marcará esa tabla de destino para que se descarten los datos destinados a la misma durante el proceso de ejecución.
- Si no existe una tabla de destino al activarse el supervisor de sucesos (o, en un entorno de 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, el supervisor de sucesos se activará la próxima vez que se active la partición.

– 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 base 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.
  - Siempre que se active un supervisor de sucesos de escritura en tabla, obtendrá bloqueos de tabla IN en cada tabla de destino para impedir que se modifiquen mientras el supervisor de sucesos esté activo. Los bloqueos de tabla se mantienen en todas las tablas mientras el supervisor de sucesos está activo. Si en cualquiera de estas tablas de destino es necesario un acceso exclusivo (por ejemplo, cuando se va a ejecutar un programa de utilidad), primero desactive el supervisor de sucesos para liberar los bloqueos de tabla antes de intentar dicho acceso.

- ***Compatibilidades***
  – Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODE puede especificarse en lugar de DBPARTITIONNUM
    - Se pueden utilizar comas para separar varias opciones en la cláusula *info-tabla-destino* o la cláusula *info-tabla-destino-wlm*

Ejemplos

Ejemplo 1: El siguiente ejemplo crea un supervisor de sucesos denominado SMITHPAY. Este supervisor de sucesos, reunirá los datos de sucesos para la base de datos así como para las sentencias de SQL realizadas por la aplicación PAYROLL propiedad del ID de autorización JSMITH. Los datos se añadirán a la vía de acceso absoluta /home/jsmith/event/smithpay/. Se crearán un máximo de 25 archivos. Cada archivo debe tener una longitud máxima de 1.024 páginas de 4K. La E/S del archivo no estará bloqueada.

```
CREATE EVENT MONITOR SMITHPAY
  FOR DATABASE, STATEMENTS
  WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
  WRITE TO FILE '/home/jsmith/event/smithpay'
  MAXFILES 25
  MAXFILESIZE 1024
  NONBLOCKED
  APPEND
```

Ejemplo 2: El ejemplo siguiente crea un supervisor de sucesos denominado DEADLOCKS_EVTS. Este supervisor de sucesos reunirá los sucesos de puntos muertos y los grabará en la vía de acceso relativa DLOCKS. Se grabará un archivo y no hay tamaño de archivo máximo. Cada vez que se active el supervisor de sucesos, se añadirán los datos de sucesos al archivo 00000000.evt si existe. El supervisor de sucesos se iniciará cada vez que se inicie la base de datos. La E/S estará bloqueada por omisión.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
  FOR DEADLOCKS
  WRITE TO FILE 'DLOCKS'
  MAXFILES 1
  MAXFILESIZE NONE
  AUTOSTART
```

Ejemplo 3: Este ejemplo crea un supervisor de sucesos denominado DB_APPLS. Este supervisor de sucesos reúne sucesos de conexión y graba datos en la conexión con nombre /home/jsmith/applpipe.

```
CREATE EVENT MONITOR DB_APPLS
  FOR CONNECTIONS
  WRITE TO PIPE '/home/jsmith/applpipe'
```

Ejemplo 4: Este ejemplo, en el que se da por supuesto un entorno de 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 tablas, todas las tablas se crearán en un espacio de tablas seleccionado por el sistema, basándose en las normas que se describen en la cláusula IN *nombre-espacio-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 tablas, la tabla se creará en un espacio de tablas seleccionado por el sistema, basándose en las normas descritas en la cláusula IN *nombre-espacio-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)),
  STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
    EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
  XACT
```

CREATE FUNCTION

La sentencia CREATE FUNCTION se utiliza para registrar o definir una función definida por el usuario o una plantilla de función en el servidor actual.

Existen cinco tipos diferentes de funciones que se pueden crear utilizando esta sentencia. Cada una de 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 fuente o plantilla. Una función fuente se implanta invocando otra función (incorporada, externa, SQL o fuente) que ya está registrada en la base de datos.

  Es posible crear una función parcial, denominada *función de plantilla*, que define qué tipos de valores van a devolverse, pero que no contiene código ejecutable. El usuario la correlaciona con una función fuente de datos dentro de un sistema federado, de esta forma se puede invocar la función fuente de datos desde una base de datos federada. Una plantilla de función puede registrarse solamente en un servidor de aplicaciones que esté designado como servidor federado.
- Escalar, tabla o fila de SQL. El cuerpo de la función está escrito en SQL y se define junto con el registro en la base de datos. Devuelve un valor escalar, una tabla o una fila individual.

CREATE FUNCTION (escalar externa)

La sentencia CREATE FUNCTION (escalar externa) se utiliza para registrar una función escalar externa definida por el usuario en el servidor actual. Una *función escalar* devuelve un solo valor cada vez que se invoca y, en general, es válida donde una expresión SQL sea válida.

Invocación

Esta sentencia 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

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

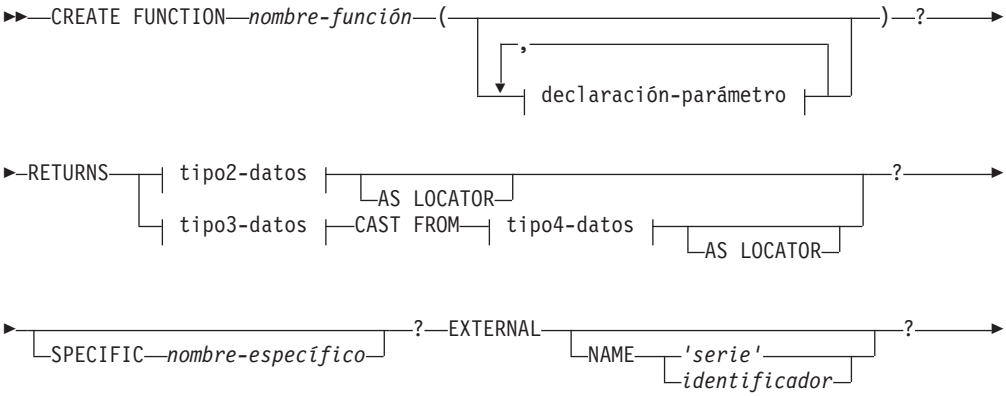
- Autorización CREATE_EXTERNAL_ROUTINE en la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema de la función no hace referencia a un esquema existente
  - Privilegio CREATEIN en el esquema, si el nombre de esquema de la función hace referencia a un esquema existente
- Autorización SYSADM o DBADM

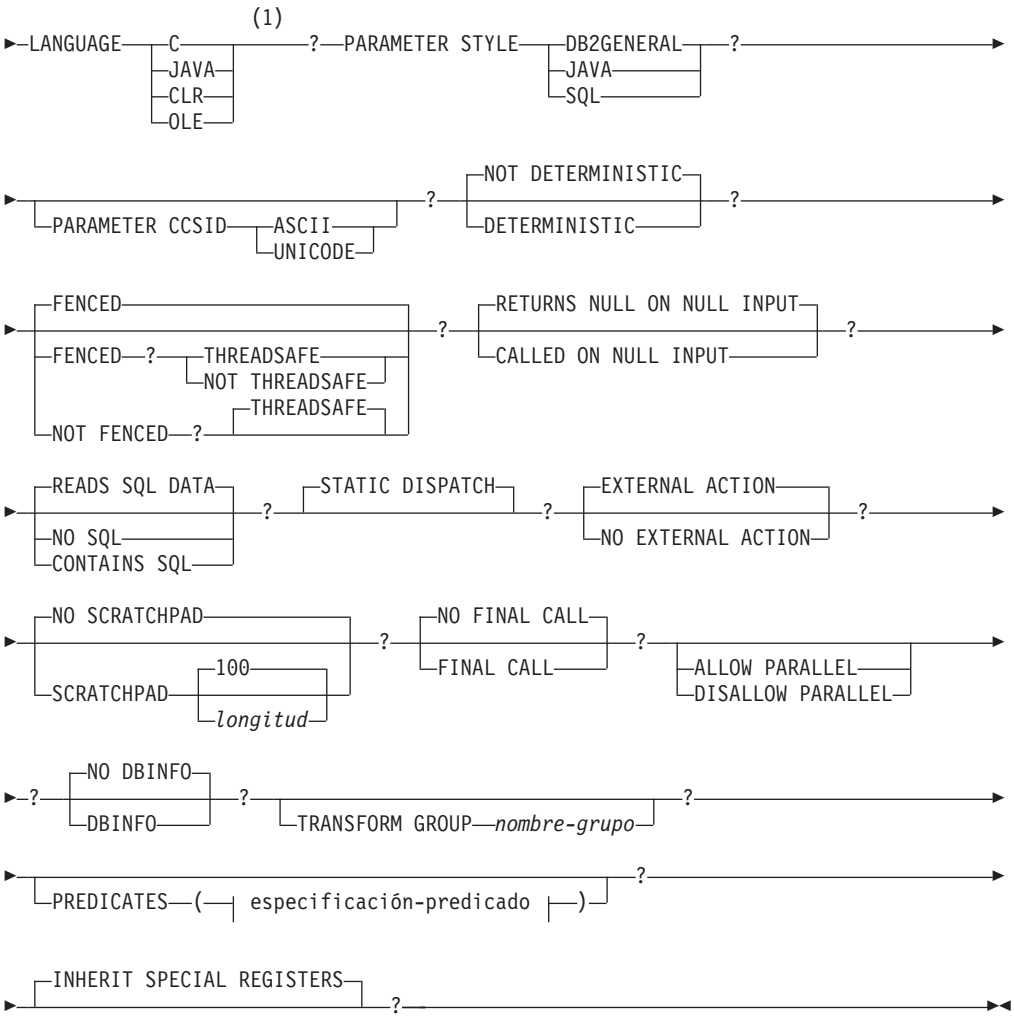
Para crear una función no restringida, el ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

- 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.

Sintaxis

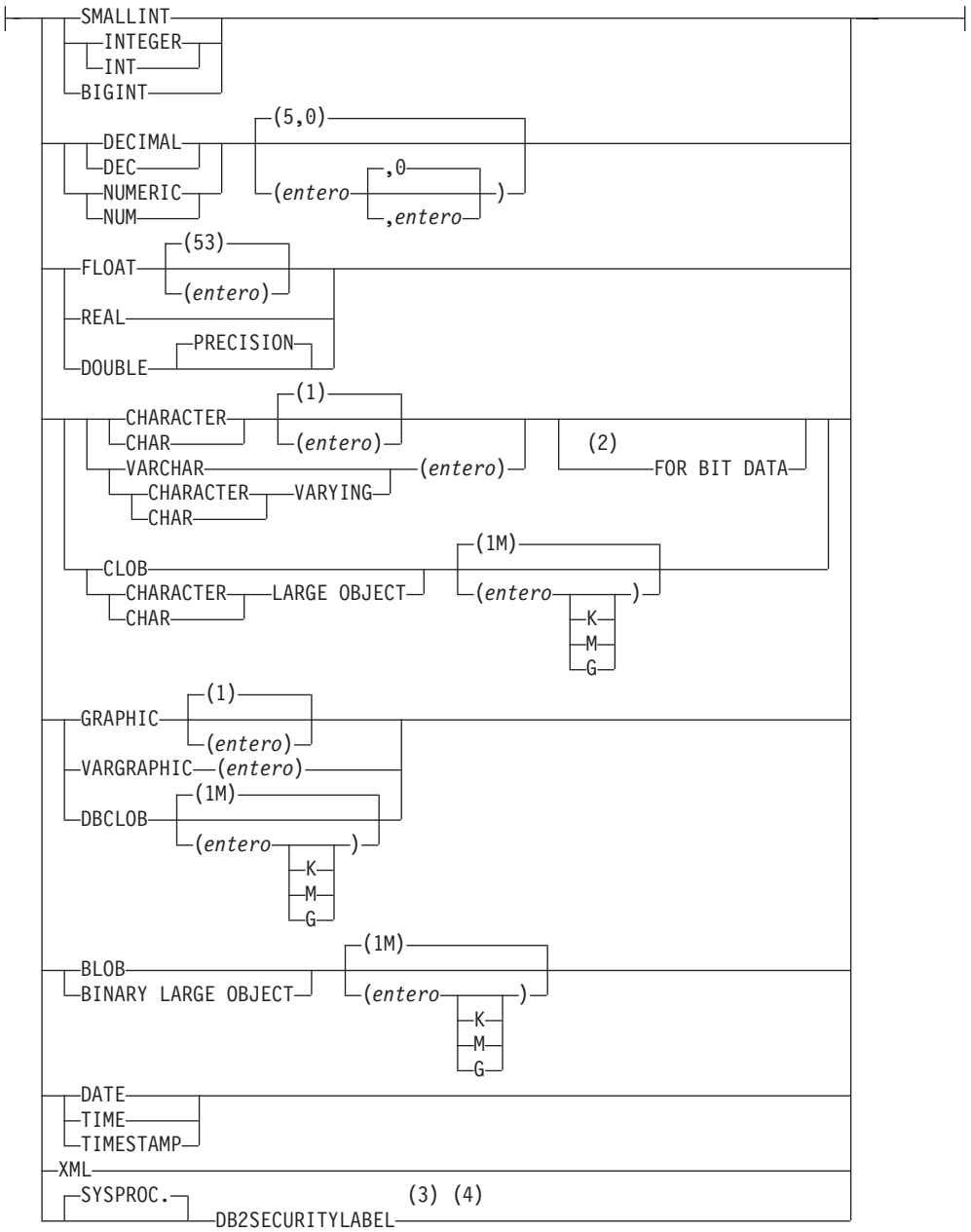




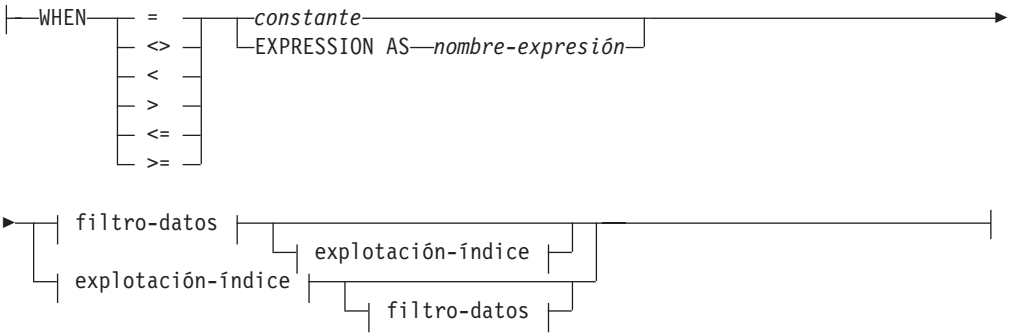
declaración-parámetro:

tipo1-datos, tipo2-datos, tipo3-datos, tipo4-datos:

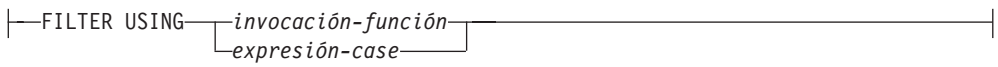
tipo-incorporado:



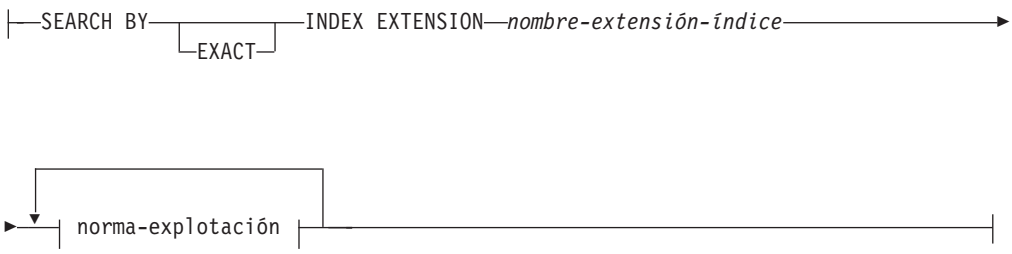
especificación-predicado:



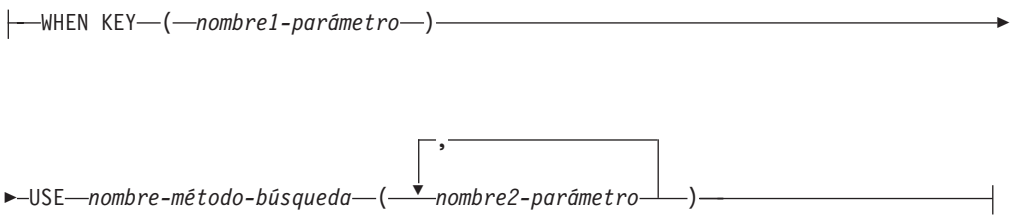
filtro-datos:



explotación-índice:



norma-explotación:



Notas:

1 También se da soporte a LANGUAGE SQL.

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

3 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

4 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada de *nombre-función* es un identificador 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 sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador 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 dé un resultado incorrecto, o incluso peor, que parezca que se ejecuta satisfactoriamente y genere un resultado diferente.

(*declaración-parámetro***,...)**

Identifica el número de parámetros de entrada de la función, 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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro

Especifica un nombre opcional para el parámetro de entrada. 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. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo1-datos
Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado, un tipo estructurado o un tipo de referencia. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Algunos tipos de datos no se soportan en todos los idiomas. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal, consulte "Tipos de datos que se correlacionan con tipos de datos de SQL en aplicaciones SQL incorporadas".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- DECFLOAT no es válido con LANGUAGE C, COBOL, CLR, JAVA y OLE (SQLSTATE 42815).
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de una función es una versión serializada del valor XML que se pasa como parámetro en la llamada de función, los parámetros del tipo XML deben declararse mediante la sintaxis `XML AS CLOB(`*n*`)`.
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- No se pueden especificar tipos array (SQLSTATE 42815).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

Para un tipo de referencia, el parámetro se puede especificar como REF(*nombre-tipo*) si el parámetro no tiene ámbito.

AS LOCATOR
Especifica que se pasa un localizador para el valor del parámetro a la función en lugar del valor real. Especifique AS LOCATOR sólo para parámetros con un tipo de datos LOB o un tipo diferenciado basado en un tipo de datos LOB (SQLSTATE 42601). El hecho de pasar localizadores en lugar de valores puede hacer que se pasen menos bytes a la función, especialmente cuando el valor del parámetro es muy grande.

La cláusula AS LOCATOR no tiene ningún efecto para determinar si los tipos de datos se pueden promocionar, ni afecta a la signatura de función, que se utiliza en la resolución de función.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS
Esta cláusula obligatoria identifica el resultado de la función.

tipo2-datos
Especifica el tipo de datos de la salida.

En este caso, se aplican exactamente las mismas consideraciones que para los parámetros de funciones externas 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.

tipo3-datos **CAST FROM** *tipo4-datos*
Especifica el tipo de datos de la salida.

Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función. Por ejemplo, en

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))
  RETURNS DATE CAST FROM CHAR(10)
  ...
```

el código de función devuelve un valor CHAR(10) al gestor de bases de datos que, a su vez, lo convierte en DATE y pasa dicho valor a la sentencia que realiza la invocación. El *tipo-datos4* debe ser convertible al parámetro *tipo-datos3*. Si no es convertible, se genera un error (SQLSTATE 42880).

Debido a que la longitud, precisión o escala de *tipo-datos3* puede inferirse de *tipo-datos4*, no es necesario (pero está permitido) especificar la longitud, precisión o escala de los tipos con parámetros especificados para *tipo-datos3*. En su lugar, pueden utilizarse paréntesis vacíos (por ejemplo, puede utilizarse VARCHAR()). No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Los tipos diferenciados, los tipos array y l os tipos estructurados no son válidos como tipo especificado en *tipo4-datos* (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 128). 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*".

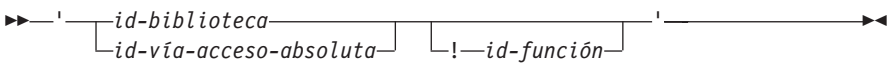
NAME *'serie'*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo.

La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función de dentro de la biblioteca). No obstante, cuando la función se utiliza en una sentencia de SQL, la biblioteca y la función que está dentro de la biblioteca deben existir y debe poder accederse a éstas desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

  La *serie* se puede especificar de la forma siguiente:



  Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

  id-biblioteca
  Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

  – En los sistemas UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.
  – En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

  id-vía-acceso-absoluta
  Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, en los sistemas UNIX, /u/jchui/mylib/myfunc haría que el gestor de bases de datos busque la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id-función
Identifica el nombre del punto de entrada de la función que debe invocarse. El signo de admiración (!) sirve de delimitador entre el ID de biblioteca y el ID de función.

Por ejemplo, en un sistema UNIX, 'mymod!func8' indicaría al gestor de bases de datos que debe buscar la biblioteca $inst_home_dir/sqllib/function/mymod y que debe utilizar el punto de entrada func8 que está dentro de esa biblioteca.

En los sistemas operativos Windows, 'mymod!func8' indicaría al gestor de bases de datos que debe cargar el archivo mymod.dll y que debe llamar a la función func8() en la biblioteca de enlace dinámico (DLL).

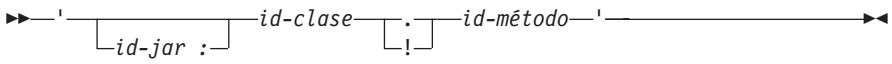
Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

El cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste deberá existir cuando se ejecute la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

  La *serie* se puede especificar de la forma siguiente:



  Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

  id-jar
  Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'.

  id-clase
  Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir

el prefijo completo del paquete, por ejemplo 'myPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../myPacks/UserFuncs/' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '...\myPacks\UserFuncs\'.

id-método
Identifica el nombre del método del objeto Java que se invocará.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se ejecuta la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos; de lo contrario, se devolverá un error (SQLSTATE 42724).

  Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.

  La *serie* se puede especificar de la forma siguiente:

  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 proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres

para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método
Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La *serie* especificada es el identificador de programa OLE (idprog) o identificador de clase (clsid) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando. No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al ejecutar la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

  La *serie* se puede especificar de la forma siguiente:

  Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

  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 del 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.

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 de una clase Java.

CLR Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En estos momentos, sólo se soporta LANGUAGE CLR en el caso de las funciones definidas por el usuario que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para las funciones definidas por el usuario que están almacenadas en DB2 para los sistemas operativos Windows. THREADSAFE no puede especificarse para las UDF que se han definido con LANGUAGE OLE (SQLSTATE 42613).

PARAMETER STYLE
Esta cláusula se utiliza para especificar los convenios utilizados para pasar parámetros a funciones y devolver el valor de las mismas.

DB2GENERAL
Se utiliza para especificar los convenios para pasar parámetros a funciones externas y para devolver los valores procedentes de esas funciones, que están definidas como método en una clase Java. Sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

El valor DB2GENRL puede utilizarse como sinónimo de DB2GENERAL.

JAVA
Significa que la función utilizará un convenio para el envío de parámetros que se ajusta a la especificación para el lenguaje Java y las Rutinas SQLJ. Esto sólo se puede especificar cuando se utiliza LANGUAGE JAVA, no se especifican tipos de datos estructurados como parámetros ni se especifican los tipos de datos CLOB, BLOB o DBCLOB como tipos de retorno (SQLSTATE 429B8). Las funciones PARAMETER STYLE JAVA no dan soporte a las cláusulas FINAL CALL, SCRATCHPAD o DBINFO.

SQL
Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es una base de datos Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo gráfico, tipo XML ni tipos definidos por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, las funciones se pueden crear con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

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, revisadas o comprobadas correctamente, puede poner en peligro 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 al mismo tiempo la función en el mismo proceso que otra rutina.

Para las funciones FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT
Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la ejecución, cualquiera de los argumentos de la función es nulo, no se invoca la función definida por el usuario y el resultado es el valor nulo.

Si se especifica CALLED ON NULL INPUT, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero corresponde a la UDF comprobar si los valores de argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO SQL, CONTAINS SQL, READS SQL DATA
Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL
Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION
Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada tarea paralela, en lugar de una nota para la función. Especifique la cláusula DISALLOW PARALLEL en el caso de las funciones que no funcionen bien con paralelismo.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

NO SCRATCHPAD o SCRATCHPAD *longitud*
Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (Es muy recomendable que las funciones definidas por el usuario sean reentrantes, por lo que una memoria de trabajo proporciona un medio para que la función “guarde el estado” entre una llamada y la siguiente.)

Si se especifica SCRATCHPAD, cuando se efectúa la primera invocación de la función definida por el usuario, se asigna memoria para que la función externa utilice una memoria de trabajo. Esta memoria de trabajo tiene las siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El tamaño por omisión es 100 bytes.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para referencia a la función externa en la sentencia de SQL. Por tanto, si la función UDFX de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarían tres memorias de trabajo.

  ```
  SELECT A, UDFX(A) FROM TABLEB
    WHERE UDFX(A) > 103 OR UDFX(A) < 19
  ```

  Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si la función se ejecuta en varias particiones de base de datos, se asigna una memoria de trabajo en cada partición donde se procesa la función, para cada referencia a la función en la sentencia de SQL. De 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 de base de datos? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique ALLOW PARALLEL.

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de base de datos.

El valor por omisión es ALLOW PARALLEL, excepto si se especifica una o varias de las opciones siguientes en la sentencia.

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

Si se especifica o está implícita alguna de estas opciones, el valor por omisión es DISALLOW PARALLEL.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

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 de SQL que realicen una transformación adecuada entre el tipo estructurado y sus atributos de tipo incorporados.

PREDICATES
Define el filtrado o la utilización de la extensión de índice que se lleva a cabo 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*
Introduce un uso específico de la función en un predicado mediante un operador de comparación ("=", "<", ">", ">=", "<=", "<>").

constante
Especifica un valor constante con un tipo de datos comparable con el tipo RETURNS de la función (SQLSTATE 42818). Cuando un predicado utiliza esta función con el mismo operador de comparación y esta constante, el optimizador puede utilizar el filtrado y la explotación del índice.

EXPRESSION AS *nombre-expresión*
Proporciona un nombre para una expresión. Cuando un predicado utiliza esta función con el mismo operador de comparación y una expresión, puede utilizarse el filtrado y la explotación del índice. La expresión se asigna como nombre de expresión, para que pueda utilizarse como argumento de una función de búsqueda. El *nombre-expresión* no puede ser igual a ningún *nombre-parámetro* de la función que se está creando (SQLSTATE 42711). Cuando se especifica una expresión, se identifica el tipo de la expresión.

FILTER USING
Permite especificar la utilización de una función externa o expresión CASE para un mayor filtrado de la tabla de resultados.

invocación-función
Especifica una función de filtro que se puede utilizar para realizar un filtrado adicional de la tabla de resultados. Esto es una versión de la función definida (utilizada en el predicado) que reduce el número de filas en el predicado definido por el usuario, para determinar si las filas cumplen los requisitos. Si los resultados producidos por el índice son cercanos a los resultados previstos para el predicado definido por

el usuario, puede no ser necesario aplicar la función de filtrado. Si no se especifica esta opción, no se realizará el filtrado de datos.

Esta función puede utilizar como argumento cualquier *nombre-parámetro*, el *nombre-expresión* o constantes (SQLSTATE 42703), y devuelve un entero (SQLSTATE 428E4). Si se devuelve el valor 1, significa que se conserva la fila; de lo contrario, se descartará.

Esta función también debe cumplir estas condiciones:

- No debe estar definida con LANGUAGE SQL (SQLSTATE 429B4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 42845)
- No debe tener un tipo de datos estructurado para ninguno de los parámetros (SQLSTATE 428E3)
- No debe incluir una subconsulta (SQLSTATE 428E4)
- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E4)

Si un argumento invoca otra función u otro método, estas normas también se aplican para la función o el método anidados. No obstante, los métodos de observador generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o método utilizado como argumento), siempre que el resultado de evaluar el argumento sea un tipo de datos incorporado.

El usuario que define la función debe disponer de privilegio EXECUTE para la función de filtro especificada.

La cláusula *invocación-función* no debe superar los 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

expresión-case
Especifica una expresión CASE para realizar un mayor filtrado de la tabla de resultados. La *cláusula-when-buscada* y la *cláusula-when-simple* pueden utilizar *nombre-parámetro*, *nombre-expresión* o una constante (SQLSTATE 42703). Como expresión-resultado se puede utilizar una función externa con las normas especificadas en FILTER USING *invocación-función*. Cualquier función o método al que se hace referencia en la *expresión-case* también debe seguir las cuatro normas indicadas en la *invocación-función*.

No se pueden utilizar subconsultas ni las expresiones XMLQUERY o XMLEXISTS en ningún lugar de la *expresión-case* (SQLSTATE 428E4).

La expresión case debe devolver un valor entero (SQLSTATE 428E4). Un valor de retorno de 1 en la expresión-resultado significa que se conserva la fila; de lo contrario, se elimina.

La cláusula *invocación-case* no debe superar los 65.536 bytes de longitud en la página de códigos de la base de datos (SQLSTATE 22001).

explotación-índice
Define un conjunto de normas para el método de búsqueda de una extensión de índice que se puede utilizar para explotar el índice.

SEARCH BY INDEX EXTENSION *nombre-extensión-índice*
Identifica la extensión de índice. El *nombre-extensión-índice* debe identificar una extensión de índice existente.

EXACT
Indica que la búsqueda por índice es exacta respecto a la evaluación del predicado. Utilice EXACT para indicar a DB2 que no es necesario aplicar la función de predicado original definida por el usuario ni el filtro después la búsqueda por índice. El predicado definido como EXACT es útil cuando la búsqueda por índice devuelve los mismos resultados que el predicado.

Si no se especifica EXACT, después de la búsqueda por índice se aplica el predicado original definido por el usuario. Si se prevé que el índice sólo proporcione una aproximación del predicado, no especifique la opción EXACT.

Si no se utiliza la búsqueda por índice, es necesario aplicar la función de filtro y el predicado original.

norma-explotación
Describe los patrones y argumentos de la búsqueda y cómo se pueden utilizar para realizar una búsqueda por índice mediante un método de búsqueda definido en la extensión de índice.

WHEN KEY (*nombre1-parámetro***)**
Define el patrón de búsqueda. Se puede especificar un solo patrón de búsqueda para una clave. El valor *nombre-parámetro1* identifica los nombres de parámetros de la función definida (SQLSTATE 42703 ó 428E8).

El tipo de datos de *nombre-parámetro1* debe coincidir con el de la clave fuente especificada en la extensión de índice (SQLSTATE 428EY). La coincidencia debe ser exacta para los tipos de datos incorporados y diferenciados y debe darse dentro de la misma jerarquía de tipos en el caso de tipos estructurados.

El resultado de esta cláusula es verdadero cuando los valores del parámetro indicado son columnas que están abarcadas por un índice, de acuerdo con la extensión de índice especificada.

USE *nombre-método-búsqueda***(***nombre2-parámetro,...***)**
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

- La determinación de si un tipo de datos es convertible a otro no tiene en cuenta la longitud, precisión ni escala de los tipos de datos con parámetros, como son CHAR y DECIMAL. Por esta razón, pueden producirse errores al utilizar una función como resultado del intento de conversión de un valor del tipo de datos fuente a un valor del tipo de datos de destino. Por ejemplo, VARCHAR es

convertible a DATE, pero si el tipo de fuente está realmente definido como VARCHAR(5), al utilizar la función se producirá un error.

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas de promoción que afectarán a sus valores de entrada (consulte el apartado "Promoción de tipos de datos"). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - INTEGER en lugar de SMALLINT
  - DOUBLE en lugar de REAL
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- Para asegurar la portabilidad de las UDF de una plataforma a otra, no deben utilizarse los tipos de datos siguientes:
  - FLOAT- utilice DOUBLE o REAL en su lugar.
  - NUMERIC- utilice DECIMAL en su lugar.
  - LONG VARCHAR- utilice CLOB (o BLOB) en su lugar.
- Una función y un método no pueden estar en una relación de alteración temporal (SQLSTATE 42745). Para obtener más información sobre la alteración temporal, consulte "CREATE TYPE (estructurada)".
- Una función no puede tener la misma signatura que un método (cuando se compara el primer *tipo-parámetro* de la función con el *tipo-sujeto* del método) (SQLSTATE 42723).
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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.
- Los parámetros de XML sólo están soportados en las funciones externas de LANGUAGE JAVA si se especifica la cláusula PARAMETER STYLE DB2GENERAL.
- ***Restricciones de acceso a las tablas***

  Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.
- ***Privilegios***

  - El usuario que define una función siempre recibe el privilegio EXECUTE y WITH GRANT OPTION 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.
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para 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

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 implementar 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 vocal 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 tuple 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 un nombre para la expresión, que se pasa como argumento a la función productora de rangos.

CREATE FUNCTION (tabla externa)

La sentencia CREATE FUNCTION (tabla externa) se utiliza para registrar una función de tabla externa definida por el usuario en el servidor actual.

En la cláusula FROM de una sentencia SELECT, puede utilizarse una *función de tabla* y ésta devuelve una tabla a la sentencia SELECT de fila en fila.

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

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

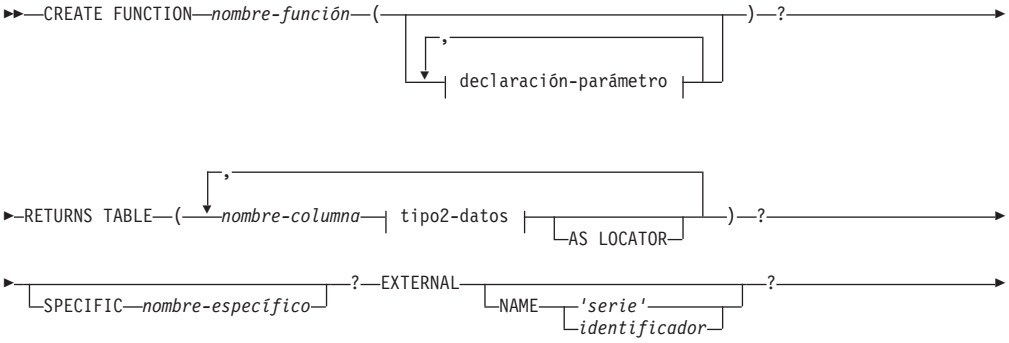
- Autorización CREATE_EXTERNAL_ROUTINE en la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
  - El privilegio CREATEIN en el esquema, si existe el nombre de esquema de la función
- Autorización SYSADM o DBADM

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.

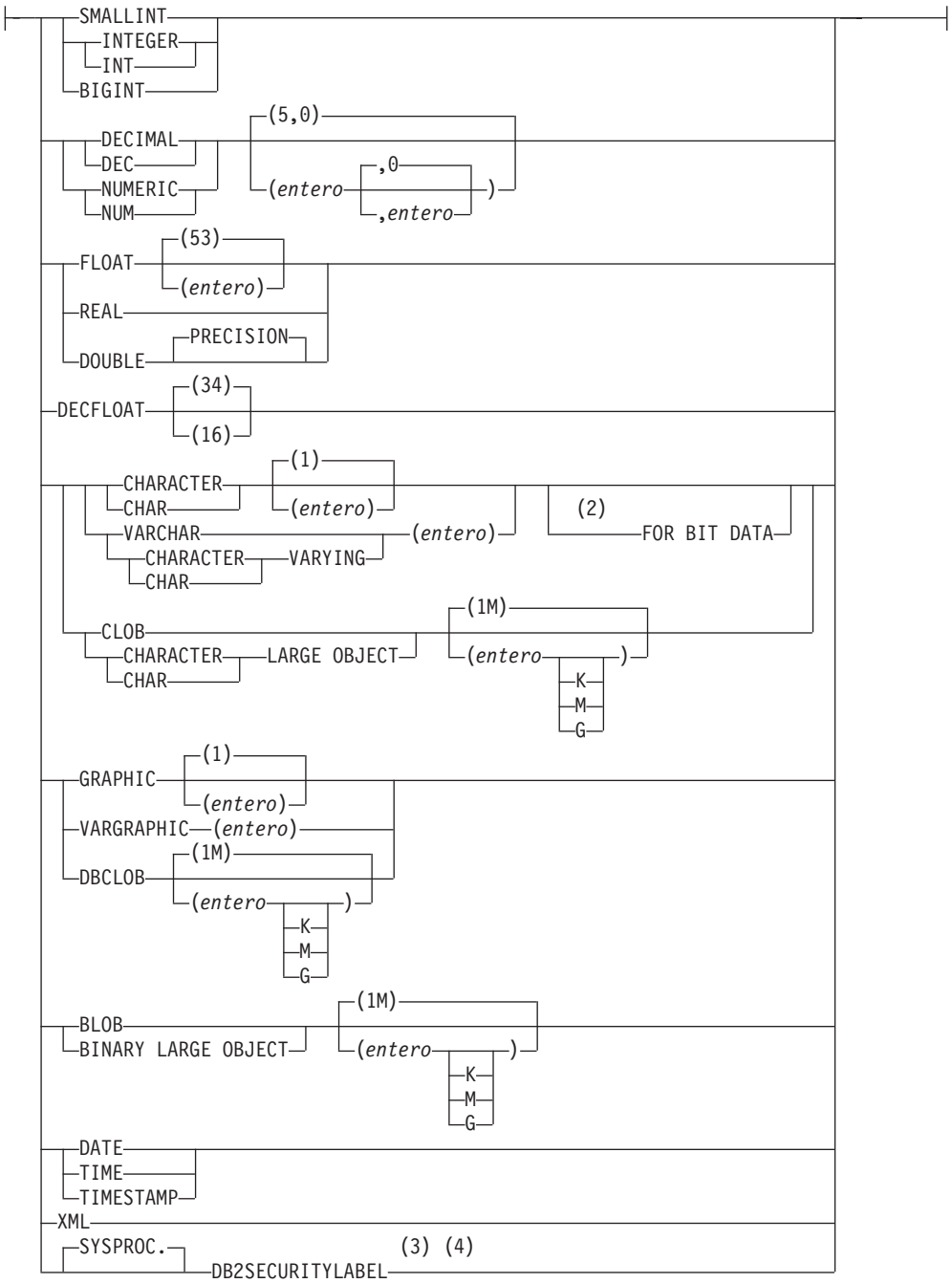
Sintaxis



declaración-parámetro:

tipo1-datos, tipo2-datos:

tipo-incorporado:



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 sobre cómo crear funciones de tabla LANGUAGE SQL, consulte “CREATE FUNCTION (escalar de SQL, tabla o fila)”.

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

3 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

4 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

Descripción

nombre-función

Indica el nombre de la función que se está definiendo. Consiste en el nombre, calificado o no calificado, que designa una función. La forma no calificada de *nombre-función* es un identificador 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 sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL. El nombre calificado no debe ser el mismo que el tipo de datos del primer parámetro, si ese parámetro es un tipo estructurado.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

(*declaración-parámetro***,...)**

Identifica el número de parámetros de entrada de la función, 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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el

mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro
Especifica un nombre opcional para el parámetro de entrada. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo1-datos
Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado, un tipo estructurado o un tipo de referencia. Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE". Algunos tipos de datos no se soportan en todos los idiomas. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal, consulte "Tipos de datos que se correlacionan con tipos de datos de SQL en aplicaciones SQL incorporadas".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- DECIMAL (y NUMERIC) no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de una función es una versión serializada del valor XML que se pasa como parámetro en la llamada de función, los parámetros del tipo XML deben declararse mediante la sintaxis `XML AS CLOB(n)`.
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- No se pueden especificar tipos array (SQLSTATE 42815).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

Para un tipo de referencia, el parámetro se puede especificar como REF(*nombre-tipo*) si el parámetro no tiene ámbito.

AS LOCATOR
Especifica que se pasa un localizador para el valor del parámetro a la función en lugar del valor real. Especifique AS LOCATOR sólo para parámetros con un tipo de datos LOB o un tipo diferenciado basado en un tipo de datos LOB (SQLSTATE 42601). El hecho de pasar localizadores en lugar de valores puede hacer que se pasen menos bytes a la función, especialmente cuando el valor del parámetro es muy grande.

La cláusula AS LOCATOR no tiene ningún efecto para determinar si los tipos de datos se pueden promocionar, ni afecta a la signatura de función, que se utiliza en la resolución de función.

Si la función es FENCED y tiene la opción NO SQL, no puede especificarse la cláusula AS LOCATOR (SQLSTATE 42613).

RETURNS 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.

tipo2-datos
Especifica el tipo de datos de la columna y puede ser cualquier tipo de datos soportado para un parámetro de una UDF escrita en el lenguaje determinado, excepto para tipos estructurados (SQLSTATE 42997).

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 (Externa escalar)".

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 128). 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, 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'*

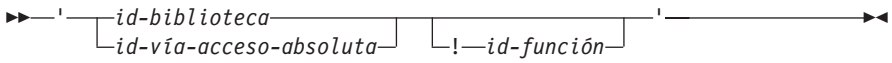
Esta cláusula identifica el código escrito por el usuario que implanta la función que se está definiendo.

La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y la función de la biblioteca, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. Al ejecutar la sentencia CREATE FUNCTION, no es preciso que exista la biblioteca (ni la función de dentro de la biblioteca). No obstante, cuando se emplea esta función en una sentencia de SQL, la biblioteca y la función de la biblioteca sí deben existir y estar accesibles desde la máquina que actúa como servidor de bases de datos.

  La *serie* se puede especificar de la forma siguiente:



Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

id-biblioteca
Identifica el nombre de la biblioteca que contiene la función. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

– En los sistemas UNIX, si se ha especificado 'myfunc' como *id_biblioteca* y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará la función en la biblioteca /u/production/sqllib/function/myfunc.

– En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

id-vía-acceso-absoluta
Identifica el nombre completo de vía de acceso del archivo que contiene la función.

Por ejemplo, en los sistemas UNIX, /u/jchui/mylib/myfunc haría que el gestor de bases de datos busque la biblioteca compartida myfunc en /u/jchui/mylib.

En sistemas operativos Windows, al especificar 'd:\mylib\ myfunc.dll' el gestor de bases de datos cargará la biblioteca de enlace dinámico myfunc.dll del directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

! id-función
Identifica el nombre del punto de entrada de la función que debe invocarse. El signo de admiración (!) sirve de delimitador entre el ID de biblioteca y el ID de función.

Por ejemplo, en un sistema UNIX, 'mymod!func8' indicaría al gestor de bases de datos que debe buscar la biblioteca

$inst_home_dir/sqllib/function/mymod y que debe utilizar el punto de entrada func8 que está dentro de esa biblioteca.

En los sistemas operativos Windows, 'mimodelo!func8' indicaría al gestor de bases de datos que debe cargar el archivo mimodelo.dll y que debe llamar a la función func8() en la biblioteca de enlace dinámico (DLL).

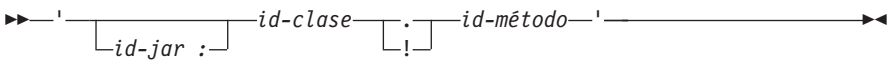
Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

En cualquier caso, el cuerpo de cada función externa debe estar en un directorio que sea accesible en todas las particiones de base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador opcional del archivo jar, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar la función definida por el usuario que se está creando. No es necesario que existan el identificador de clase ni el identificador de método cuando se ejecuta la sentencia CREATE FUNCTION. Si se especifica un *id_jar*, éste deberá existir cuando se ejecute la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, debe existir el identificador de método y debe ser accesible desde la máquina servidora de bases de datos.

  La *serie* se puede especificar de la forma siguiente:



  Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

  id-jar
  Designa el identificador de jar que se asignó a la colección jar cuando se instaló en la base de datos. Puede ser un identificador simple o un identificador calificado por un esquema. Algunos ejemplos son 'miJar' y 'miEsquema.miJar'

  id-clase
  Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo completo del paquete, por ejemplo 'miPacks.UserFuncs'. La máquina virtual Java buscará en el directorio '.../myPacks/UserFuncs/' las clases. En los sistemas operativos Windows de 32 bits, la máquina virtual Java buscará en el directorio '...\myPacks\UserFuncs\'.

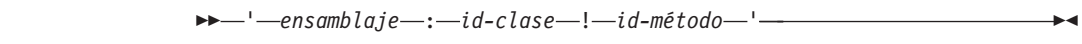
  id-método
  Identifica el nombre del método del objeto Java que se invocará.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método de la clase que el gestor de bases de datos invoca para ejecutar la función que se crea. No es necesario que existan el módulo, la clase y el método cuando se ejecuta la sentencia CREATE FUNCTION. No obstante, cuando se utiliza la función en una sentencia de SQL, el módulo, la clase y el método deben existir y se deben poder acceder desde la máquina del servidor de bases de datos; de lo contrario, se devolverá un error (SQLSTATE 42724).

Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe conocer que se está utilizando la infraestructura .NET en una función definida por el usuario para tomar decisiones necesarias en tiempo de ejecución. Todas las funciones definidas por el usuario utilizando la infraestructura .NET deben estar catalogadas como 'LANGUAGE CLR'.

La *serie* se puede especificar de la forma siguiente:



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 proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método
Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La *serie* especificada es el identificador de programa OLE (idprog) o identificador de clase (clsid) y el identificador del método, que invoca el gestor de bases de datos para ejecutar la función definida por el usuario que se está creando. No es preciso que exista el identificador de programa ni el identificador de clase y el identificador de método al ejecutar la sentencia CREATE FUNCTION. No obstante, cuando la función se utiliza en una sentencia de SQL, el identificador de método

debe existir y debe poder accederse a éste desde la máquina del servidor de bases de datos; de lo contrario, se devuelve un error (SQLSTATE 42724).

La *serie* se puede especificar de la forma siguiente:

Dentro de las comillas simples no está permitido especificar espacios en blanco adicionales.

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 del método del objeto OLE que se debe invocar.

NAME *identificador*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta la función que se está definiendo. El *identificador* especificado es un identificador SQL. El identificador SQL se utiliza como el *id-biblioteca* en la serie. A menos que sea un identificador delimitado, se convierte a mayúsculas. Si el identificador está calificado con un nombre de esquema, se ignora la parte correspondiente al nombre del esquema. Este formato de NAME sólo puede utilizarse con LANGUAGE C.

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje 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 de una clase Java.

CLR Significa que el gestor de bases de datos llamará a la función definida por el usuario como un método en una clase .NET. En estos momentos, sólo se soporta LANGUAGE CLR en el caso de las funciones definidas

por el usuario que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE Significa que el gestor de bases de datos llamará a la función definida por el usuario como si fuese un método expuesto por un objeto de automatización OLE. La función definida por el usuario debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para las funciones definidas por el usuario que están almacenadas en DB2 para 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, que están definidas como método en una clase Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

SQL

Se utiliza para especificar los convenios para pasar parámetros y para devolver el valor de funciones externas que cumplen con los convenios de llamada y enlace del lenguaje C, los métodos expuestos por los objetos de automatización OLE o los métodos estáticos públicos de un objeto .NET. Se debe especificar cuando se utiliza LANGUAGE C, LANGUAGE CLR o LANGUAGE OLE.

PARAMETER CCSID

Especifica el esquema de codificación que se debe utilizar para todos los datos de serie que se pasan a la función y desde ella. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII

Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca la función, la página de códigos de la aplicación para la función es la página de códigos de la base de datos.

UNICODE

Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En cualquier caso, cuando se invoca la función, la página de códigos de la aplicación para la función es 1208.

Si la base de datos no es Unicode y se crea una función con PARAMETER CCSID UNICODE, la función no puede tener ningún tipo gráfico ni ningún tipo definido por el usuario (SQLSTATE 560C1).

Si la base de datos no es Unicode, se pueden crear funciones de tabla con PARAMETER CCSID UNICODE, pero se aplican las normas siguientes:

- Se debe especificar un orden de clasificación alternativo en la configuración de la base de datos antes de crear la función de tabla (SQLSTATE 56031). Las funciones de tabla PARAMETER CCSID UNICODE se clasifican con el orden de clasificación alternativo especificado en la configuración de la base de datos.
- No se pueden utilizar conjuntamente las tablas o las funciones de tablas creadas con CCSID ASCII y las tablas o las funciones de tablas creadas con CCSID UNICODE en una sola sentencia de SQL (SQLSTATE 53090). Esto se aplica a las tablas y a las funciones de tabla a las que se hace referencia directamente en la sentencia, así como a las tablas y las 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, el tipo XML y los tipos definidos por el usuario no se pueden utilizar como parámetros en las funciones de tabla PARAMETER CCSID UNICODE (SQLSTATE 560C1).
- Las sentencias que hacen referencia a una función de tabla PARAMETER CCSID UNICODE sólo se pueden invocar desde un cliente DB2 Versión 8.1 o posterior (SQLSTATE 42997).
- Las sentencias de SQL siempre se interpretan en la página de códigos de la base de datos. En particular, significa que cada carácter de los literales, literales hexadecimales e identificadores delimitados debe tener una representación en la página de códigos de la base de datos; de lo contrario, el carácter se sustituirá por el carácter de sustitución.

Si la base de datos no es Unicode y se ha especificado un orden de clasificación alternativo en la configuración de la base de datos, las funciones se pueden crear con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos de serie que se pasan a la función y desde ella se convertirán a la página de códigos adecuada.

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

DETERMINISTIC o NOT DETERMINISTIC

Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas. Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados. Un ejemplo de una función de tabla que no es determinante es aquel que hace referencia a registros especiales, variables globales, funciones no determinantes o secuencias de un modo que afecta al tipo de resultado de la función de tabla.

FENCED o NOT FENCED

Esta cláusula especifica si la función se considera o no "segura" para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos (NOT FENCED) o no (FENCED).

Si una función se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) para que la función no pueda acceder a ellos. La mayoría de las funciones tienen la opción de ejecutarse como FENCED o NOT FENCED. En general, una función que se ejecute como FENCED no funcionará tan bien como otra de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para funciones no codificadas, revisadas o comprobadas correctamente, puede poner en peligro la integridad de DB2. DB2 toma algunas precauciones para muchas de las anomalías inadvertidas 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 al mismo tiempo la función en el mismo proceso que otra rutina.

Para las funciones FENCED, THREADSAFE es el valor por omisión si el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si la función se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para las funciones NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT

Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el

usuario está definida para no tener ningún parámetro, entonces por supuesto, esta condición de argumento nulo no puede surgir y no importa cómo se haya codificado esta especificación.

Si se especifica RETURNS NULL ON NULL INPUT y, durante la apertura de la función de tabla, cualquiera de los argumentos de la función es nulo, no se invoca la función definida por el usuario. El resultado de la función de tabla será una tabla vacía (una tabla sin ninguna fila).

Si se especifica CALLED ON NULL INPUT, entonces con independencia de si los argumentos son nulos o no, se invoca la función definida por el usuario. Puede devolver un valor nulo o un valor normal (no nulo). Pero corresponde a la UDF comprobar si los valores de argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO SQL, CONTAINS SQL, READS SQL DATA
Indica si la función debe emitir o no alguna sentencia de SQL y, en caso afirmativo, de qué tipo.

NO SQL
Indica que la función no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001). Si se especifican todas las cláusulas ALLOW PARALLEL, EXECUTE ON ALL DATABASE PARTITIONS y RESULT TABLE DISTRIBUTED, sólo estará permitida la opción NO SQL.

CONTAINS SQL
Indica que la función puede ejecutar las sentencias de SQL que no leen ni modifican datos de SQL (SQLSTATE 38004 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

READS SQL DATA
Indica que en la función pueden incluirse algunas sentencias de SQL que no modifican datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no reciben soporte en ninguna función devuelven un error distinto (SQLSTATE 38003 ó 42985).

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION
Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada

tarea paralela, en lugar de una nota para la función. Especifique la cláusula DISALLOW PARALLEL en el caso de las funciones que no funcionen bien con paralelismo.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

NO SCRATCHPAD o SCRATCHPAD *longitud*
Esta cláusula opcional especifica si debe proporcionarse una memoria de trabajo para una función externa. (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, establece el tamaño en bytes de la memoria de trabajo, que debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.
- El valor de inicialización consta sólo de ceros hexadecimales.
- 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 o ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

Especifica si, para una única referencia a la función, debe paralelizarse la función.

DISALLOW PARALLEL

Especifica que en cada invocación de la función, DB2 invoca la función en una única partición de base de datos.

ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

Especifica que en cada invocación de la función, DB2 invoca la función en todas las particiones de base de datos. Se devuelve todos los conjuntos de resultados obtenidos en cada partición de base de datos. La función no puede ejecutar sentencias de SQL (debe especificarse también la cláusula NO SQL).

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica por DB2 a la función como argumento en tiempo de invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613).

Si se especifica DBINFO, se pasará una estructura que contiene la información siguiente:

- Nombre de la base de datos: el nombre de la base de datos conectada actualmente
- ID de aplicación: ID de aplicación exclusivo que se establece para cada conexión a la base de datos
- ID de autorización de aplicación: el ID de autorización de ejecución de la aplicación, sin tener en cuenta las funciones anidadas entre esta función y la aplicación
- Página de códigos: página de códigos de la base de datos
- Nombre de esquema: no se puede aplicar a funciones de tabla externas

- Nombre de tabla: no se puede aplicar a funciones de tabla externas
- Nombre de columna: no se puede aplicar a funciones de tabla externas
- Versión o release de la base de datos: la versión, release y nivel de modificación del servidor de la base de datos que invoca la función
- Plataforma: el tipo de plataforma del servidor
- Números de columna del resultado del método de tabla: matriz de números de la columna del resultado que utiliza la sentencia que hace referencia a la función; esta información permite que la función sólo devuelva los s valores de columna necesarios y no todos los valores de ésta
- Número de partición de base de datos: el número de partición de base de datos en el que se invoca la función de tabla externa; en un entorno de una partición de base de datos, este valor es 0

CARDINALITY *entero*

Esta cláusula opcional proporciona una estimación del número esperado de filas que debe devolver la función con fines de optimización. Los valores válidos para *entero* están comprendidos dentro del rango 0 a 9.223.372.036.854.775.807, ambos incluidos.

Si no se especifica la cláusula CARDINALITY para una función de tabla, DB2 asumirá un valor finito como valor por omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

Aviso: Si una función tiene, de hecho, una cardinalidad infinita — es decir, devuelve una fila cada vez que se llama para ello y nunca devuelve una condición de "fin de tabla" — las consultas que requieran una condición de fin de tabla para funcionar correctamente serán infinitas y se deberán interrumpir. Las consultas que contienen una cláusula GROUP BY o ORDER BY son ejemplos de esta clase de consultas. No se recomienda escribir estas UDF.

TRANSFORM GROUP *nombre-grupo*

Indica el grupo de transformación que se debe utilizar, cuando se invoca la función, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación si la definición de la función incluye un tipo estructurado definido por el usuario, en calidad de tipo de datos de parámetro. Si no se especifica esta cláusula, se utiliza el nombre de grupo por omisión, DB2_FUNCTION. Si el *nombre-grupo* especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Si una función de transformación necesaria FROM SQL no está definida para el nombre de grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales que pueden actualizarse de la función heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una función que se invoca en la sentencia-select de un cursor, los valores iniciales se heredan del entorno cuando se abre el cursor. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que invoca la función no se le devolverá ninguno de los cambios realizados en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

Normas

- 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).
- ***Restricciones de acceso a las tablas*** Si una función se ha definido como READS SQL DATA, ninguna sentencia de la función podrá acceder a la tabla que la sentencia que ha invocado la función está modificando (SQLSTATE 57053). Por ejemplo, supongamos que la función definida por el usuario BONUS() se ha definido como READS SQL DATA. Si se invoca la sentencia UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO), no podrá leerse ninguna sentencia de SQL de la función BONUS desde la tabla EMPLOYEE.

Notas

- Al elegir los tipos de datos para los parámetros de una función definida por el usuario, tenga en cuenta las normas para la promoción que afectarán a sus valores de entrada. Por ejemplo, una constante que puede utilizarse como un valor de entrada podría tener un tipo de datos incorporado distinto del que se espera y, todavía más importante, podría no promoverse al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - INTEGER en lugar de SMALLINT
  - DOUBLE en lugar de REAL
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- Para garantizar la portabilidad de las UDF entre distintas plataformas, se recomienda utilizar los tipos de datos siguientes:
  - DOUBLE o REAL en lugar de FLOAT
  - DECIMAL en lugar de NUMERIC
  - CLOB (o BLOB) en lugar de LONG VARCHAR
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Se invocará una rutina Java definida como NOT FENCED como si se hubiese definido como FENCED THREADSAFE.
- ***Privilegios***
  - El usuario que define una función siempre recibe el privilegio EXECUTE y WITH GRANT OPTION 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.

- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 para 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 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.

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
```

CREATE FUNCTION (tabla externa OLE DB)

La sentencia CREATE FUNCTION (tabla externa OLE DB) se utiliza para registrar una función de tabla externa OLE DB definida por el usuario para acceder a los datos de un proveedor OLE DB.

Puede utilizarse una *función de tabla* en la cláusula FROM de SELECT.

Invocación

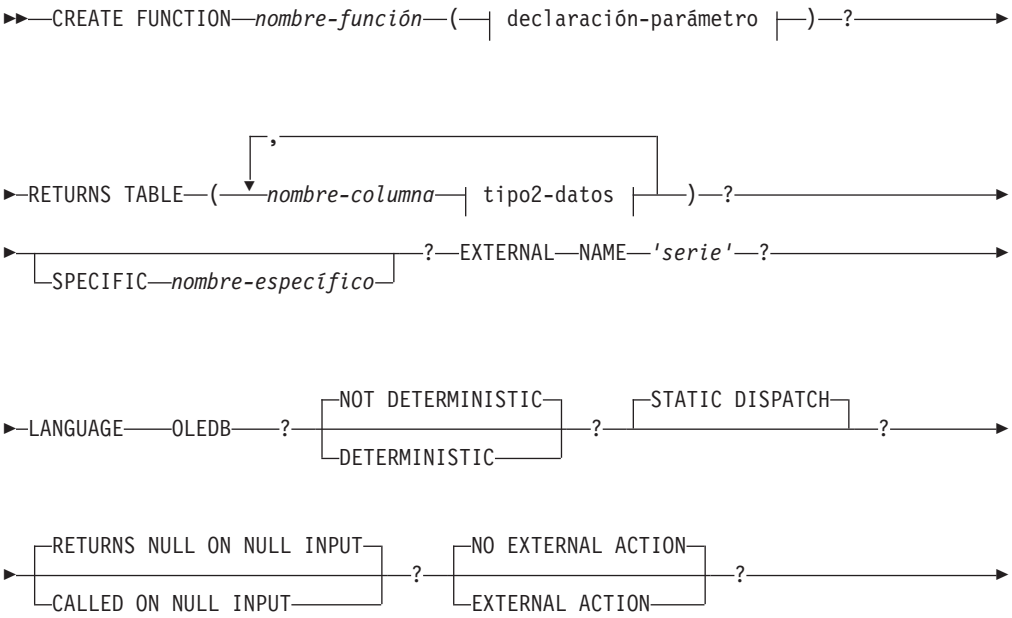
Esta sentencia puede incorporarse en un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

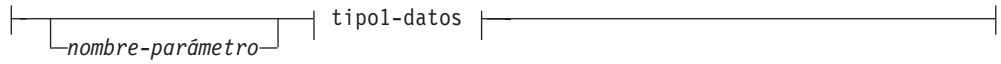
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización CREATE_EXTERNAL_ROUTINE en la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
  - El privilegio CREATEIN en el esquema, si existe el nombre de esquema de la función
- Autorización SYSADM o DBADM

Sintaxis

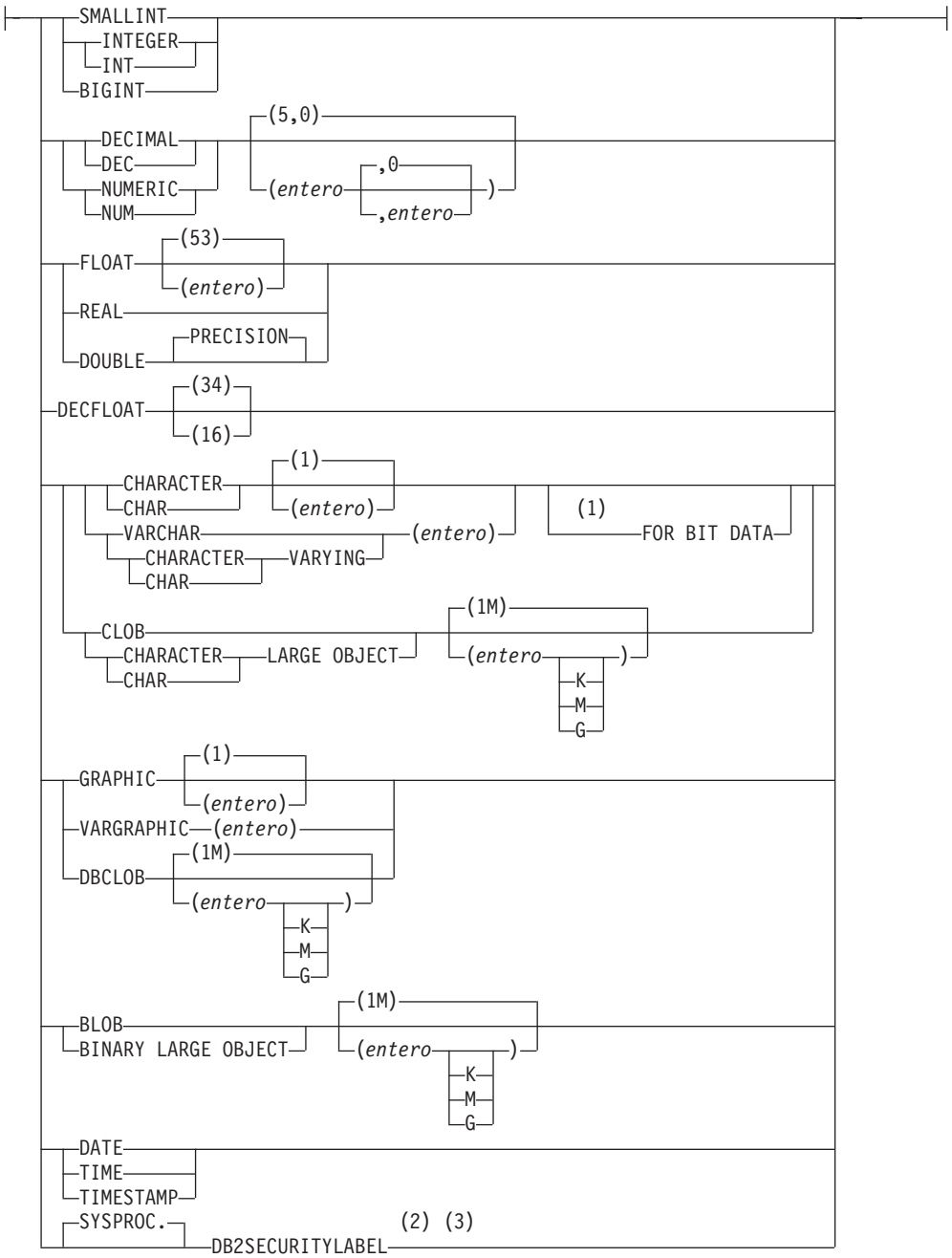


declaración-parámetro:



tipo1-datos, tipo2-datos:

tipo-incorporado:



Notas:

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

2 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

3 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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 sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

(*declaración-parámetro***,...)**

Identifica el 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 pasa el texto de mandato a un proveedor OLE DB.

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por consiguiente, se considera que CHAR(8) y CHAR(35) son del mismo tipo. Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro

Especifica un nombre opcional para el parámetro de entrada.

tipo1-datos

Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser cualquier tipo de datos de serie gráfica o de caracteres o un tipo

diferenciado basado en un tipo de datos de serie gráfica o de caracteres. No se da soporte a parámetros del tipo XML (SQLSTATE 42815).

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

RETURNS TABLE
Especifica que el resultado de la función es una tabla. El paréntesis que sigue a esta palabra clave delimita una lista de nombres y tipos de las columnas de la tabla, parecido al estilo de una sentencia CREATE TABLE simple que no tenga especificaciones adicionales (restricciones, por ejemplo).

nombre-columna
Especifica el nombre de la columna que debe ser igual que el nombre de columna del conjunto de filas correspondiente. El nombre no puede estar calificado y no se puede utilizar el mismo nombre para más de una columna de la tabla.

tipo2-datos
Especifica el tipo de datos de la columna. XML no es válido (SQLSTATE 42815).

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al 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).

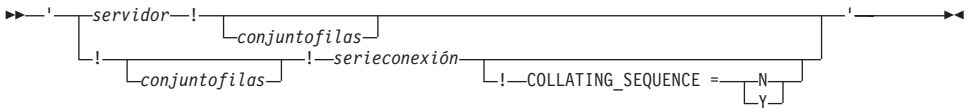
Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo es SQL seguido de una indicación de fecha y hora en forma de serie caracteres: SQLaammddhhmmssxxx.

EXTERNAL NAME *'serie'*
Esta cláusula identifica la tabla externa y el proveedor OLE DB.

La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes.

La serie especificada se utiliza para establecer una conexión y una sesión con un proveedor OLE DB y recuperar los datos de un conjunto de filas. No es necesario que existan el proveedor OLE DB ni la fuente de datos cuando se ejecuta la CREATE FUNCTION.

La *serie* se puede especificar de la forma siguiente:



servidor
Identifica el nombre local de una fuente de datos como lo define "CREATE SERVER".

conjuntofilas
Identifica el conjunto de filas (tabla) expuesto por el proveedor OLE DB. Deben proporcionarse nombres de tabla completamente calificados para los proveedores OLE DB que soportan nombres de catálogo o de esquema.

serieconexión
Versión de la serie de las propiedades de inicialización necesaria para conectarse a la fuente de datos. El formato básico de una serie de conexión se basa en la serie de conexión ODBC. La serie contiene una serie de pares de palabra clave/valor separados por puntos y comas. El signo igual (=) separa cada palabra clave y su valor. Las palabras clave son las descripciones de las propiedades de inicialización de OLE DB (conjunto de propiedades DBPROPSET_DBINIT) o palabras clave específicas del proveedor.

COLLATING_SEQUENCE
Especifica si la fuente de datos utiliza el mismo orden de clasificación que DB2 Database para Linux, UNIX y Windows. Para obtener detalles, consulte "CREATE SERVER". Los valores válidos son los siguientes:

- Y = El mismo orden de clasificación
- N = Diferente orden de clasificación

Si no se especifica COLLATING_SEQUENCE, se da por supuesto que la fuente de datos tiene un orden de clasificación distinto de DB2 Database para Linux, UNIX y Windows.

Si se proporciona el *servidor*, *serie-conexión* o COLLATING_SEQUENCE no están permitidos en el nombre externo. Se definen como las opciones de servidor CONNECTSTRING y COLLATING_SEQUENCE. Si no se proporciona ningún *servidor*, debe proporcionarse una *serieconexión*. Si no se proporciona *conjuntofilas*, la función de tabla debe tener un parámetro de entrada para realizar el paso a través del texto del mandato al proveedor OLE DB.

LANGUAGE OLEDB
Esto significa que el gestor de bases de datos desplegará un cliente OLE DB genérico incorporado para recuperar los datos de OLE DB. El desarrollado no necesita ninguna implantación de función de tabla.

Las funciones de tabla LANGUAGE OLEDB pueden crearse en cualquier plataforma, pero sólo se ejecutan en plataformas soportadas por Microsoft OLE DB.

DETERMINISTIC o NOT DETERMINISTIC
Esta cláusula opcional especifica si la función siempre devuelve los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si la función depende de ciertos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, una función DETERMINISTIC siempre debe devolver la misma tabla ante invocaciones sucesivas con entradas idénticas.

Con la especificación de NOT DETERMINISTIC, se evitan las optimizaciones que aprovechan el hecho de que las entradas idénticas siempre producen los mismos resultados.

STATIC DISPATCH
Esta cláusula opcional indica que, durante la resolución de la función, DB2 elige una función basada en los tipos estáticos (tipos declarados) de los parámetros de la función.

RETURNS NULL ON NULL INPUT o CALLED ON NULL INPUT
Esta cláusula opcional puede utilizarse para evitar una llamada a la función externa si cualquiera de los argumentos es nulo. Si la función definida por el usuario está definida para no tener parámetros, esta condición de argumento nulo no puede producirse.

Si se especifica RETURNS NULL ON NULL INPUT y si, en tiempo de ejecución, alguno de los argumentos de la función es nulo, no se llama a la función definida por el usuario y el resultado es la tabla vacía; es decir, una tabla sin filas.

Si se especifica CALLED ON NULL INPUT, entonces durante la ejecución sin tener en cuenta si los argumentos son o no nulos, se invoca la función definida por el usuario. Puede devolver una tabla vacía o no, dependiendo de su lógica. Pero corresponde a la UDF comprobar si los valores de argumentos son nulos.

El valor NULL CALL puede utilizarse como sinónimo de CALLED ON NULL INPUT para mantener la compatibilidad con versiones anteriores. De manera similar, puede utilizarse NOT NULL CALL como sinónimo de RETURNS NULL ON NULL INPUT.

NO EXTERNAL ACTION o EXTERNAL ACTION
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es NO EXTERNAL ACTION.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

EXTERNAL ACTION
Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

CARDINALITY *entero*
Esta cláusula opcional proporciona una estimación del número esperado de filas que debe devolver la función con fines de optimización. Los valores válidos de *entero* están en el rango de 0 a 2.147.483.647 inclusive.

Si no se especifica la cláusula CARDINALITY para una función de tabla, DB2 asumirá un valor finito como valor por omisión (el mismo valor asumido para las tablas para las que el programa de utilidad RUNSTATS no ha reunido estadísticas).

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

- 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 OPTION para la función, así como el derecho para poder eliminar la función.
- ***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

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, ' CONCAT
                ' stores.stor_name as name, ' CONCAT
                ' sum(sales. qty) as sales ' CONCAT
              ' from sales, stores ' CONCAT
              ' where sales.stor_id = stores.stor_id ' CONCAT
              ' group by sales.stor_id, stores.stor_name ' CONCAT
              ' order by sum(sales.qty) desc )) as f;
```

CREATE FUNCTION (con fuente o plantilla)

La sentencia CREATE FUNCTION (con fuente o plantilla) se utiliza para:

- Registrar una función definida por el usuario, basada en otra función escalar o de columna existente, en el servidor actual.
- 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 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).

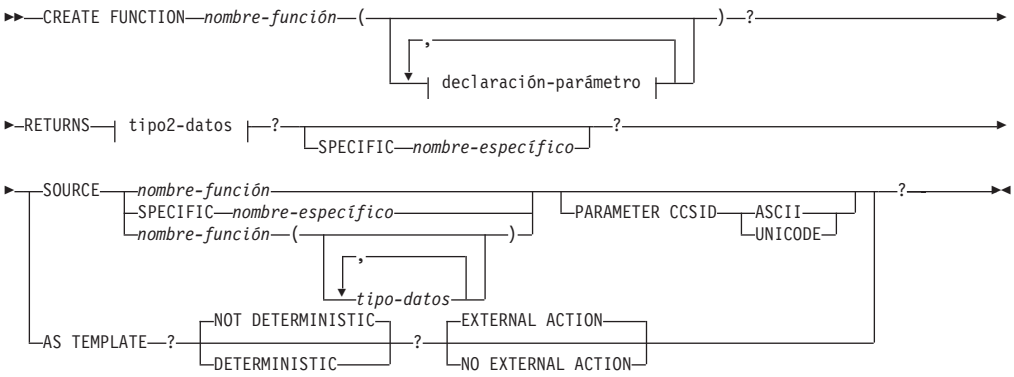
Autorización

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

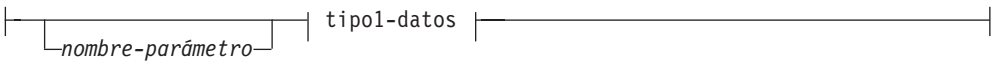
- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la función no existe
- El privilegio CREATEIN en el esquema, si existe el nombre de esquema de la función
- Autorización SYSADM o DBADM

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.

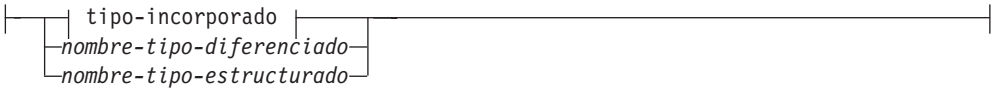
Sintaxis



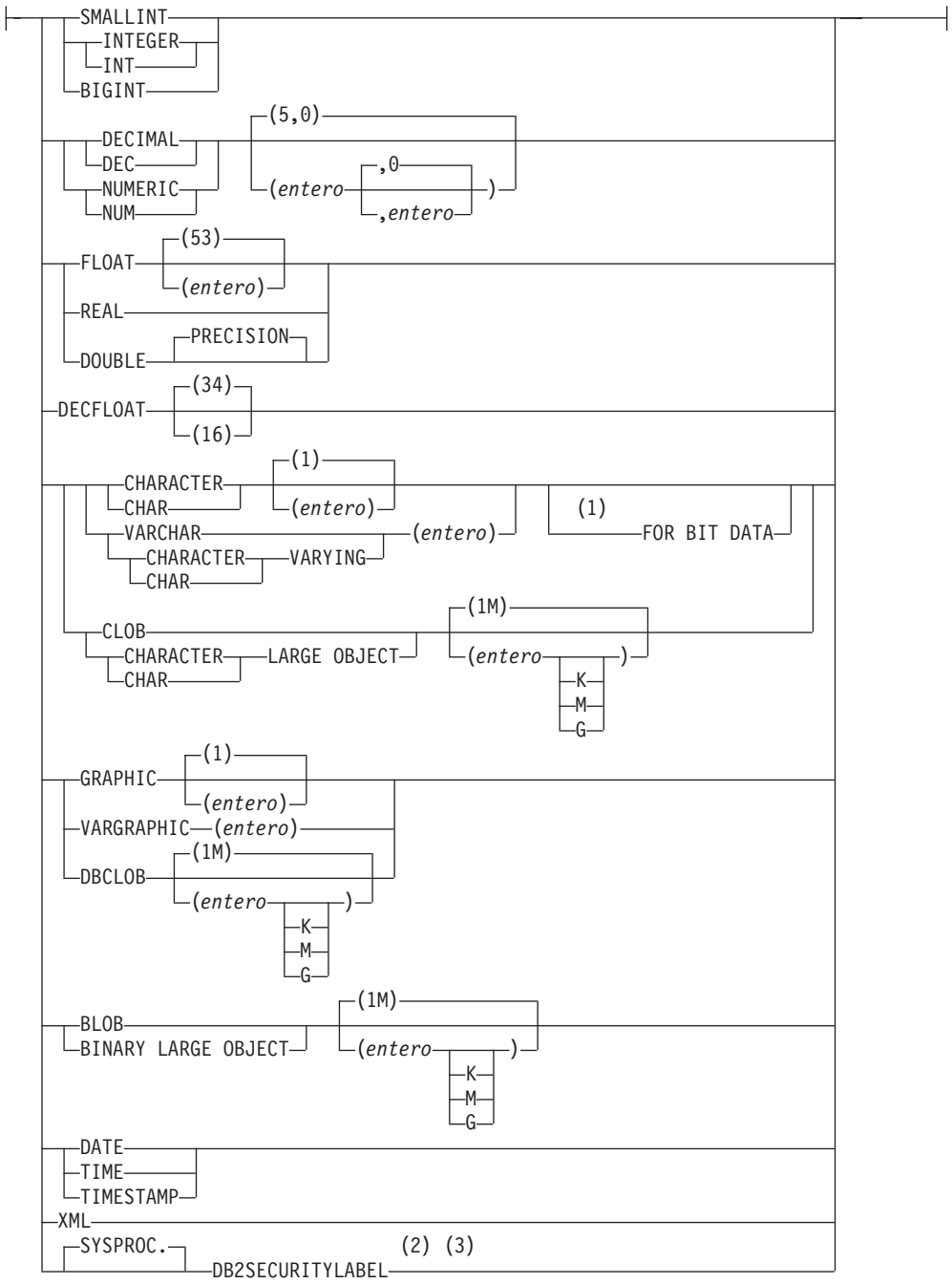
declaración-parámetro:



tipo1-datos, tipo2-datos:



tipo-incorporado:



Notas:

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

2 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

3 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE

42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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 sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador 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 fuente en una función ya existente con el fin de dar soporte a la misma función con un tipo diferenciado definido por el usuario, puede utilizarse el mismo nombre que el de la función con fuente. 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.

(*declaración-parámetro***,...)**
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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Esta restricción también es aplicable a una función y una plantilla de función con el mismo nombre en el mismo esquema. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por esta razón, se considera que CHAR(8) y CHAR(35) tienen el mismo tipo, al igual que

DECIMAL(11,2) y DECIMAL (4,3). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro
Especifica un nombre opcional para el parámetro de entrada. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo1-datos
Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado o un tipo estructurado.

Se puede utilizar cualquier tipo de datos SQL válido si éste se puede convertir en el tipo del parámetro correspondiente de la función identificada en la cláusula SOURCE (para obtener información consulte "Conversiones entre tipos de datos"). Sin embargo, esta comprobación no garantiza que no se vaya a producir un error cuando se invoque la función.

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte "CREATE TABLE".

- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.
- No se pueden especificar tipos array (SQLSTATE 42879).
- No se puede especificar un tipo de referencia especificado como REF(*nombre-tipo*) (SQLSTATE 42879).

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo estructurado definido por el usuario, deben existir las funciones de transformación apropiadas en el grupo de transformación asociado.

Puesto que la función es con fuente, 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 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 omite los paréntesis, se utiliza la longitud por omisión para el tipo de datos (vea "CREATE TABLE").

RETURNS
Esta cláusula obligatoria identifica el resultado de la función o plantilla de función.

tipo2-datos
Especifica el tipo de datos de la salida.

Con una función escalar con fuente, se acepta cualquier tipo de datos SQL válido, como un tipo diferenciado, siempre que se pueda convertir desde el tipo del resultado de la función fuente. Un tipo array no puede especificarse como tipo de datos de un parámetro (SQLSTATE 42879).

El parámetro de un tipo con parámetros no es necesario especificarlo, tal como se describió anteriormente para los parámetros de una función con fuente. 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 fuente en otra función, consulte el apartado "Normas" de esta sentencia.

Con una plantilla de función, los paréntesis vacíos no están permitidos (SQLSTATE 42611). Se debe especificar la longitud, la precisión o la escala para los tipos de datos con parámetros. Se recomienda especificar la misma longitud, precisión o escala que las de la función remota.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia de la función que se está definiendo. Este nombre específico puede utilizarse cuando se utiliza esta función como fuente, al 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 fuente puede ser una función incorporada (excepto para COALESCE, DATAPARTITIONNUM, DBPARTITIONNUM, GREATEST, HASHEDVALUE, LEAST, MAX (escalar), MIN (escalar), NULLIF, NVL, RAISE_ERROR, RID, RID_BIT, 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 fuente y sólo es válida si existe exactamente una función específica en el esquema con este *nombre-función* para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, para localizar la función se utiliza la vía de acceso de SQL actual (el valor del registro especial CURRENT PATH). El primer esquema de la vía de acceso de SQL que tiene una función con este nombre y para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función con este nombre en el esquema nombrado o si el esquema no está calificado y no hay ninguna función con este nombre en la vía de acceso a SQL, se devuelve un error (SQLSTATE 42704). Si hay más de una instancia específica autorizada de la función en el esquema nombrado o localizado, se devuelve un error (SQLSTATE 42725). Si existe una función con este nombre y el ID de autorización de la sentencia no tiene el privilegio EXECUTE en esta función, se devuelve un error (SQLSTATE 42501).

SPECIFIC *nombre-específico*
Identifica una función determinada definida por el usuario que debe servir como fuente, por el *nombre-específico* ya sea especificado o tomado por omisión en el tiempo de creación de la función. Esta variante de sintaxis no es válida para una función fuente que sea una función incorporada.

Si se proporciona un nombre no calificado, se utiliza la vía de acceso SQL actual para localizar la función. Se selecciona el primer esquema de la vía de acceso a SQL que tiene una función con este nombre específico para la que el ID de autorización de la sentencia dispone de privilegio EXECUTE.

Si no existe ninguna función para este *nombre-específico* en el esquema nombrado o si el nombre no está calificado y no hay ninguna función con este *nombre-específico* en la vía de acceso SQL, se devuelve un error (SQLSTATE 42704). Si existe una función con este *nombre-específico* y el ID de autorización de la sentencia no dispone de privilegio EXECUTE para esta función, se devuelve un error (SQLSTATE 42501).

nombre-función (tipo-datos,...)
Proporciona la signatura de la función, que identifica de manera exclusiva la función fuente. Esta es la única variante de sintaxis válida para una función fuente que es una función incorporada.

Las normas para la resolución de funciones se aplicarán con el fin de seleccionar una función entre las funciones que tienen el mismo nombre de función, según los tipos de datos que se han especificado en la cláusula SOURCE. Sin embargo, el tipo de datos de cada parámetro de la función seleccionada debe tener exactamente el tipo de datos correspondiente que se ha especificado en la función fuente.

nombre-función
Constituye el nombre de la función fuente. Si se proporciona un nombre no calificado, entonces se tienen en cuenta los esquemas de la vía de acceso a SQL del usuario.

tipo-datos
Debe coincidir con el tipo de datos que se haya especificado en la sentencia CREATE FUNCTION en la posición correspondiente (separado por comas).

Para los tipos de datos con parámetros no es necesario especificar la longitud, la precisión ni la escala. En su lugar, puede haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse al buscar una coincidencia del tipo de datos. Por ejemplo, DECIMAL() coincidirá con un parámetro cuyo tipo de datos esté definido como DECIMAL(7,2)).

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

De todas formas, si la longitud, la precisión o la escala están codificadas, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE FUNCTION. Puede ser útil para asegurarse de que se va a utilizar la función que desea. Tenga en cuenta que los sinónimos de los tipos de datos también se considerarán una coincidencia (por ejemplo, DEC y NUMERIC coincidirán).

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

Si no existe ninguna función con la 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 fuente (SQLSTATE 53090).

AS TEMPLATE
Indica que esta sentencia se utilizará para crear una plantilla de función y no una función con código ejecutable.

NOT DETERMINISTIC o DETERMINISTIC
Especifica si la función devuelve los mismos resultados para argumentos de entrada idénticos. El valor por omisión es NOT DETERMINISTIC.

NOT DETERMINISTIC
Especifica que es posible que la función no devuelva el mismo resultado cada vez que la función se invoque con los mismos argumentos de entrada. La función depende de algunos valores de estado que afectan a los resultados. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL. Un ejemplo de una función que no es determinante es la que genera números aleatorios.

Una función que no es determinante puede recibir resultados incorrectos si se ejecuta por tareas paralelas.

DETERMINISTIC
Especifica que la función siempre devuelve el mismo resultado cada vez que se invoca con los mismos argumentos de entrada. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL. Un ejemplo de función determinante es la que calcula la raíz cuadrada del argumento de entrada.

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION
Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona. EXTERNAL ACTION se debe especificar implícita o explícitamente si el cuerpo de la rutina de SQL invoca una función que está definida con EXTERNAL ACTION (SQLSTATE 428C2).

Una función con acciones externas puede devolver resultados incorrectos si se ejecuta por tareas paralelas. Por ejemplo, si la función envía una nota para cada llamada inicial a esa función, se enviará una nota para cada tarea paralela, en lugar de una nota para la función.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

Normas

- 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 compara 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, tenga en cuenta las normas de promoción que afectarán a sus valores de entrada (consulte el apartado "Promoción de tipos de datos"). Por ejemplo, una constante que puede utilizarse como un valor de entrada puede tener un tipo de datos incorporado diferente al que se espera y, lo que es más significativo, es posible que no se promueva al tipo de datos que se espera. De acuerdo con las normas para la promoción, suele aconsejarse la utilización de los siguientes tipos de datos para parámetros:
  - INTEGER en lugar de SMALLINT
  - DOUBLE en lugar de REAL
  - VARCHAR en lugar de CHAR
  - VARGRAPHIC en lugar de GRAPHIC
- La creación de una función con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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 fuente 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 fuente es una función incorporada.

  - El usuario que define la función dispone de EXECUTE WITH GRANT OPTION para la función fuente.
  - La función es una plantilla.
- **CREATE FUNCTION MAPPING con funciones de tabla o fila:** las correlaciones de creación de función con funciones remotas que devuelven una tabla o una fila no están soportadas en las bases de datos federadas.

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
```

CREATE FUNCTION (escalar de SQL, tabla o fila)

La sentencia CREATE FUNCTION (escalar de SQL, tabla o fila) se utiliza para definir una función escalar de SQL, tabla o 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 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

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

- 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 en 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
- Autorización SYSADM o DBADM

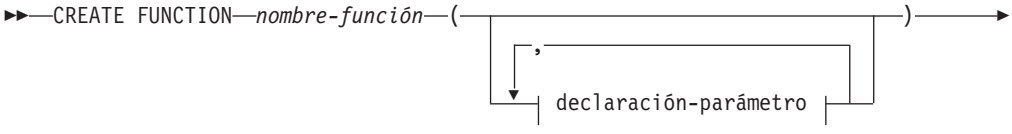
Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE FUNCTION.

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.

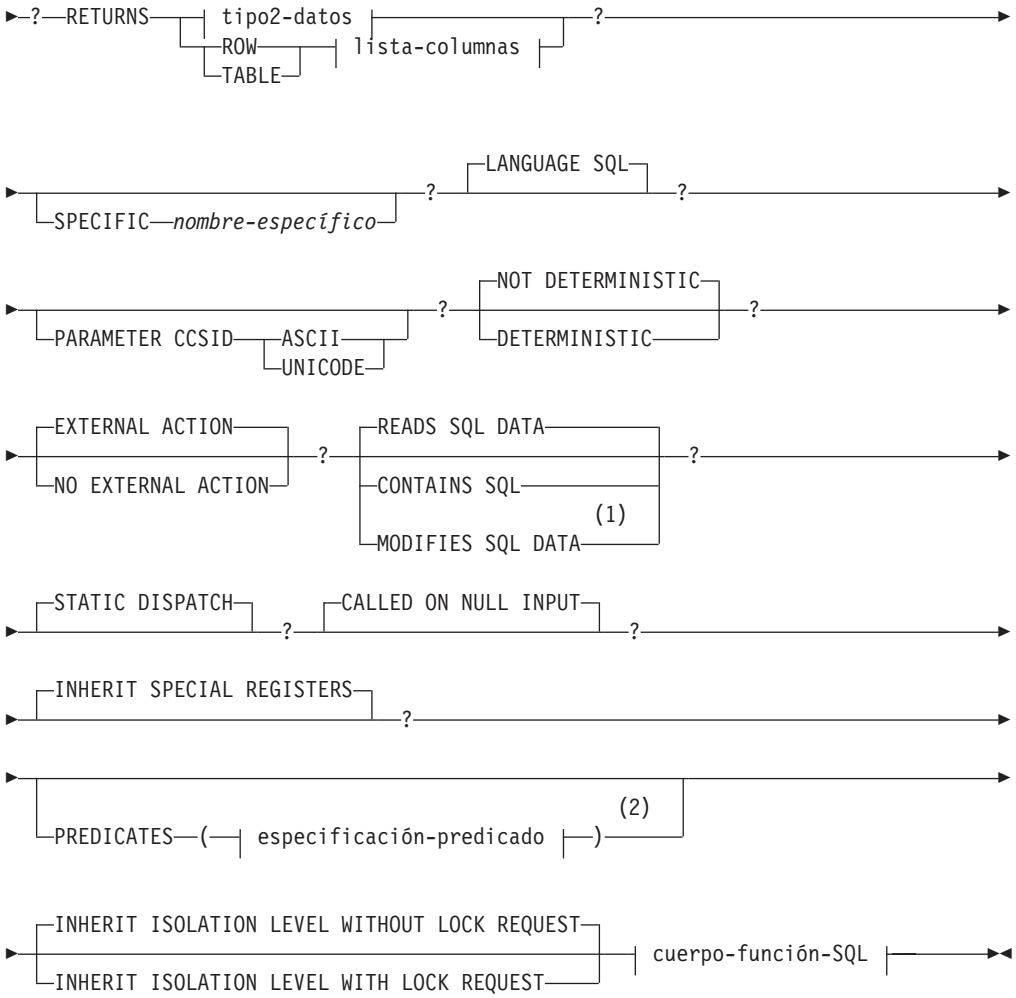
Si un usuario que define una función sólo puede crear la función porque dicho usuario dispone de autorización SYSADM, se le otorga autorización DBADM implícita para poder crear la función.

Si el ID de autorización de la sentencia no tiene la autorización SYSADM o DBADM, los privilegios del ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL que se especifican en el cuerpo de la función.

Sintaxis



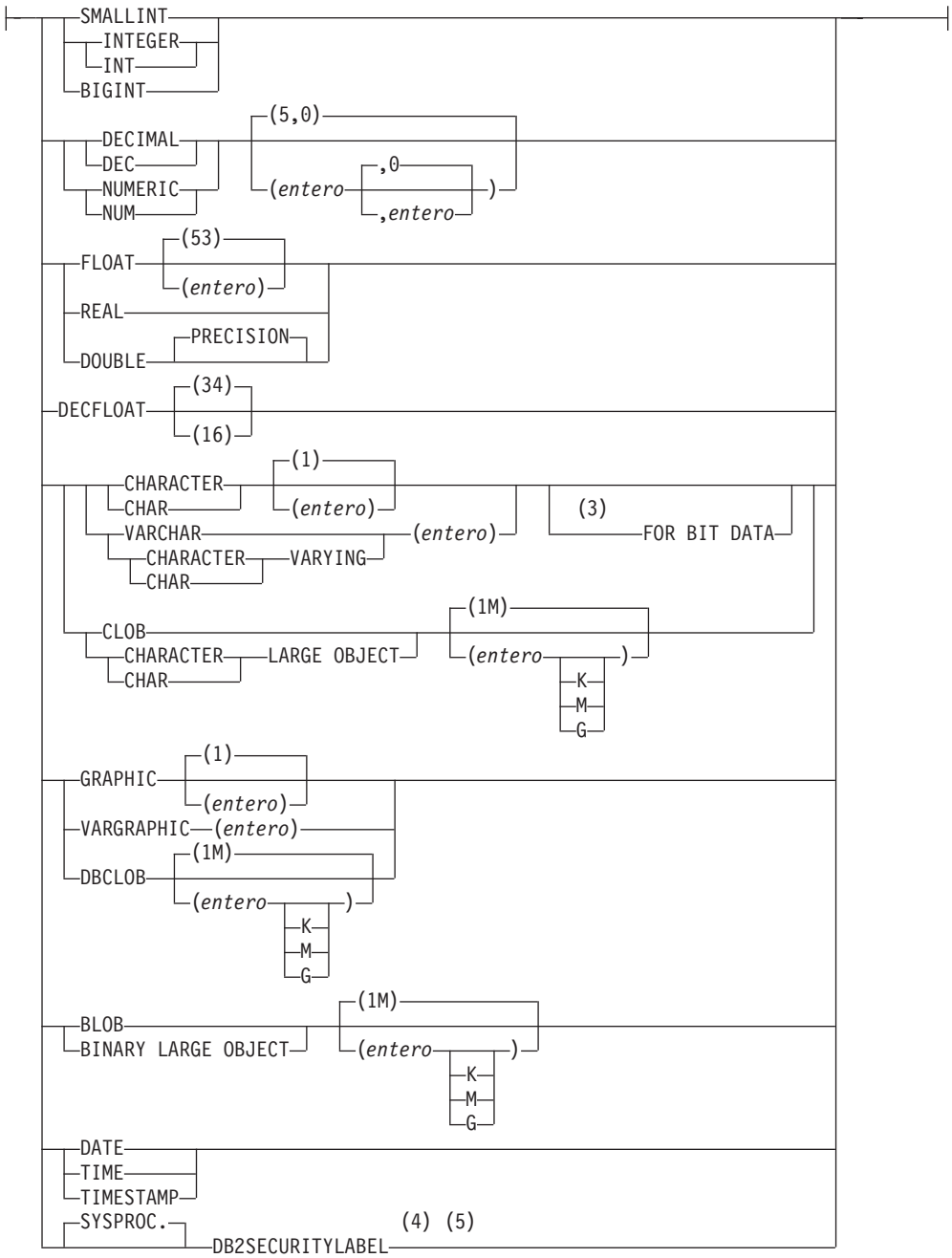
CREATE FUNCTION (escalar de SQL, tabla o fila)



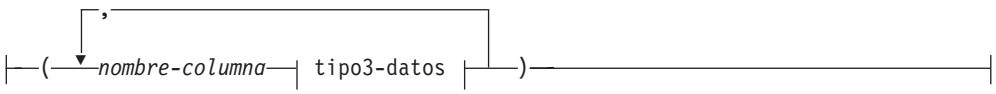
declaración-parámetro:

tipo1-datos, tipo2-datos, tipo3-datos:

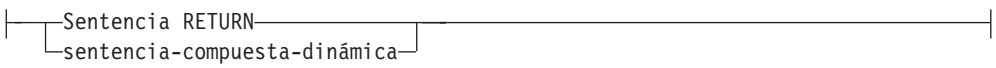
tipo-incorporado:



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*)

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

4 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

5 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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 sentencias de SQL estático, la opción de precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos y explícitos, junto con el número de parámetros y el tipo de datos de cada parámetro (sin tener en cuenta ningún atributo de longitud, precisión o escala del tipo de datos) no debe identificar una función descrita en el catálogo (SQLSTATE 42723). El nombre no calificado, junto con el número y el tipo de datos de los parámetros, que por supuesto es exclusivo en su esquema, no es necesario que lo sea en todos los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no pueden utilizarse como *nombre-función* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Se puede utilizar el mismo nombre para más de una función si hay alguna diferencia en la signatura de las funciones. Aunque no hay ninguna prohibición al respecto, una función de tabla externa definida por el usuario no debe tener el mismo nombre que una función incorporada.

(*declaración-parámetro***,...)**
Identifica el número de parámetros de entrada de la función, 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 (SQLSTATE 54023).

Existe la posibilidad de registrar una función que no tenga ningún parámetro. En este caso, sigue siendo necesaria la codificación de los paréntesis, sin incluir ningún tipo de datos. Por ejemplo:

```
CREATE FUNCTION WOOFER() ...
```

En un esquema, no se permite que dos funciones que se denominen igual tengan exactamente el mismo tipo para todos los parámetros correspondientes. Las longitudes, precisiones y escalas no se consideran en esta comparación de tipos. Por consiguiente, se considera que CHAR(8) y CHAR(35) son del mismo tipo, igual que DECIMAL(11,2) y DECIMAL (4,3), así como DECFLOAT(16) y DECFLOAT(34). Para una base de datos Unicode, se considera que CHAR(13) y GRAPHIC(8) son del mismo tipo. Hay una agrupación más profunda de los tipos que hace que se traten como el mismo tipo para esta finalidad como, por ejemplo, DECIMAL y NUMERIC. Una signatura duplicada devuelve un error (SQLSTATE 42723).

nombre-parámetro
Especifica un nombre para el parámetro de entrada. El nombre no puede ser el mismo que el de cualquier otro *nombre-parámetro* de la lista de parámetros (SQLSTATE 42734).

tipo1-datos
Especifica el tipo de datos del parámetro de entrada. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado, un tipo estructurado o un tipo de referencia.

Para obtener una descripción más completa de cada tipo de datos incorporado, consulte “CREATE TABLE”.

- No se pueden especificar tipos de datos de matriz o XML (SQLSTATE 42815).
- Un parámetro de tipo fecha y hora se pasa como tipo de datos de tipo carácter y los datos se pasan en formato ISO.

Para un tipo diferenciado definido por el usuario, los atributos de longitud, precisión o escala para el parámetro son los del tipo fuente del tipo diferenciado (los especificados en CREATE TYPE). Un parámetro de tipo diferenciado se pasa como tipo fuente del tipo diferenciado. Si el nombre del tipo diferenciado no está calificado, el gestor de base de datos resuelve el nombre de esquema buscando los esquemas en la vía de acceso de SQL.

Para un tipo de referencia, el parámetro se puede especificar como REF(*nombre-tipo*) si el parámetro no tiene ámbito. 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.

RETURNS
Esta cláusula obligatoria identifica el tipo de salida de la función.

tipo2-datos
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.

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si la función puede realizar una acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. Un ejemplo de una acción externa es enviar un mensaje o grabar un registro en un archivo. El valor por omisión es EXTERNAL ACTION.

EXTERNAL ACTION
Especifica que la función realiza una acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona.

NO EXTERNAL ACTION
Especifica que la función no realiza ninguna acción que cambie el estado de un objeto que el gestor de bases de datos no gestiona. El gestor de bases de datos utiliza esta información durante la optimización de las sentencias de SQL.

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 los registros especiales de hora y fecha,reflejan una propiedad de la sentencia que se está ejecutando y, por consiguiente, nunca se heredan de quien llama.

PREDICATES
Para los predicados que hacen uso de esta función, esta cláusula indica quiénes pueden explotar las extensiones de índice y pueden utilizar la cláusula opcional SELECTIVITY para la condición de búsqueda del predicado. Si se especifica la cláusula PREDICATES, la función debe definirse como DETERMINISTIC con NO EXTERNAL ACTION (SQLSTATE 42613). Si se especifica la cláusula PREDICATES y la base de datos no es Unicode, no se debe especificar PARAMETER CCSID UNICODE (SQLSTATE 42613).

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

- La resolución de las llamadas de función dentro del cuerpo de la función se realiza de acuerdo con la vía de acceso de SQL que está vigente para la sentencia CREATE FUNCTION y no cambia una vez creada la función.

- Si una función SQL contiene varias referencias a cualquiera de los registros especiales de fecha u hora, todas las referencias devuelven el mismo valor, y será el mismo valor devuelto por la invocación de registro en la sentencia donde se invocó la función.
- El cuerpo de una función SQL no puede contener una llamada recursiva a sí misma o a otra función o método que la llame, dado que una función de este tipo no puede existir para llamarla.
- 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.

- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 para 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

Ejemplos

Ejemplo 1: Defina una función escalar que devuelve la tangente de un valor utilizando las funciones de seno y coseno existentes.

```
CREATE FUNCTION TAN (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

Ejemplo 2: Defina una función de transformación para el tipo estructurado PERSON.

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

Ejemplo 3: Defina una tabla de función que muestra los empleados que trabajan en un número de departamento especificado.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
    SELECT EMPNO, LASTNAME, FIRSTNME
      FROM EMPLOYEE
      WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

Ejemplo 4: Defina la función de tabla del Ejemplo 3 con auditoría.

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  MODIFIES SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  BEGIN ATOMIC
    INSERT INTO AUDIT
    VALUES (USER,
            'Table: EMPLOYEE Prd: DEPTNO = ' CONCAT DEPTNO);
```

```
    RETURN
      SELECT EMPNO, LASTNAME, FIRSTNME
        FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
  END
```

Ejemplo 5: Defina una función escalar que invierta una serie de caracteres.

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(4000))
  RETURNS VARCHAR(4000)
  DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
  BEGIN ATOMIC
  DECLARE REVSTR, RESTSTR VARCHAR(4000) DEFAULT '';
  DECLARE LEN INT;
  IF INSTR IS NULL THEN
  RETURN NULL;
  END IF;
  SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
  WHILE LEN > 0 DO
  SET (REVSTR, RESTSTR, LEN)
    = (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
    SUBSTR(RESTSTR, 2, LEN - 1),
    LEN - 1);
  END WHILE;
  RETURN REVSTR;
END
```

CREATE FUNCTION MAPPING

La sentencia CREATE FUNCTION MAPPING se utiliza para:

- Definir una correlación entre una función o una plantilla de función, situadas en una base de datos federada, y una función de 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.

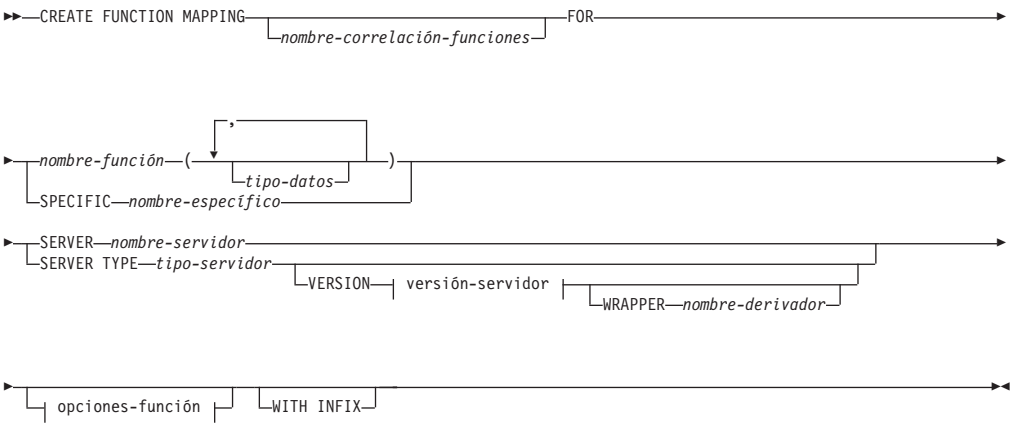
Invocación

Esta sentencia puede incorporarse 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).

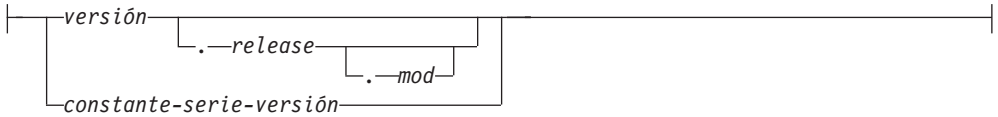
Autorización

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

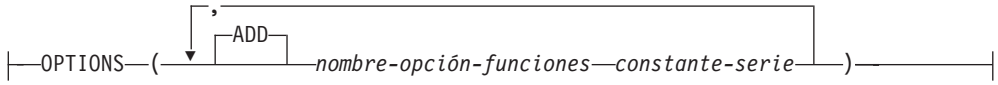
Sintaxis



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 DECFLOAT, LONG VARCHAR, LONG VARGRAPHIC, XML ni un tipo definido por el usuario.

Se pueden utilizar paréntesis vacíos en lugar de especificar la longitud, la precisión o la escala para tipos de datos con parámetros. Se recomienda utilizar los paréntesis vacíos para tipos de datos con parámetros; por ejemplo, CHAR(). Un tipo de datos con parámetros es cualquiera de los tipos de datos que se puede definir con una longitud, una escala o una precisión específica. Los tipos de datos con parámetros son los tipos de datos de serie y los tipos de datos decimales. Si especifica la longitud, la precisión o la escala, deben ser las mismas que las de la plantilla de función. Si se omiten los paréntesis, se utilizará la longitud por omisión para el tipo de datos (consulte la descripción de la sentencia CREATE TABLE).

SPECIFIC *nombre-específico*
Identifica la función o la plantilla de función desde la cual se debe realizar la correlación. Especifique *nombre-específico* para crear un nombre de función conveniente.

SERVER *nombre-servidor*
Nombra la fuente de datos que contiene la función que se está correlacionando.

SERVER TYPE *tipo-servidor*
Identifica el tipo de fuente de datos que contiene la función que se está correlacionando.

VERSION
Identifica la versión de la fuente de datos indicada por *tipo-servidor*.

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

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

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

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

WRAPPER *nombre-derivador*
Especifica el nombre del derivador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*.

OPTIONS
Indica las opciones de correlación de funciones que se deben habilitar.

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 de DB2 que se correlaciona con una función de fuente de datos, el optimizador desarrolla estrategias para invocar cualquiera de las dos funciones cuando se procesa la petición. Se invoca la función DB2 si para ello se necesita menos actividad general que para invocar la función de fuente de datos. De lo contrario, si la invocación de la función DB2 requiere más actividad general, se invocará a la función de fuente de datos.
- Si una petición distribuida hace referencia a una plantilla de función DB2 que se correlaciona con una función de fuente de datos, sólo se puede invocar la función de fuente de datos cuando se procesa la petición. La plantilla no puede invocarse porque no tiene ningún código ejecutable.
- Las correlaciones de funciones por omisión pueden pasar a ser no operativas inhabilitándolas (no se pueden desactivar). Para inhabilitar la correlación de funciones, codifique la sentencia CREATE FUNCTION MAPPING de modo que especifique el nombre de la función de DB2 en la correlación y establezca la opción DISABLE en 'Y'.
- Las funciones del esquema SYSIBM no tienen ningún nombre específico. Para alterar temporalmente la correlación de funciones por omisión para una función

del esquema SYSIBM, especifique *nombre-función* utilizando el calificador explícito SYSIBM; por ejemplo, SYSIBM.LENGTH().

- Una sentencia CREATE FUNCTION MAPPING de una unidad de trabajo (UOW) determinada no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- **CREATE FUNCTION MAPPING con funciones de tabla o fila:** las correlaciones de creación de función con funciones remotas que devuelven una tabla o una fila no están soportadas en las bases de datos federadas.

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)
  SERVER TYPE ORACLE
  OPTIONS (DISABLE 'Y')
```

Ejemplo 4: Correlacione la función federada UCASE(CHAR) con una UDF que se utiliza en una fuente de datos Oracle denominada ORACLE2. Incluya el número estimado de instrucciones por invocación de la UDF Oracle.

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4
  FOR SYSFUN.UCASE(CHAR)
  SERVER ORACLE2
  OPTIONS
    (REMOTE_NAME 'UPPERCASE',
    INSTS_PER_INVOC '1000')
```

CREATE HISTOGRAM TEMPLATE

La sentencia CREATE HISTOGRAM TEMPLATE define una plantilla que describe el tipo de histograma que puede utilizarse para alterar temporalmente uno o más histogramas por omisión de una clase de servicio o de una clase de trabajo.

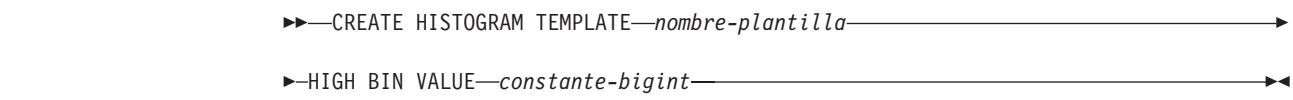
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-plantilla
Da nombre a la plantilla del histograma. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar una plantilla de histograma existente en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

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

Normas

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

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

Notas

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

Ejemplo

Cree una plantilla de histograma denominada LIFETIMETEMP en la clase de servicio PAYROLL de la superclase de servicio ADMIN que alterará temporalmente la plantilla de histograma de tiempo de vida de actividad por omisión con un valor binario superior nuevo de 90 000, que representa 90 000 microsegundos. Esta acción producirá un histograma con rangos binarios que se incrementarán exponencialmente y que finalizará con un binario cuyo rango irá de 90 000 al infinito.

```
CREATE HISTOGRAM TEMPLATE LIFETIMETEMP
  HIGH BIN VALUE 90000

CREATE SERVICE CLASS PAYROLL
  UNDER ADMIN ACTIVITY LIFETIME HISTOGRAM TEMPLATE LIFETIMETEMP
```

CREATE INDEX

La sentencia CREATE INDEX se utiliza para:

- Definir un índice en una tabla DB2. Un índice puede definirse en datos XML o en datos relacionales.
- Crear una especificación de índice (metadatos que indican al optimizador que una tabla de fuente de datos tiene un índice)

Invocación

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

Autorización

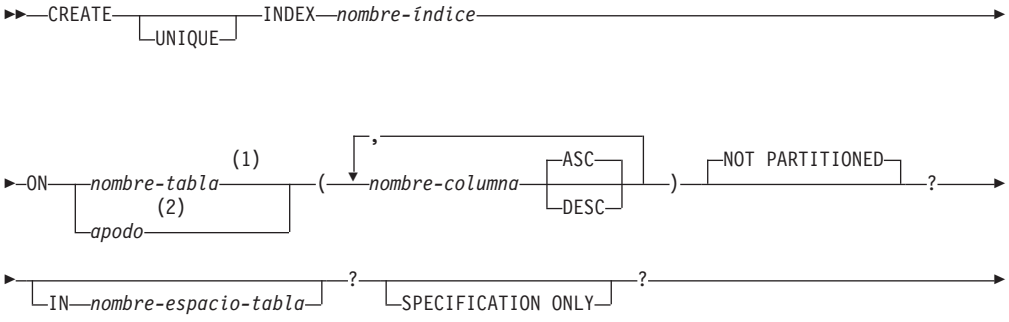
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

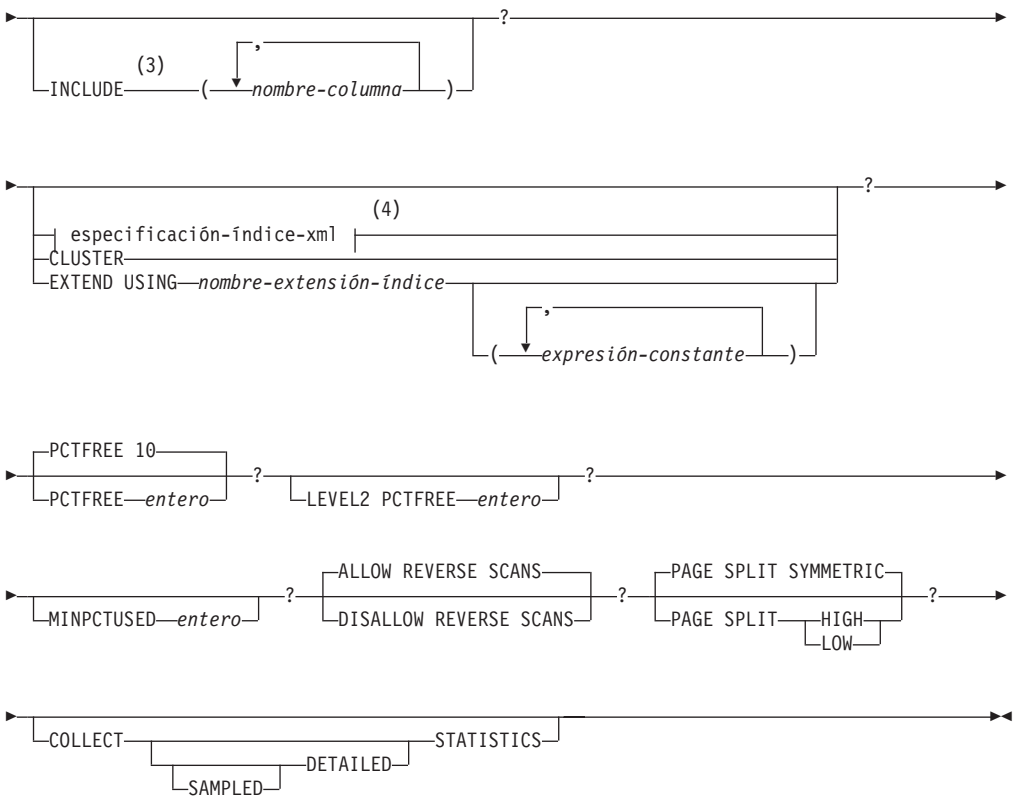
- Uno de los siguientes:
  - El privilegio CONTROL sobre la tabla o el apodo en el que está definido el índice
  - El privilegio INDEX en la tabla o el apodo en el que está definido el índice

  y uno de estos:
  - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito del índice no existe
  - Privilegio CREATEIN en el esquema, si el nombre de esquema del índice hace referencia a un esquema existente
- Autorización SYSADM o DBADM

No se necesita ningún privilegio explícito para crear un índice en una tabla temporal declarada.

Sintaxis





Notas:

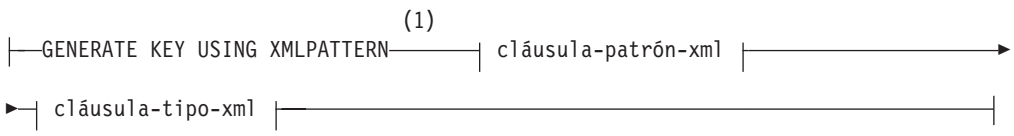
1 En un sistema federado, *nombre-tabla* debe identificar una tabla en la base de datos federada. No puede identificar una tabla de fuente de datos.

2 Si se especifica *apodo*, la sentencia CREATE INDEX crea una especificación de índice. En este caso, no se pueden especificar INCLUDE, *especificación-índice-xml*, CLUSTER, EXTEND USING, PCTFREE, MINPCTUSED, DISALLOW REVERSE SCANS, ALLOW REVERSE SCANS, PAGE SPLIT o COLLECT STATISTICS.

3 La cláusula INCLUDE sólo se puede especificar si se especifica UNIQUE.

4 Si se especifica *especificación-índice-xml*, no se puede especificar *nombre-columna* DESC, INCLUDE o CLUSTER.

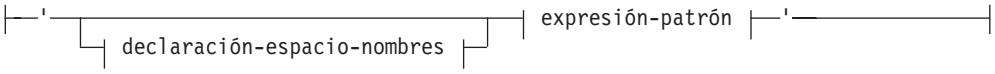
especificación-índice-xml:



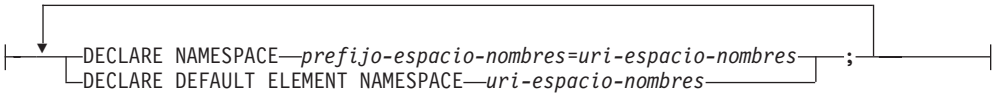
Notas:

1 Puede utilizarse la sintaxis alternativa GENERATE KEYS USING XMLPATTERN.

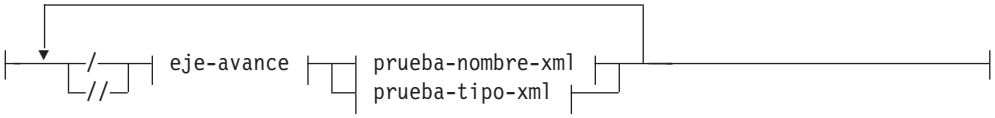
cláusula-patrón-xml:



declaración-espacio-nombres:



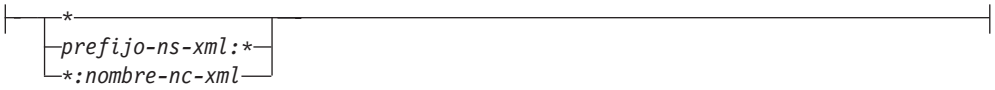
expresión-patrón:



eje-avance:

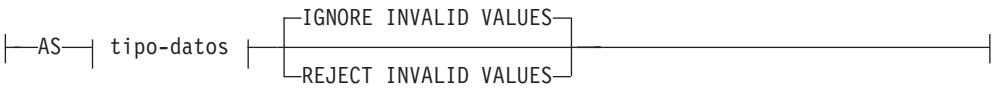
prueba-nombre-xml:

comodín-xml:



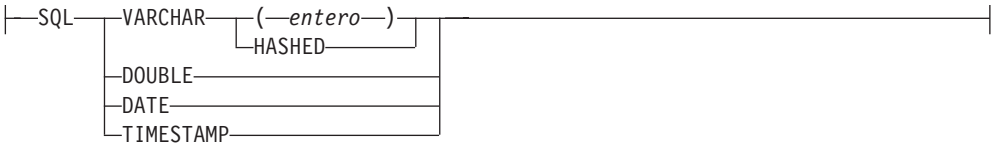
prueba-tipo-xml:

cláusula-tipo-xml:



tipo-datos:

tipo-datos-sql:



Descripción

UNIQUE

Si se especifica ON *nombre-tabla*, UNIQUE evita que la tabla contenga dos o más filas con el mismo valor de la clave de índice. La unicidad de valores se aplica al final de la sentencia de SQL que actualiza filas o inserta nuevas filas.

La exclusividad también se comprueba durante la ejecución de la sentencia CREATE INDEX. Si la tabla ya contiene filas con valores de clave duplicados, no se crea el índice.

Si el índice se encuentra en una columna XML (se trata de un índice de datos XML), la unicidad se aplica a los valores para los que se ha especificado *expresión-patrón* para todas las filas de la tabla. La unicidad se aplica en cada valor después de convertir el valor al *tipo-datos-sql* especificado. Puesto que la conversión al *tipo-datos-sql* especificado puede dar lugar a una pérdida de precisión o de rango, o a que distintos valores se generen aleatoriamente en el mismo valor clave, varios valores que parecen ser exclusivos en el documento XML pueden dar lugar a errores de claves duplicadas. La unicidad de series de caracteres depende de la semántica de XQuery donde los blancos de cola son significativos. Por lo tanto, los valores que serían duplicados en SQL pero que difieren en blancos de cola se consideran valores exclusivos en un índice sobre datos XML.

Cuando se utiliza UNIQUE, los valores nulos se tratan como cualquier otro valor. Por ejemplo, si la clave es una sola columna que puede contener valores nulos, esa columna no puede contener más de un valor nulo.

Si se especifica la opción UNIQUE, y la tabla tiene una clave de distribución, las columnas de la clave de índice deben ser un superconjunto de la clave de distribución. Es decir, las columnas especificadas para una clave de índice exclusiva debe incluir todas las columnas de la clave de distribución (SQLSTATE 42997).

Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE).

Si se especifica ON *apodo*, sólo se debe especificar UNIQUE si los datos para la clave de índice contienen valores exclusivos para cada fila de la tabla de fuente de datos. No se comprobará la exclusividad.

Para un índice sobre datos XML, UNIQUE sólo puede especificarse si la *expresión-patrón* especifica una única vía de acceso completa y no contiene un eje de tipo descendant o descendant-or-self, "//", un *comodín-xml*, *node()*, o *processing-instruction()* (SQLSTATE 429BS).

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 64 columnas. Si *nombre-tabla* es una tabla con tipo, pueden especificarse hasta 63 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 las longitudes de las columnas especificadas no debe ser superior al límite de longitud de claves de índice para el tamaño de página. Para los límites de longitud de clave, consulte “Límites de SQL”. Si *nombre-tabla* es una tabla con tipo, la longitud de la clave de índice debe ser todavía 4 bytes menos. Tenga en cuenta que la actividad general del sistema puede reducir todavía más este límite de longitud, que varía en función del tipo de datos de la columna y de si la columna puede contener nulos. Para obtener más información sobre cómo la actividad general afecta a este límite, consulte “Número total de bytes” en “CREATE TABLE”.

Tenga en cuenta que la actividad general del sistema puede reducir esta longitud, que varía en función del tipo de datos de la columna y en función de

si puede contener nulos. Para obtener más información sobre cómo la actividad general afecta a este límite, consulte “Número total de bytes” en “CREATE TABLE”.

No se puede utilizar ninguna columna LOB, LONG VARCHAR, LONG VARGRAPHIC, ni ninguna columna de tipo distinto basada en LOB, LONG VARCHAR o LONG VARGRAPHIC, como parte de un índice, aun cuando el atributo de longitud de la columna sea lo suficientemente pequeño para caber dentro del límite de longitud de la clave de índice para el tamaño de página (SQLSTATE 54008). Las columnas de tipo estructurado sólo se pueden especificar si también se especifica la cláusula EXTEND USING (SQLSTATE 42962). Si se especifica la cláusula EXTEND USING, sólo puede especificarse una columna, y el tipo de la columna debe ser un tipo estructurado o un tipo diferenciado no basado en LOB, LONG VARCHAR o LONG VARGRAPHIC (SQLSTATE 42997).

Si un índice tiene sólo una columna y los datos de dicha columna son de tipo XML, y también se especifica la cláusula GENERATE KEY USING XMLPATTERN, el índice es un índice sobre datos XML. Una columna con datos de tipo XML sólo puede especificarse si también se especifica la cláusula GENERATE KEY USING XMLPATTERN (SQLSTATE 42962). Si se especifica la cláusula GENERATE KEY USING XMLPATTERN, sólo se puede especificar una columna y el tipo de la columna debe ser XML.

ASC
Especifica que las entradas de índice se deben mantener en el orden ascendente de los valores de columna; este es el valor por omisión. ASC no se puede especificar para índices que están definidos con EXTEND USING (SQLSTATE 42601).

DESC
Especifica que las entradas de índice se deben mantener en el orden descendente de los valores de columna. DESC no se puede especificar para índices que están definidos con EXTEND USING o si el índice es un índice sobre datos XML (SQLSTATE 42601).

NOT PARTITIONED
Indica que sólo debe crearse un índice que se expanda por todas las particiones de datos definidas para la tabla. El *nombre-tabla* debe identificar una tabla definida con particiones de datos (SQLSTATE 53036).

IN *nombre-espacio-tablas*
Especifica el espacio de tablas en el que debe crearse el índice. Esta cláusula sólo puede utilizarse para índices de tablas particionadas. Esta cláusula puede especificarse aun cuando se haya especificado la cláusula INDEX IN al crear la tabla. Ésta se grabará encima de aquella cláusula.

El espacio de tablas especificado por *nombre-espacio-tabla* debe estar en el mismo grupo de particiones de base de datos que los espacios de tablas de datos para la tabla y gestionar el espacio del mismo modo que los demás espacios de tablas de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE.

Si no se especifica la cláusula IN, el índice se crea en el espacio de tablas que se ha especificado con la cláusula INDEX IN en la sentencia CREATE TABLE. Si no se ha especificado ninguna cláusula INDEX IN, se utiliza el espacio de tablas de la primera partición de datos visibles o adjunta de la tabla. Ésta es la primera partición de la lista de particiones de datos almacenadas en la base de

las especificaciones de rango. Si no se especifica la cláusula IN, el ID de autorización de la sentencia no será necesario para poseer el privilegio USE en el espacio de tablas por omisión.

SPECIFICATION ONLY
Indica que esta sentencia será utilizada para crear una especificación de índice que se suscribe a la tabla de fuente de datos a la que se hace referencia mediante *apodo*. Se debe especificar SPECIFICATION ONLY si se especifica *apodo* (SQLSTATE 42601). No se puede especificar si se especifica *nombre-tabla* (SQLSTATE 42601).

Si la especificación de índice se aplica a un índice que es exclusivo, DB2 no verifica si los valores de la columna de la tabla remota son exclusivos. Si los valores de la columna remota no son exclusivos, es posible que las consultas realizadas sobre el apodo que incluyan la columna del índice devuelvan datos incorrectos o errores.

Esta cláusula no 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). Debe especificarse UNIQUE si se especifica INCLUDE (SQLSTATE 42613). Los límites para el número de columnas y la suma de los atributos de longitud se aplican a todas las columnas del índice y la clave de unicidad.

Esta cláusula no 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, si se ha especificado *apodo*, o si el índice es un índice de valores XML (SQLSTATE 42601).

especificación-índice-xml
Especifica cómo se generan las claves de índice a partir de documentos XML que están almacenados en una columna XML. No se puede especificar *especificación-índice-xml* si hay más de una columna de índice o si la columna no tiene el tipo de datos XML.

Esta cláusula sólo se aplica a columnas XML (SQLSTATE 429BS).

GENERATE KEY USING XMLPATTERN *cláusula-patrón-xml*
Especifica las partes de un documento XML que deben indexarse. Los valores de patrón XML son los valores indexados generados por la *cláusula-patrón-xml*. El índice no soporta los nodos de tipo de datos de lista. Si un nodo es calificado por la *cláusula-patrón-xml* y existe un esquema XML que especifica que el nodo es de tipo de datos de lista, el nodo de

tipo de datos de lista no se puede indexar (SQLSTATE 23526 para sentencias CREATE INDEX o SQLSTATE 23525 para sentencias INSERT y UPDATE).

cláusula-patrón-xml
Contiene una expresión de patrón que identifica los nodos que deben indexarse. Consta de una *declaración-espacio-nombres* opcional y de una *expresión-patrón* necesaria.

declaración-espacio-nombres
Si la expresión de patrón contiene nombres calificados, debe especificarse una *declaración-espacio-nombres* para definir prefijos de espacios de nombres. Puede definirse un espacio de nombres por omisión para nombres sin calificar.

DECLARE NAMESPACE *prefijo-espacio-nombres*=*uri-espacio-nombres*
Correlaciona el *prefijo-espacio-nombres*, que es un nombreNC, con el *uri-espacio-nombres*, que es un literal de cadena. La *declaración-espacio-nombres* puede contener varias correlaciones *prefijo-espacio-nombres*-a-*uri-espacio-nombres*. El *prefijo-espacio-nombres* debe ser exclusivo dentro de la lista de *declaración-espacio-nombres* (SQLSTATE 10503).

DECLARE DEFAULT ELEMENT NAMESPACE
uri-espacio-nombres
Declara el URI de espacio de nombres por omisión para los tipos o los nombres de elemento sin calificar. Si no se ha declarado ningún espacio de nombres por omisión, los tipos y los nombres de elemento sin calificar no se encuentran en ningún espacio de nombres. Sólo se puede declarar un espacio de nombres por omisión (SQLSTATE 10502).

expresión-patrón
Especifica los nodos de un documento XML que están indexados. La *expresión-patrón* puede contener caracteres comodín (*). Es parecido a una expresión de vía de acceso en XQuery, pero da soporte a un subconjunto del lenguaje XQuery al que DB2 da soporte.

/ (barra inclinada)
Separa los pasos de la expresión de vía de acceso.

// (dos barras inclinadas)
Es la sintaxis abreviada para /descendant-or-self::node()/. Si se especifica UNIQUE, no se puede utilizar *// (dos barras inclinadas)*.

eje-avance

child::
Especifica los hijos del nodo de contexto. Es el valor por omisión, si no se especifica ningún otro eje de avance.

@ Especifica los atributos del nodo de contexto. Ésta es la sintaxis abreviada del atributo::.

attribute::
Especifica los atributos del nodo de contexto.

descendant::
Especifica los descendientes del nodo de contexto. No se puede utilizar *descendant::* si también se especifica UNIQUE.

self::
Especifica el nodo de contexto propiamente dicho.

descendant-or-self::
Especifica el nodo de contexto y los descendientes del nodo de contexto. No se puede utilizar descendant-or-self:: si también se especifica UNIQUE.

prueba-nombre-xml
Especifica el nombre del nodo para el paso en la vía utilizando un nombre XML calificado (nombre-xml-c) o un comodín (comodín-xml).

nombre-nc-xml
Nombre XML según define XML 1.0. No puede incluir el carácter de dos puntos.

nombre-xml-c
Especifica un nombre XML calificado (conocido por QName) que puede tener dos formas posibles:

- prefijo-ns-xml:nombre-nc-xml, donde prefijo-ns-xml es un nombre-nc-xml que identifica un espacio de nombres del ámbito.
- nombre-nc-xml, que indica que el espacio de nombres por omisión debe aplicarse como prefijo-ns-xml implícito.

comodín-xml
Especifica un nombre-xml-c como un comodín que puede tener tres formas posibles:

- * (un único carácter de asterisco) indica cualquier nombre-xml-calificado
- *prefijo-ns-xml*:* indica cualquier nombre-nc-xml dentro del espacio de nombres especificado
- *:*nombre-nc-xml* indica un nombre XML específico dentro de cualquier espacio de nombres del ámbito

No se puede utilizar *comodín-xml* si también se especifica UNIQUE.

prueba-tipo-xml
Utilice estas opciones para especificar con qué tipos de nodo coincide el patrón. Las opciones disponibles son:

node()
Coincide con cualquier nodo. No se puede utilizar *node()* si también se especifica UNIQUE.

text()
Coincide con cualquier nodo de texto.

comment()
Coincide con cualquier nodo de comentario.

processing-instruction()
Coincide con cualquier nodo de instrucción de proceso. No se puede utilizar processing-instruction() si también se especifica UNIQUE.

cláusula-tipo-xml

AS *tipo-datos*
Especifica el tipo de datos a los que se convierten los valores indexados antes de almacenarlos. Los valores se convierten al tipo de datos de índice XML que corresponde al tipo de datos de índice SQL especificado.

Tabla 17. Tipos de datos de índice correspondientes

Tipo de datos de índice XML	Tipo de datos de índice SQL
xs:string	VARCHAR(*entero*), VARCHAR HASHED
xs:double	DOUBLE
xs:date	DATE
xs:dateTime	TIMESTAMP

Para VARCHAR(*entero*) y VARCHAR HASHED, el valor se convierte a un valor xs:string utilizando la función XQuery fn:string. El atributo de longitud de VARCHAR(*entero*) se aplica como restricción al valor xs:string resultante. Un tipo de datos de índice SQL de VARCHAR HASHED aplica un algoritmo hash al valor xs:string resultante para generar un código hash que se inserta en el índice.

Para índices que utilizan los tipos de datos DOUBLE, DATE y TIMESTAMP, el valor se convierte al tipo de datos de índice XML mediante la expresión de conversión de XQuery.

Si el índice es exclusivo, la unicidad del valor se aplica después de convertir el valor al tipo indexado.

tipo-datos
Se da soporte al tipo de datos siguiente:

tipo-datos-sql
Los tipos de datos SQL son:

VARCHAR(*entero***)**
Si se especifica este formato de VARCHAR, DB2 utiliza *entero* como restricción. Si los nodos de documento que deben indexarse tienen valores superiores al *entero*, los documentos no se insertan en la tabla si el índice ya existe. Si el índice no existe, el índice no se crea. *entero* es un valor entre 1 y un máximo que depende del tamaño de la página. La Tabla 18 muestra el valor máximo para cada tamaño de página.

Tabla 18. Longitud máxima de los nodos del documento por tamaño de página

Tamaño de página	Longitud máxima del nodo de documento (bytes)
4 KB	817
8 KB	1841
16 KB	3889

Tabla 18. Longitud máxima de los nodos del documento por tamaño de página (continuación)

Tamaño de página	Longitud máxima del nodo de documento (bytes)
32 KB	7985

La semántica XQuery se utiliza para comparaciones de series, donde los blancos de cola son significativos. Esto difiere de la semántica SQL, donde los blancos de cola no son significantes durante las comparaciones.

VARCHAR HASHED
Especifique VARCHAR HASHED para gestionar la indexación de series de caracteres de longitud arbitraria. La longitud de una serie indexada no tiene límite. DB2 genera un código hash de ocho bytes para toda la serie. Los índices que utilizan estas series de caracteres con códigos hash sólo pueden utilizarse para búsquedas de igualdad. La semántica XQuery se utiliza para comparaciones de igualdad de series, donde los blancos de cola son significativos. Esto difiere de la semántica SQL, donde los blancos de cola no son significantes durante las comparaciones. El hash de la serie preserva la igualdad de la semántica de XQuery y no la semántica de SQL.

DOUBLE
Especifica que se utiliza el tipo de datos DOUBLE para indexar valores numéricos. Los tipos de decimal no definido y los enteros de 64 bits pueden perder precisión cuando se almacenan como un valor DOUBLE. Los valores para DOUBLE pueden incluir los valores numéricos especiales *NaN*, *INF*, *-INF*, *+0* y *-0*, aunque el tipo de datos de SQL DOUBLE no soporte estos valores.

DATE
Especifica que se utiliza el tipo de datos DATE para indexar valores XML. Tenga en cuenta que el tipo de datos de esquema XML para xs:date permite una mayor precisión que el tipo de datos SQL. Si se encuentra un valor fuera de rango, se devuelve un error.

TIMESTAMP
Especifica que se utiliza el tipo de datos TIMESTAMP para indexar valores XML. Tenga en cuenta que el tipo de datos de esquema XML para xs:dateTime permite una mayor precisión que el tipo de datos SQL. Si se encuentra un valor fuera de rango, se devuelve un error.

IGNORE INVALID VALUES
Especifica que los valores de patrón XML que no son válidos para el tipo de datos XML de índice de destino se ignoran y que la sentencia CREATE INDEX no indexa los valores correspondientes de los documentos XML almacenados. Por omisión, los valores no

válidos se ignoran. Durante las operaciones de inserción y actualización, los valores de patrón XML no válidos no se indexan, pero los documentos XML se siguen insertando en la tabla. No se genera ningún error o aviso, porque la especificación de estos tipos de datos no es una restricción en los valores de patrón XML (las expresiones XQuery que buscan el tipo de datos de índice XML específico no tendrán en cuenta estos valores).

El índice sólo puede ignorar valores de patrón XML no válidos para el tipo de datos XML de índice. Los valores válidos se deben ajustar a la representación de DB2 del valor para el tipo de datos XML de índice o se devolverá un error. Un valor de patrón XML asociado con el tipo de datos XML de índice xs:string siempre es válido. Sin embargo, la restricción de longitud adicional del tipo de datos SQL de índice asociado VARCHAR(*entero*) puede seguir generando un error, si se excede la longitud máxima. Si se devuelve un error, no se insertan ni se actualizan datos XML en la tabla si el índice ya existe (SQLSTATE 23525). Si el índice no existe, no se crea ningún índice (SQLSTATE 23526).

REJECT INVALID VALUES
Especifica que todos los valores de patrón XML deben ser válidos para el tipo de datos XML de índice. Si no se puede convertir ningún valor de patrón XML al tipo de datos XML de índice, se devuelve un error. Los datos XML no se insertan ni se actualizan en la tabla si el índice ya existe (SQLSTATE 23525). Si el índice no existe, no se crea ningún índice (SQLSTATE 23526).

CLUSTER
Especifica que el índice es el índice de 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 especifica *apodo* o si el índice es un índice sobre datos XML (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 tener una longitud de más de 32.768 en la página de códigos de base de datos (SQLSTATE 22001).

PCTFREE *entero*
Especifica qué porcentaje de cada página de índice se va a dejar como espacio libre cuando se cree el índice. La primera entrada de una página se añade sin restricciones. Cuando se colocan entradas adicionales en una página de índice, en cada página se deja, como mínimo, un *entero* como porcentaje de espacio libre. El valor de *entero* entra en un rango que va de 0 a 99. Si se especifica un valor superior a 10, sólo se dejará un 10 por ciento de espacio libre en las páginas que no son hojas. 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. 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 el índice sólo permite búsquedas o exploraciones hacia delante del índice en el orden definido en el momento de crear el índice.

DISALLOW REVERSE SCANS no puede especificarse junto con *apodo* (SQLSTATE 42601).

ALLOW REVERSE SCANS
Especifica que un índice permite búsquedas hacia delante y hacia atrás; es decir, permite explorar el índice en el orden definido en el momento de su creación y en el orden opuesto.

ALLOW REVERSE SCANS no puede especificarse junto con *apodo* (SQLSTATE 42601).

PAGE SPLIT
Especifica un comportamiento de división de índice. El valor por omisión es SYMMETRIC.

SYMMETRIC
Especifica que las páginas se deben dividir por la mitad.

HIGH
Especifica un comportamiento de división de páginas de índice que utiliza espacio de páginas de índice de forma eficiente cuando los valores de las claves de índice que se insertan siguen un determinado patrón. Para un subconjunto de valores de claves de índices, la columna o columnas que hay más a la izquierda del índice deben contener el mismo valor y la columna o columnas que hay más a la derecha del índice deben contener valores que aumentan con cada inserción. 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. 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 falla (SQLSTATE 01550) cuando se intenta crear un índice que coincide con un índice ya existente.

Se utilizan diversos factores para determinar si dos índices coinciden. Estos factores se combinan de varias maneras diferentes para conformar las normas que determinan si dos índices coinciden. Los factores siguientes se utilizan para determinar si dos índices coinciden:

1. Los conjuntos de columnas de índice, incluida cualquier columna INCLUDE, son iguales en ambos índices.
2. El orden de las columnas de claves de índice, incluida cualquier columna INCLUDE, es el mismo en ambos índices.
3. Las columnas de claves del nuevo índice son las mismas o un superconjunto de las columnas de claves en el índice existente.
4. Los atributos de orden de las columnas son los mismos en ambos índices.
5. El índice existente es exclusivo.
6. Ambos índices son no exclusivos.

Las combinaciones siguientes de estos factores forman las normas que determinan cuando dos índices se consideran duplicados:

- 1 + 2 + 4 + 5
- 1 + 2 + 4 + 6
- 1 + 2 + 3 + 5

Excepción: Para los índices de datos XML, dos descripciones de índice no se consideran duplicadas si los nombres del índice son distintos, aun cuando la columna XML indexada, los patrones XML y los tipos de datos, incluidas sus opciones, sean idénticos.

- Los índices exclusivos de las MQT mantenidas por el sistema no reciben soporte (SQLSTATE 42809).
- Las opciones COLLECT STATISTICS no reciben soporte si se ha especificado un apodo (SQLSTATE 42601).

Notas

- Los índices de datos XML no dan soporte al acceso de escritura simultánea mientras se ejecuta CREATE INDEX.
- Sólo para índices relacionales: 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 en las que se aplican las especificaciones.
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para z/OS:
    - Se tolera y se pasa por alto la sintaxis siguiente:
      - CLOSE
      - DEFINE
      - FREEPAGE
      - GBPCACHE
      - PIECESIZE
      - TYPE 2
      - utilizando-bloque
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - COPY NO
      - DEFER NO

Ejemplos

Ejemplo 1: Cree un índice denominado UNIQUE_NAM en la tabla PROJECT. La finalidad de este índice consiste en garantizar que en la misma tabla no habrá dos entradas que tengan el mismo valor para el nombre del proyecto (PROJNAME). Las entradas de índice deben estar en orden ascendente.

```
CREATE UNIQUE INDEX UNIQUE_NAM
   ON PROJECT(PROJNAME)
```

Ejemplo 2: Cree un índice denominado JOB_BY_DPT en la tabla EMPLOYEE. Disponga las entradas de índice en orden ascendente por el título de los trabajos (JOB) dentro de cada departamento.

```
CREATE INDEX JOB_BY_DPT
   ON EMPLOYEE (WORKDEPT, JOB)
```

Ejemplo 3: El apodo EMPLOYEE hace referencia a una tabla de fuente de datos denominada CURRENT_EMP. Tras la creación de este apodo, se ha definido un índice en CURRENT_EMP. Las columnas escogidas para la clave de índice fueron WORKDEBT y JOB. Cree una especificación de índice que describa a este índice. Mediante esta especificación, el optimizador sabrá que el índice existe y cuál es su clave. Con esta información, el optimizador puede mejorar su estrategia de acceso a la tabla.

```
CREATE UNIQUE INDEX JOB_BY_DEPT
  ON EMPLOYEE (WORKDEPT, JOB)
  SPECIFICATION ONLY
```

Ejemplo 4: Cree un tipo de índice ampliado denominado SPATIAL_INDEX en una ubicación de columna de tipo estructurado. La descripción de la extensión de índice GRID_EXTENSION se utiliza para mantener SPATIAL_INDEX. El literal se proporciona a GRID_EXTENSION para crear el tamaño de cuadrícula de índice.

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
  EXTEND USING (GRID_EXTENSION (x'00010010001000100040001 0'))
```

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
```

Ejemplo 8: Crea un índice exclusivo denominado A_IDX en una tabla particionada denominada MYNUMBERDATA en el espacio de tablas IDX_TBSP.

```
CREATE UNIQUE INDEX A_IDX ON MYNUMBERDATA (A) IN IDX_TBSP
```

Ejemplo 9: Cree un índice no exclusivo denominado B_IDX en una tabla particionada denominada MYNUMBERDATA en el espacio de tablas IDX_TBSP.

```
CREATE INDEX B_IDX ON MYNUMBERDATA (B)
  NOT PARTITIONED IN IDX_TBSP
```

Ejemplo 10: Cree un índice sobre datos XML en una tabla denominada COMPANYINFO, que contiene una columna XML denominada COMPANYDOCS. La columna XML COMPANYDOCS contiene un número elevado de documentos XML parecidos al siguiente:

```
<company name="Empresa1">
  <emp id="31201" salary="60000" gender="Mujer">
    <name>
      <first>Laura</first>
      <last>Brown</last>
    </name>
    <dept id="M25">
      Finanzas
    </dept>
  </emp>
</company>
```

Los usuarios de la tabla COMPANYINFO a menudo deben recuperar información de los empleados utilizando el ID de empleado. Un índice como el siguiente puede hacer que la recuperación de datos sea más eficiente.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
  GENERATE KEY USING XMLPATTERN '/company/emp/@id'
    AS SQL DOUBLE
```

Ejemplo 11: Por lo general, el índice siguiente es equivalente desde el punto de vista lógico al índice creado en el ejemplo anterior, excepto que utiliza una sintaxis abreviada.

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
  GENERATE KEY USING XMLPATTERN '/child::company/child::emp/attribute::id'
    AS SQL DOUBLE
```

Ejemplo 12: Cree un índice en una columna denominada DOC, indexando solamente el título del manual como VARCHAR(100). Puesto que el libro del manual debe ser exclusivo entre todos los manuales, el índice también debe ser exclusivo.

```
CREATE UNIQUE INDEX MYDOCSIDX ON MYDOCS(DOC)
  GENERATE KEY USING XMLPATTERN '/book/title'
    AS SQL VARCHAR(100)
```

Ejemplo 13: Cree un índice en una columna denominada DOC, indexando el número del capítulo como DOUBLE. Este ejemplo incluye declaraciones de espacios de nombres.

```
CREATE INDEX MYDOCSIDX ON MYDOCS(DOC)
  GENERATE KEY USING XMLPATTERN
    'declare namespace b="http://www.foobar.com/book/";
     declare namespace c="http://acme.org/chapters";
       /b:book/c:chapter/@number'
    AS SQL DOUBLE
```

Ejemplo 14: Cree un índice exclusivo denominado IDXPROJEST en la tabla PROJECT e incluya la columna PRSTAFF para permitir el acceso de sólo índice de la información de dotación media de personal estimada.

```
CREATE UNIQUE INDEX IDXPROJEST ON PROJECT (PROJNO) INCLUDE (PRSTAFF)
```

SCREATE INDEX EXTENSION

La sentencia CREATE INDEX EXTENSION define un objeto de extensión para utilizar con índices de tablas que tienen columnas de tipo estructurado o de tipo diferenciado.

Invocación

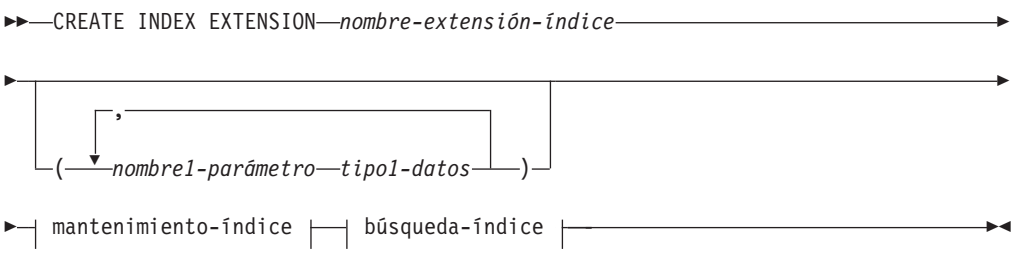
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

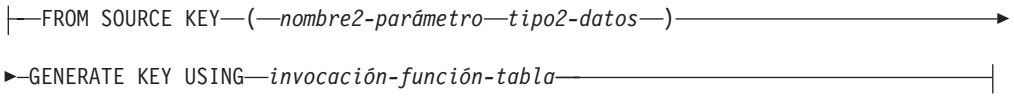
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema de la extensión de índice no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la extensión de índice hace referencia a un esquema existente
- Autorización SYSADM o DBADM

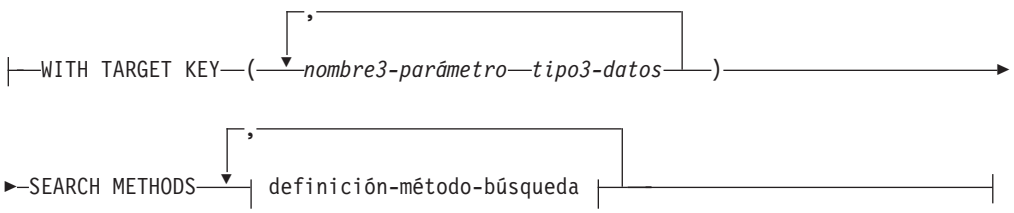
Sintaxis



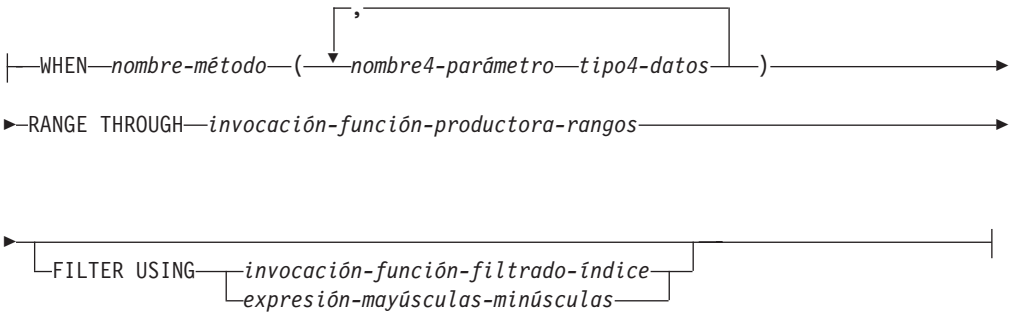
mantenimiento-índices:



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).

tipo1-datos
Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que la extensión de índice espera recibir. Los únicos tipos de datos SQL que se pueden especificar son los que se pueden utilizar como constantes, tales como VARCHAR, INTEGER, DECIMAL, DOUBLE o VARGRAPHIC (SQLSTATE 429B5). No se puede especificar el tipo de datos de coma flotante decimal (SQLSTATE 429B5). El valor de parámetro que la extensión de índice recibe al ejecutarse CREATE INDEX debe coincidir exactamente con *tipo-datos1*, incluida la longitud, precisión y escala (SQLSTATE 428E0).

mantenimiento-índices
Especifica cómo se realiza el mantenimiento de las claves de índice de una columna de tipo estructurado o de tipo diferenciado. El mantenimiento de índices es el proceso de transformar la columna fuente en una clave destino. El proceso de transformación se define utilizando una función de tabla que previamente se ha definido en la base de datos.

FROM SOURCE KEY (*nombre2-parámetro tipo2-datos***)**
Especifica un tipo de datos estructurado o tipo diferenciado para la columna de la clave fuente que está soportada por esta extensión de índice.

nombre-parámetro2
Identifica el parámetro que está asociado a la columna de la clave fuente. Una columna de clave fuente es la columna de clave de índice (definida en la sentencia CREATE INDEX) que tiene el mismo tipo de datos que *tipo-datos2*.

tipo2-datos
Especifica el tipo de datos para *nombre2-parámetro*; *tipo2-datos* debe ser un tipo estructurado definido por el usuario o un tipo diferenciado cuya fuente no sea LOB, XML, DECFLOAT, LONG VARCHAR o 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, XML, 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)
- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E3)
- Debe devolver columnas cuyos tipos de datos siguen las restricciones para los tipos de datos de columnas de un índice definido sin la cláusula EXTEND USING

Si un argumento invoca otra operación o rutina, debe ser un método observador (SQLSTATE 428E3).

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

búsqueda-índice
Especifica cómo se realiza la búsqueda proporcionando una correlación de los argumentos de búsqueda con rangos de búsqueda.

WITH TARGET KEY
Especifica los parámetros de clave destino que son la salida de la función generadora de índices especificada en la cláusula GENERATE KEY USING.

nombre3-parámetro
Identifica el parámetro asociado a una clave destino determinada. *nombre-parámetro3* corresponde a las columnas de la tabla RETURNS tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING. El número de parámetros especificados debe coincidir con el número de columnas devueltas por esa función de tabla (SQLSTATE 428E2).

tipo-datos3
Especifica el tipo de datos para cada *nombre-parámetro3* correspondiente. *tipo3-datos* debe coincidir exactamente con el tipo de datos de las columnas de salida correspondientes de la tabla RETURNS, tal como está especificado en la función de tabla de la cláusula GENERATE KEY USING (SQLSTATE 428E2), incluida la longitud, precisión y tipo.

SEARCH METHODS
Presenta los métodos de búsqueda que están definidos para el índice.

definición-método-búsqueda
Especifica las características del método de la búsqueda por índice. Consta de un nombre de método, los argumentos de búsqueda, una función productora de rangos y una función opcional de filtrado de índice.

WHEN *nombre-método*
Es el nombre de un método de búsqueda. Es un identificador SQL que está asociado con el nombre de método especificado en la norma de explotación de índice (situada en la cláusula PREDICATES de una función definida por el usuario). El *nombre-método-búsqueda* puede estar referenciado por una sola cláusula WHEN en la definición del método de búsqueda (SQLSTATE 42713).

nombre4-parámetro
Identifica el parámetro de un argumento de búsqueda. Estos nombres se utilizan en las cláusulas RANGE THROUGH y FILTER USING.

tipo4-datos
Es el tipo de datos asociado a un parámetro de búsqueda.

RANGE THROUGH *invocación-función-productora-rangos*
Especifica una función de tabla externa que produce rangos de búsqueda. Esta función utiliza *nombre-parámetro1*, *nombre-parámetro4* o una constante como argumentos y devuelve un conjunto de rangos de búsqueda.

La función de tabla que se utiliza en *invocación-función-producción-rango* debe cumplir estas condiciones:

- Su resolución debe producir una función de tabla (SQLSTATE 428E4)
- No debe incluir una subconsulta (SQLSTATE 428E3) ni una función SQL (SQLSTATE 428E4) en sus argumentos
- No debe incluir una expresión XMLQUERY o XMLEXISTS en sus argumentos (SQLSTATE 428E3)

- No debe estar definida con PARAMETER CCSID UNICODE si esta base de datos no es Unicode (SQLSTATE 428E4)
- No debe estar definida con LANGUAGE SQL (SQLSTATE 428E4)
- No debe estar definida como NOT DETERMINISTIC ni como EXTERNAL ACTION (SQLSTATE 428E4)
- Debe haberse definido con NO SQL (SQLSTATE 428E4).

El número y los tipos de los resultados de esta función deben estar relacionados con los resultados de la función de tabla especificada en la cláusula GENERATE KEY USING (SQLSTATE 428E1) de la manera siguiente:

- Devolviendo un número máximo de columnas igual al doble de las devueltas por la función de transformación de claves
- Teniendo un número par de columnas, donde la primera mitad de las columnas devueltas definen el inicio del rango (valores de clave de inicio) y la segunda mitad de las columnas devueltas definen el final del rango (valores de clave de parada)
- Teniendo cada columna de clave de inicio el mismo tipo que la correspondiente columna de clave de parada
- Teniendo cada columna de clave de inicio el mismo tipo que la correspondiente columna de la función de transformación de claves

Más concretamente, 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 las columnas m <= n y "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)
- No debe incluir una expresión XMLQUERY o XMLEXISTS (SQLSTATE 428E3)

Si un argumento invoca otra función u otro método, estas normas también se aplican para la función o el método anidados. Sin embargo, los métodos observadores generados por el sistema pueden utilizarse como argumentos de la función de filtro (o de cualquier función o método utilizado como argumento), siempre que el resultado sea un argumento de tipo de datos incorporado.

El usuario que ha definido la extensión de índice debe disponer de privilegio EXECUTE para esta función.

expresión-case
Especifica una expresión CASE para el filtrado de entradas de índice. Se puede utilizar *nombre-parámetro1*, *nombre-parámetro3*, *nombre-parámetro4* o una constante (SQLSTATE 42703) en la *cláusula-when-búsqueda* y en la *cláusula-when-simple*. Como *expresión-resultante* se puede utilizar una función externa con las normas especificadas en FILTER USING *invocación-función-filtrado-índice*. Cualquier función referenciada en la *expresión-case* debe también seguir las normas descritas para *invocación-función-filtrado-índice*. Además, no se pueden utilizar subconsultas ni expresiones XMLQUERY o XMLEXISTS en ningún lugar de la *expresión-case* (SQLSTATE 428E4). La expresión case debe devolver un valor entero (SQLSTATE 428E4). Si la *expresión-resultante* devuelve el valor 1, significa que se conserva la entrada de índice; de lo contrario, ésta se descarta.

Notas

- La creación de una extensión de índice con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema, siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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
```

CREATE METHOD

La sentencia CREATE METHOD se utiliza para asociar un cuerpo de método a una especificación de método que ya forma parte de la definición de un tipo estructurado definido por el usuario.

Invocación

Esta sentencia 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

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

- Privilegio CREATEIN para el esquema del tipo estructurado al que se hace referencia en la sentencia CREATE METHOD
- Propietario del tipo estructurado al que se hace referencia en la sentencia CREATE METHOD
- Autorización SYSADM o DBADM

Para asociar un cuerpo de método externo a su especificación de método, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

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

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:

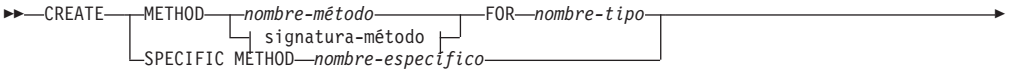
- Privilegio CONTROL para esa tabla, vista o apodo o
- Privilegio SELECT para esa tabla, vista o apodo

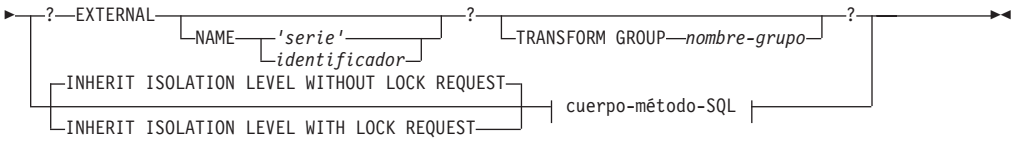
Si el usuario que define un método de SQL sólo puede crear el método porque dicho usuario dispone de autorización SYSADM, se le otorgará autorización DBADM implícita para que pueda crear el método.

Los privilegios de grupo distintos de PUBLIC no se consideran para ninguna tabla o vista que se haya especificado en la sentencia CREATE METHOD.

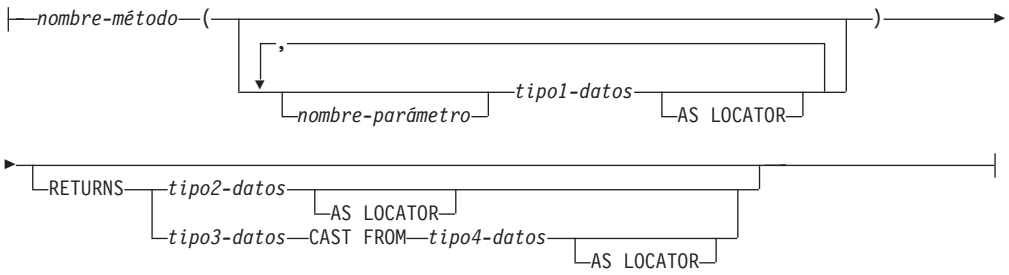
Los requisitos de autorización de la fuente de datos para la tabla o vista a la que hace referencia el apodo se aplican al invocarse el método. El ID de autorización de la conexión se puede correlacionar con un ID de autorización remoto diferente.

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.

tipo1-datos

Especifica el tipo de datos de cada parámetro. No se da soporte a los tipos de matriz (SQLSTATE 42815).

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.

tipo2-datos
Especifica el tipo de datos de la salida. No se da soporte a los tipos de matriz (SQLSTATE 42815).

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que el método debe devolver un localizador de LOB, en lugar del valor real.

tipo3-datos **CAST FROM** *tipo4-datos*
Este formato de cláusula RETURNS se utiliza para devolver un tipo de datos diferente a la sentencia de invocación del tipo de datos que se ha devuelto por el código de función.

AS LOCATOR
Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede utilizar la cláusula AS LOCATOR para indicar que el método debe devolver un localizador de LOB, en lugar del valor real.

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

SPECIFIC METHOD *nombre-específico*
Identifica el método concreto, utilizando el nombre específico que se especificó o se tomó por omisión al ejecutar CREATE TYPE. El nombre-específico debe identificar una especificación de método del esquema especificado o implícito; de lo contrario se produce un error (SQLSTATE 42704).

EXTERNAL
Esta cláusula indica que se utiliza la sentencia CREATE METHOD para registrar un método, basándose en el código escrito en un lenguaje de programación externo y de acuerdo con los convenios documentados para enlaces e interfaces. La especificación-método asociada de CREATE TYPE debe especificar un valor distinto de SQL para LANGUAGE. Cuando se invoca el método, el sujeto del método se pasa a la implementación como primer parámetro implícito.

Si no se especifica la cláusula NAME, se supone "NAME *nombre-método*".

NAME
Esta cláusula identifica el nombre del código escrito por el usuario que aplica el método que se está definiendo.

'*serie*'
La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada. Para obtener más información sobre los convenios específicos de lenguaje, consulte la "sentencia 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 del esquema. Esta modalidad de NAME sólo puede utilizarse con LANGUAGE C (tal como está definido en la especificación-método en CREATE TYPE).

TRANSFORM GROUP *nombre-grupo*
Indica el grupo de transformación que se utiliza, cuando se invoca el método, para las transformaciones de tipo estructurado definidas por el usuario. Es necesaria una transformación, pues la definición de método incluye un tipo estructurado definido por el usuario.

Se recomienda especialmente especificar un nombre de grupo de transformación; si no se especifica esta cláusula, el nombre de grupo por omisión que se utiliza es DB2_FUNCTION. Si el nombre-grupo especificado (o tomado por omisión) no está definido para un tipo estructurado referenciado, se produce en error (SQLSTATE 42741). Similarmente, si una función de transformación necesaria FROM SQL o TO SQL no está definida para el nombre-grupo y tipo estructurado proporcionados, se produce un error (SQLSTATE 42744).

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST o INHERIT ISOLATION LEVEL WITH LOCK REQUEST
Especifica si una petición de bloqueo puede asociarse o no a una cláusula de aislamiento de la sentencia cuando el método hereda el nivel de aislamiento de la sentencia que invoca el método. El valor por omisión es INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST.

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST
Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, no se puede invocar en el contexto de una sentencia de SQL que incluya una cláusula de petición de bloqueo como parte de una cláusula de aislamiento especificada (SQLSTATE 42601).

INHERIT ISOLATION LEVEL WITH LOCK REQUEST
Especifica que, como el método hereda el nivel de aislamiento de la sentencia que invoca, también hereda la cláusula de petición de bloqueo especificada.

cuerpo-método-SQL
El cuerpo-método-SQL define cómo se implementa el método si la especificación de método en CREATE TYPE es LANGUAGE SQL.

El cuerpo-método-SQL debe adaptarse a las siguientes partes de la especificación de método:

- DETERMINISTIC o NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION o NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL o READS SQL DATA (SQLSTATE 42985)

En el cuerpo-método-SQL se puede hacer referencia a nombres de parámetros. El sujeto del método se pasa a la implementación de método como un primer parámetro implícito llamado SELF.

Para obtener más detalles, consulte la "sentencia de 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
- El tipo de datos XML no se puede utilizar en un método.

Notas

- Si el método permite SQL, el programa externo no debe intentar acceder a ningún objeto federado (SQLSTATE 55047).
- ***Privilegios***

  El 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
```

CREATE NICKNAME

La sentencia CREATE NICKNAME define un apodo para un objeto de fuente 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).

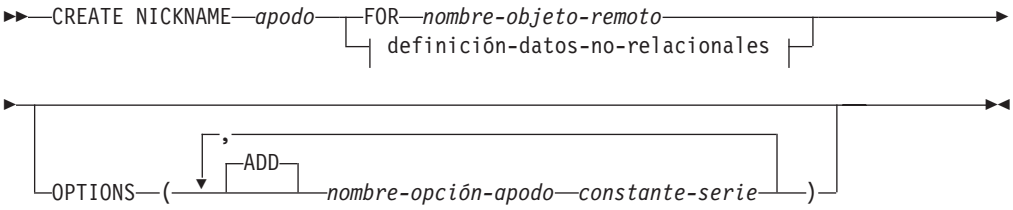
Autorización

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

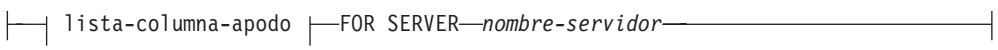
- Autorización CREATETAB para la base de datos federada, así como uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos federada, si el nombre de esquema implícito o explícito del apodo no existe
  - Privilegio CREATEIN en el esquema, si el nombre de esquema del apodo hace referencia a un esquema existente
- Autorización 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.

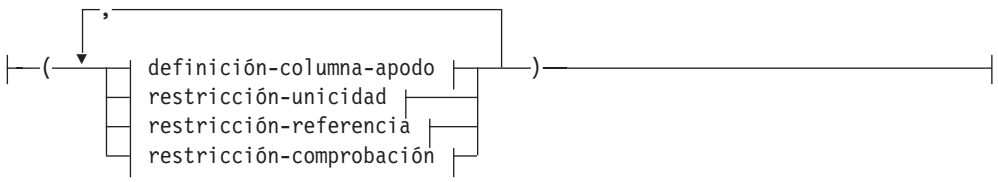
Sintaxis



definición-datos-no-relacionales:



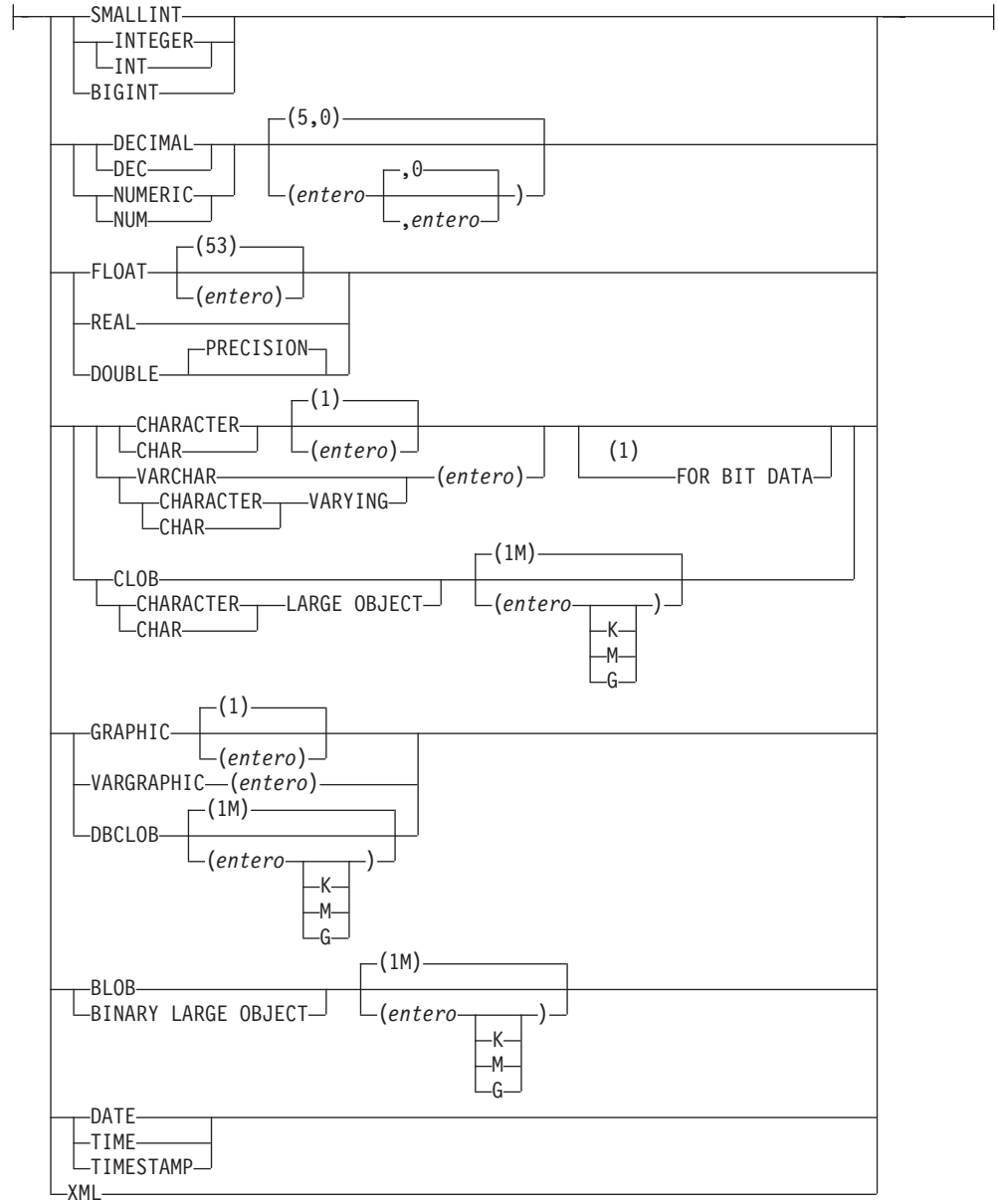
lista-columna-apodo:



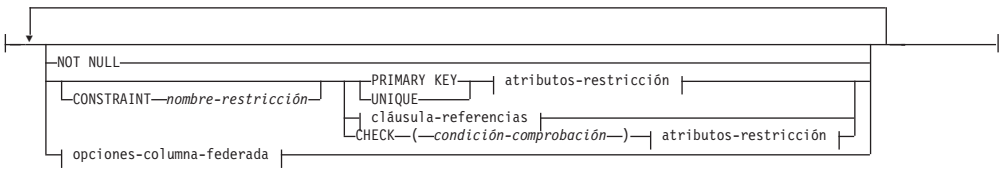
definición-columna-apodo:

tipo-datos-locales:

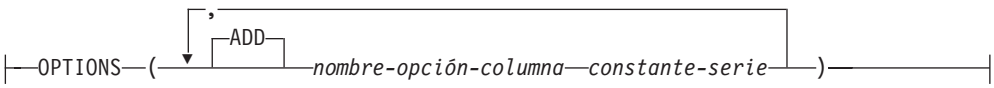
tipo-incorporado:



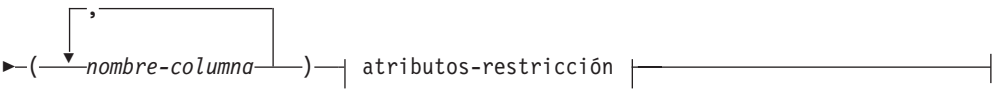
opciones-columna-apodo:



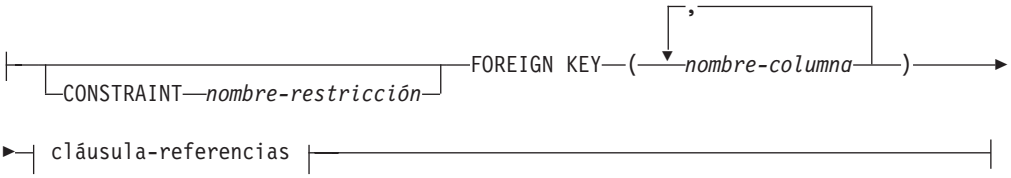
opciones-columna-federada:



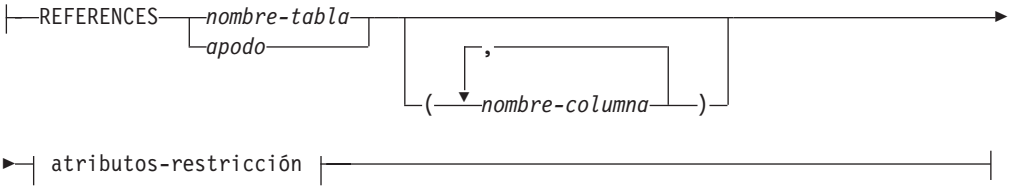
restricción-unicidad:



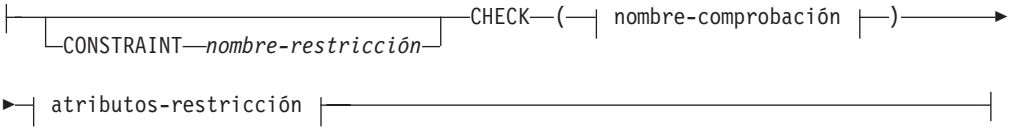
restricción-referencia:



cláusula-referencias:



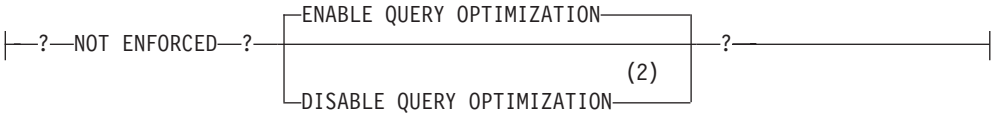
restricción-comprobación:



condición-error:

dependencia-funcional:

atributos-restricción:



Notas:

1 La cláusula FOR BIT DATA se puede especificar en cualquier orden con 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 derivador no relacional.

definición-columna-apodo
Define los atributos locales de la columna para el apodo. Algunos derivadores necesitan que se especifiquen estos atributos, mientras que otros derivadores admiten que los atributos se determinen a partir de la fuente de datos.

nombre-columna
Especifica el nombre local de la columna. El nombre puede ser diferente del de la columna correspondiente del *nombre-objeto-remoto*.

tipo-datos-locales
Especifica el tipo de datos locales para la columna. Algunos derivadores sólo dan soporte a un subconjunto de tipos de datos SQL. Para ver descripciones de tipos de datos específicos, consulte “CREATE TABLE” .

opciones-columna-apodo
Especifica opciones adicionales relacionadas con columnas del apodo.

NOT NULL
Especifica que la columna no admite valores nulos.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar una restricción que ya se haya especificado en la misma sentencia CREATE NICKNAME (SQLSTATE 42710).

Si se omite esta cláusula, el sistema genera un identificador de 18 bytes que es exclusivo entre los identificadores de las restricciones existentes que se han definido en el apodo. (El identificador se compone de la palabra 'SQL' seguida de una secuencia de 15 caracteres numéricos generados por una función basada en la indicación de fecha y hora.)

Cuando se utiliza con una restricción PRIMARY KEY o UNIQUE, se puede utilizar el *nombre-restricción* como nombre de una especificación de índice que se crea para dar soporte a la restricción.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. De este modo, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada.

Consulte PRIMARY KEY en *restricción-unicidad* más adelante.

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

Consulte UNIQUE en *restricción-unicidad* más adelante.

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

Consulte el apartado *cláusula-referencias* en *restricción-referencia*, que encontrará más adelante.

CHECK (*condición-comprobación***)**
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 derivadores 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 exclusiva compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte “Número total de bytes” en “CREATE TABLE”. Para ver información sobre los límites de longitud de clave, consulte el apartado “Límites de SQL y XQuery”. No se puede utilizar como parte de una clave exclusiva ninguna columna LOB, columna de tipo diferenciado basado en un LOB o columna de tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

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 fecha y hora (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

PRIMARY KEY (*nombre-columna***,...)**
Define una clave primaria formada por las columnas identificadas. La cláusula no debe especificarse más de una vez y las columnas identificadas deben definirse como NOT NULL. Cada *nombre-columna* debe identificar una columna del apodo y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XQuery". No se puede utilizar como parte de una clave primaria ninguna columna LOB, columna de tipo diferenciado basado en un LOB o columna de tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

Sólo se puede definir una única clave primaria en un apodo.

La descripción del apodo, tal como está registrada en el catálogo, incluye la clave primaria y su especificación de índice. Se creará automáticamente una especificación de índice para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre de la especificación de índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice o una especificación de índice existente en el esquema donde se crea el apodo. Si el nombre de la especificación de índice entra en conflicto, el nombre será la palabra 'SQL', seguida por los caracteres de la indicación de fecha y hora (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

restricción-referencia
Define una restricción de referencia.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de referencia.

FOREIGN KEY (*nombre-columna***,...)**
Define una restricción de referencia con el *nombre-restricción* especificado.

Deje que N1 indique el apodo de objeto de la sentencia. La clave foránea de la restricción de referencia se compone de las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de N1 y la misma columna no se debe identificar más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte "Número total de bytes" en "CREATE TABLE". Para ver información sobre los límites de longitud de clave, consulte el apartado "Límites de SQL y XQuery". Las claves foráneas pueden definirse en columnas de longitud variable cuya longitud sea superior a 255 bytes. No se puede utilizar como parte de una clave foránea ninguna columna LOB, columna de tipo diferenciado basado en un LOB o 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 contienen tipos de datos compatibles (ambas columnas son numéricas, de serie de caracteres, gráficas, de fecha y hora o tienen el mismo tipo diferenciado).

cláusula-referencias
Especifica la tabla padre o el apodo padre y la clave padre para la restricción de referencia.

REFERENCES *nombre-tabla* **o** *apodo*
La tabla o el apodo especificados en una cláusula REFERENCES debe identificar una tabla base o un apodo que se describa en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción de referencia está duplicada si su clave foránea, clave padre y tabla padre o apodo padre son iguales a la clave foránea, clave padre y tabla padre o apodo padre de una restricción de referencia especificada previamente. Las restricciones de referencia duplicadas se pasan por alto y se emite un aviso (SQLSTATE 01543).

En la siguiente explicación, N2 indicará la tabla padre identificada o el apodo padre y N1 indicará el apodo que se está creando (o modificando). N1 y N2 pueden ser el mismo apodo.

La clave foránea especificada debe tener el mismo número de columnas que la clave padre de N2, y la descripción de la columna número *n* de la clave foránea debe ser comparable a la descripción de la columna número *n* de esa clave padre. Las columnas de indicación de fecha y hora no se consideran compatibles con las columnas de serie al aplicar esta norma.

La restricción de referencia especificada por una cláusula FOREIGN KEY define una relación en la que N2 es el padre y N1 es dependiente.

(*nombre-columna***,...)**
La clave padre de la restricción de referencia se compone de las columnas identificadas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de N2. La misma columna no se puede identificar más de una vez.

La lista de nombres de columnas debe coincidir con el conjunto de columnas (en cualquier orden) de la clave primaria o una restricción exclusiva que exista en N2 (SQLSTATE 42890). Si no se especifica una lista de nombres de columna, N2 debe tener una clave primaria (SQLSTATE 42888). La omisión de la lista de nombres de columna es una especificación implícita de las columnas de dicha clave primaria en la secuencia especificada originalmente.

atributos-restricción
Define los atributos que se asocian a las restricciones de comprobación o de integridad referencial.

NOT ENFORCED
El gestor de bases de datos no aplica la restricción durante la realización de las operaciones normales como, por ejemplo, la inserción, la actualización o la supresión.

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

DISABLE QUERY OPTIMIZATION
No se puede utilizar la restricción para la optimización de consulta.

restricción-comprobación
Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o que define una dependencia funcional entre columnas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de comprobación.

CHECK (*condición-comprobación***)**
Define una restricción de comprobación. La *condición-comprobación* debe ser verdadera o desconocida para cada fila del apodo.

condición-búsqueda
La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia a columna debe ser a una columna del apodo que se crea.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- No puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
  - Subconsultas
  - Operaciones de eliminación de referencia o funciones DEREF donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
  - Especificaciones CAST con una cláusula SCOPE
  - Funciones de columna
  - Funciones que no sean deterministas
  - Funciones definidas para que exista una acción externa
  - Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
  - Variables del lenguaje principal
  - Marcadores de parámetro
  - Registros especiales
  - Variables globales
  - Referencias a columnas generadas que no correspondan a la columna de identidad

dependencia-funcional
Define una dependencia funcional entre columnas.

El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la columna no debe ser un tipo de datos LOB, un tipo diferenciado basado en el tipo de datos LOB ni un tipo estructurado (SQLSTATE 42962). Ninguna columna del conjunto de columnas hijo puede ser una columna 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 ejemplos de objetos de fuente de datos relacionales son: 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).
- El objeto de fuente de datos al que el apodo hace referencia ya debe existir en la fuente de datos indicada por el primer calificador en el *nombre-objeto-remoto*.
- La lista de los tipos de datos de fuente de datos soportados varía según el derivador. Los iniciadores no dan soporte a los tipos de datos de fuente de datos que corresponden con los siguientes tipos de datos DB2: XML, DECFLOAT, 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 128 bytes. Si se está creando un apodo para una tabla relacional que tiene un índice cuyo nombre sobrepasa esta longitud, no se cataloga todo el nombre. En su lugar, DB2 lo trunca a 128 bytes. Si la serie formada por estos caracteres no es exclusiva en el esquema al que pertenece el índice, DB2 intenta convertirla en exclusiva sustituyendo el último carácter por un 0. Si el resultado continúa sin ser exclusivo, DB2 cambia el último carácter por un 1. DB2 repite este proceso con los números 2 a 9 y, si es necesario, con los números 0 a 9 para el 127 carácter del nombre, para el 126 carácter y así sucesivamente hasta que se genera un número exclusivo. Como ilustración: El nombre de 130 bytes de un índice de la tabla de fuente de datos se llama AREALLY...REALLYLONGNAME. Los nombres AREALLY...REALLYLONGNA y AREALLY...REALLYLONGN0 ya existen en el esquema al que pertenece el índice. El nombre nuevo tiene más de 128 bytes; por lo tanto, DB2 lo trunca a AREALLY...REALLYLONGNA. Puesto que este nombre ya existe en el esquema, DB2 cambia la versión truncada por AREALLY...REALLYLONGN0. Y puesto que este nombre ya existe, DB2 cambia la versión truncada por AREALLY...REALLYLONGN1. Este nombre aún no existe en el esquema, por lo que DB2 lo acepta como el nuevo nombre.
- Cuando se crea un apodo para un objeto de fuente de datos, DB2 almacena los nombres de las columnas de apodo en el catálogo. Cuando el objeto de fuente de datos es una tabla o una vista, DB2 hace que los nombres de las columnas de apodo sean los mismos que los nombres de las columnas de tabla o de vista. Si un nombre excede la longitud máxima permitida para los nombres de columna de DB2, DB2 trunca el nombre en función de esa longitud. Si la versión truncada no es exclusiva entre los demás nombres de las columnas de la tabla o vista, DB2 hace que sea exclusivo siguiendo el procedimiento que se describe en el párrafo anterior.
- Si el objeto de fuente de datos tiene índice definidos, se crean especificaciones de índice para cada índice cuando se crea el apodo. No se crean especificaciones de índice en la fuente de datos para los índices que tienen:
  - Nombres de columnas duplicados
  - Más de 64 columnas
  - Más de 1.024 bytes en la suma de la longitud de las partes de la clave de índice
- Si se cambia la definición de un objeto de fuente de datos remota (por ejemplo, se suprime una columna o se cambia el tipo de datos), se debe descartar y volver a crear el apodo; de lo contrario, se pueden producir errores cuando se utilice el apodo en una sentencia de SQL.
- ***Almacenamiento en antememoria y objetos protegidos:*** Cuando se crea un apodo, si el objeto de fuente de datos no está protegido, ALLOW CACHING está en vigor para el apodo. Si el servidor federado puede detectar que el objeto de fuente de datos está protegido, DISALLOW CACHING está en vigor para el apodo. La opción DISALLOW CACHING garantiza que cada vez que se utiliza el apodo se devuelven los datos del ID de autorización apropiados desde la fuente de datos en el momento de la ejecución de la consulta. Esto se realiza limitando el uso del apodo en la definición de una tabla de consulta materializada en el servidor federado, que podría utilizarse para guardar en la antememoria los datos de apodo. La sentencia ALTER NICKNAME puede utilizarse para cambiar entre ALLOW CACHING y DISALLOW CACHING.

Ejemplos

*Ejemplo 1:*Cree un apodo para una vista, DEPARTMENT, que esté en un esquema llamado HEDGES. Esta vista se almacena en una fuente de datos DB2 para z/OS denominada OS390A.

```
CREATE NICKNAME DEPT
  FOR OS390A.HEDGES.DEPARTMENT
```

Ejemplo 2: seleccione todos los registros de la vista para la cual se ha creado un apodo en el Ejemplo 1. Se debe hacer referencia a la vista mediante su apodo. Se puede hacer referencia a la vista remota utilizando el nombre por el cual se conoce en la fuente de datos únicamente en sesiones de paso a través.

```
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"."japón"
```

Ejemplo 4: Cree un apodo para el archivo estructurado de tabla DRUGDATA1.TXT. Incluya las opciones de apodo FILE_PATH, COLUMN DELIMITER, KEY_COLUMN y VALIDATE_DATA_FILE en la sentencia.

```
CREATE NICKNAME DRUGDATA1
  (Dcode         INTEGER,
  DRUG           CHAR(20),
  MANUFACTURER   CHAR(20))
  FOR SERVER biochem_lab
  OPTIONS
    (FILE_PATH '/usr/pat/DRUGDATA1.TXT',
     COLUMN_DELIMITER ',',
     KEY_COLUMN 'DCODE',
     SORTED 'Y',
     VALIDATE_DATA_FILE 'Y')
```

Ejemplo 5: Cree el apodo padre CUSTOMERS en múltiples archivos XML bajo la vía de acceso de directorio especificada /home/db2user. Incluya las opciones siguientes:

- Opciones de columna:
  - La opción de columna XPATH para la columna VARCHAR(5) denominada ID, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna XPATH para la columna VARCHAR(16) denominada NAME, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna XPATH para la columna VARCHAR(30) denominada ADDRESS, que indica el elemento o el atributo de los archivos XML del que se extraen los datos de la columna
  - La opción de columna PRIMARY_KEY para la columna VARCHAR(16) denominada CID, que identifica el apodo de los clientes como apodo padre en una jerarquía de apodos
- Opciones de apodo:

- La opción de apodo DIRECTORY_PATH para indicar la ubicación de los archivos XML que proporcionan los datos
- La opción de apodo XPATH para indicar el elemento de los archivos XML donde empiezan los datos
- La opción de apodo STREAMING para indicar que los datos fuente XML se separan y procesan elemento por elemento. En este ejemplo, el elemento es un registro de cliente.

```
CREATE NICKNAME clientes
  (id      VARCHAR(5)   OPTIONS(XPATH './@id'),
   name    VARCHAR(16)  OPTIONS(XPATH './/name'),
   address VARCHAR(30)  OPTIONS(XPATH './/address/@street'),
   cid     VARCHAR(16)  OPTIONS(PRIMARY_KEY 'YES'))
   FOR SERVER xml_server
   OPTIONS
     (DIRECTORY_PATH '/home/db2user',
      XPATH '//customer',
      STREAMING 'YES')
```

CREATE PROCEDURE

La sentencia CREATE PROCEDURE define un procedimiento en el servidor actual.

Mediante esta sentencia, pueden crearse tres tipos distintos de procedimientos. Cada uno de los tipos se describe por separado.

- Externo. El cuerpo del procedimiento se escribe en un lenguaje de programación. Al ejecutable externo hace referencia un procedimiento que se ha definido en el servidor actual, junto con diversos atributos del procedimiento.
- Con fuente. El cuerpo del procedimiento forma parte del procedimiento con fuente, al que hace referencia el procedimiento con fuente definido en el servidor actual, junto con distintos atributos del procedimiento. Un procedimiento con fuente cuyo procedimiento fuente se encuentra en una fuente de datos también se denomina *procedimiento federado*.
- SQL. El cuerpo del procedimiento se escribe en SQL y se define en el servidor actual, junto con diversos atributos del procedimiento.

CREATE PROCEDURE (externo)

La sentencia CREATE PROCEDURE (externo) define un procedimiento externo en el servidor actual.

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

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

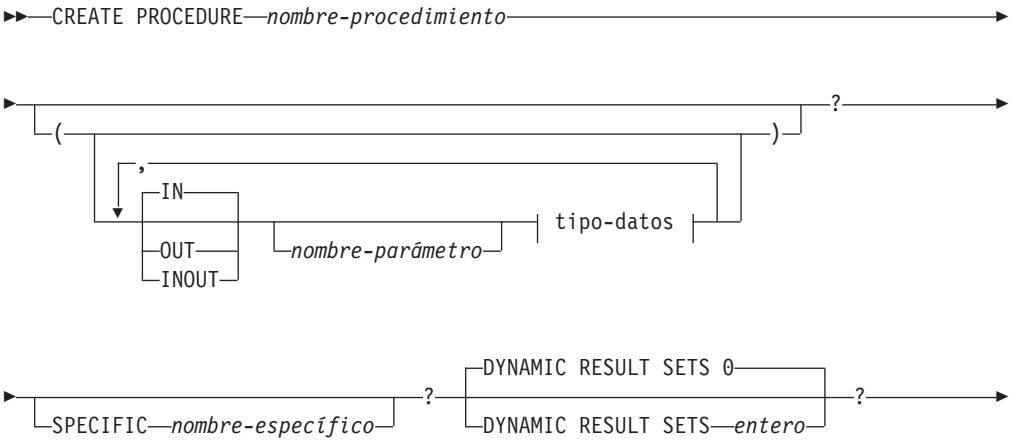
- Autorización CREATE_EXTERNAL_ROUTINE en la base de datos y, como mínimo, uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del procedimiento no hace referencia a un esquema existente
  - Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente
- Autorización SYSADM o DBADM

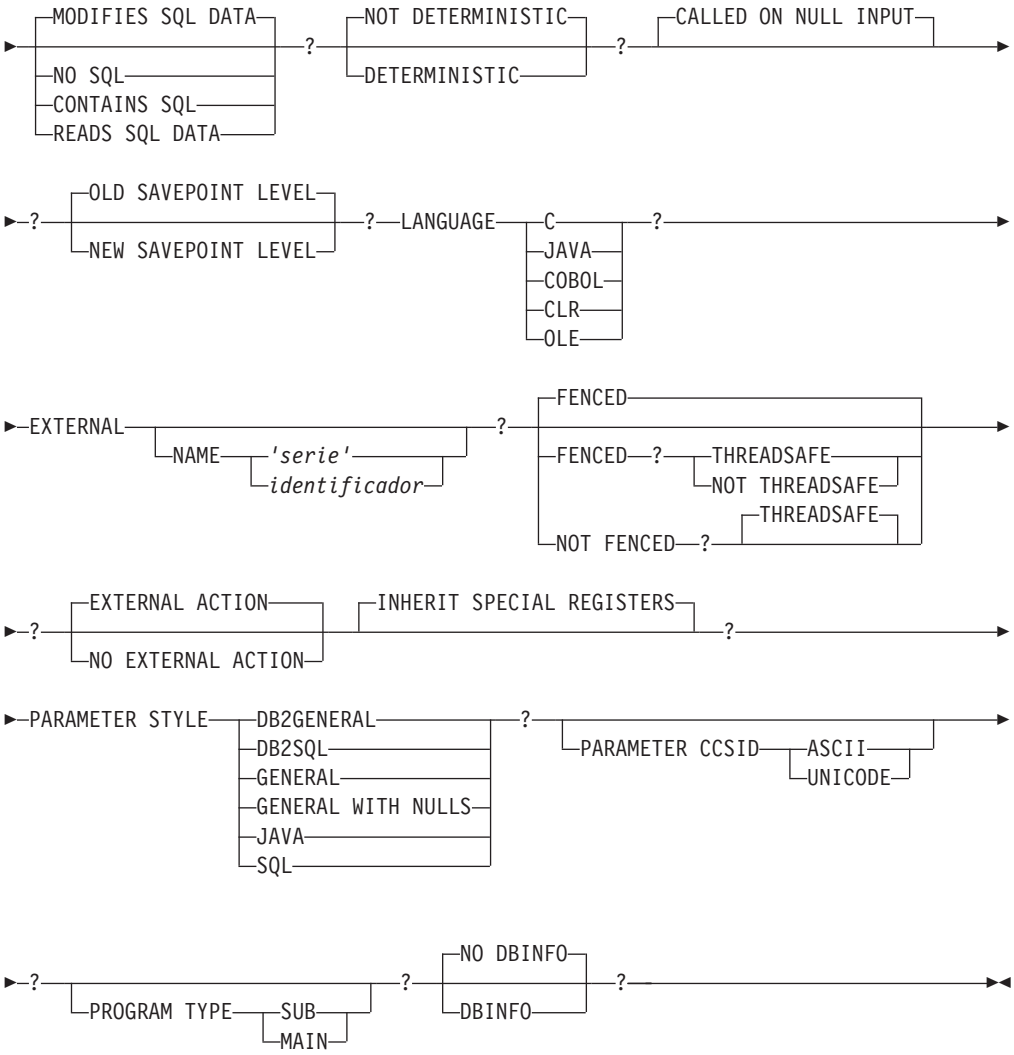
Para crear un procedimiento no delimitado, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes:

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

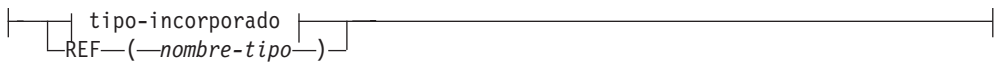
Para crear un procedimiento delimitado, no necesita autorizaciones ni privilegios adicionales.

Sintaxis

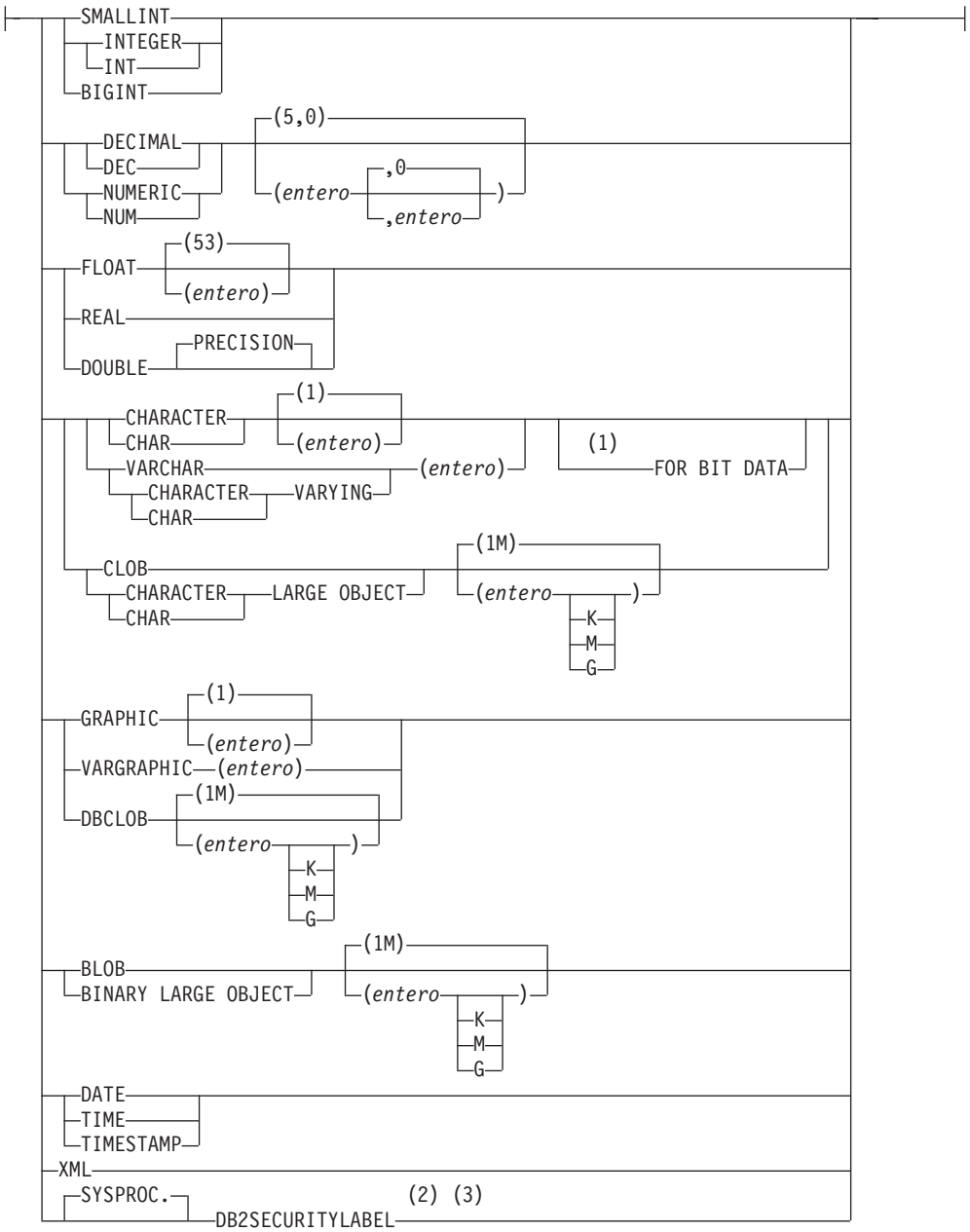




tipo-datos:



tipo-incorporado:



Notas:

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

2 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

3 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

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 sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador 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. Una signatura duplicada devuelve un error de SQL (SQLSTATE 42723).

Por ejemplo, si se emiten estas sentencias:

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

la segunda sentencia no se ejecutará satisfactoriamente porque el número de parámetros del procedimiento es el mismo, aunque los tipos de datos no lo sean.

Si el procedimiento devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT
Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT
Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro
Opcionalmente, especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos
Especifica el tipo de datos del parámetro.

- 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 LONG VARCHAR ni a LONG VARGRAPHIC como tipos de parámetros para procedimientos externos.
- XML no es válido con LANGUAGE OLE.
- Dado que el valor XML que se ve dentro de un procedimiento es una versión serializada del valor XML que se pasa como parámetro en la llamada de procedimiento, los parámetros del tipo XML deben declararse mediante la sintaxis `XML AS CLOB(n)`.
- CLR no da soporte a una escala DECIMAL mayor que 28 (SQLSTATE 42613).
- La coma flotante decimal no se soporta con los lenguajes C, Java COBOL, CLR y OLE (SQLSTATE 42613).
- Sólo se da soporte a los parámetros del tipo de matriz en procedimientos Java (SQLSTATE 42815).

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 128). 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 será 'SQL' seguido de una indicación de fecha y hora expresada en forma de caracteres: 'SQL*aammddhhmmssxxx*'.

DYNAMIC RESULT SETS *entero*

Indica el límite superior estimado de los conjuntos de resultados devueltos para el procedimiento.

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

Indica si el procedimiento emite sentencias de SQL y, en caso afirmativo, de qué tipo.

NO SQL

Indica que el procedimiento no puede ejecutar ninguna sentencia de SQL (SQLSTATE 38001).

CONTAINS SQL

Indica que el procedimiento puede ejecutar sentencias de SQL que no lean ni modifiquen datos de SQL (SQLSTATE 38004). Las sentencias que no están soportadas en ningún procedimiento devuelven un error diferente (SQLSTATE 38003).

READS SQL DATA
Indica que en el procedimiento pueden incluirse algunas sentencias de SQL que no modifiquen datos de SQL (SQLSTATE 38002 ó 42985). Las sentencias que no están soportadas en ningún procedimiento devuelven un error diferente (SQLSTATE 38003).

MODIFIES SQL DATA
Indica que el procedimiento puede ejecutar cualquier sentencia de SQL excepto las que no están soportadas en ningún procedimiento (SQLSTATE 38003).

DETERMINISTIC o NOT DETERMINISTIC
Esta cláusula especifica si el procedimiento devuelve siempre los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si el procedimiento depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un procedimiento DETERMINISTIC debe siempre devolver el mismo resultado ante invocaciones sucesivas con entradas de datos idénticas.

Actualmente, esta cláusula no afecta al proceso del procedimiento.

CALLED ON NULL INPUT
CALLED ON NULL INPUT siempre se aplica a los procedimientos. Esto significa que el procedimiento recibe llamada independientemente de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento comprobar si hay valores de argumento nulos.

OLD SAVEPOINT LEVEL o NEW SAVEPOINT LEVEL
Especifica si este procedimiento establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información acerca de los niveles de punto de salvaguarda, consulte la sección “Normas” de la descripción de la sentencia SAVEPOINT.

LANGUAGE
Esta cláusula obligatoria se emplea para especificar el convenio de la interfaz de lenguaje en el que está escrito el cuerpo del procedimiento.

C Esto significa que el gestor de bases de datos llamará al procedimiento como si fuera un procedimiento C. El procedimiento debe ajustarse al convenio de llamada y enlace del lenguaje C, definido por el prototipo C de ANSI estándar.

JAVA
Esto significa que el gestor de bases de datos llamará al procedimiento como un método de una clase Java.

COBOL
Esto significa que el gestor de bases de datos llamará al procedimiento como si fuera un procedimiento COBOL.

CLR
Esto significa que el gestor de bases de datos llamará al procedimiento como método de una clase .NET. En este momento, sólo se soporta LANGUAGE CLR para los procedimientos que se ejecutan en los sistemas operativos Windows. No se puede especificar NOT FENCED para una rutina CLR (SQLSTATE 42601).

OLE
Esto significa que el gestor de bases de datos llamará al procedimiento

como si fuera un método expuesto por un objeto de automatización OLE. El procedimiento almacenado debe ajustarse a los tipos de datos de automatización y al mecanismo de invocación de OLE. Además, el objeto de automatización OLE necesita implementarse como servidor en proceso (DLL). Estas restricciones se describen en la publicación *OLE Automation Programmer's Reference*.

LANGUAGE OLE sólo recibe soporte para los procedimientos almacenados en DB2 para los sistemas operativos Windows. THREADSAFE no puede especificarse para los procedimientos que se han definido con LANGUAGE OLE (SQLSTATE 42613).

EXTERNAL
Esta cláusula indica que la sentencia CREATE PROCEDURE se emplea para registrar un nuevo procedimiento basado en el código escrito en un lenguaje de programación externo y cumple los convenios estipulados para los enlaces e interfaces.

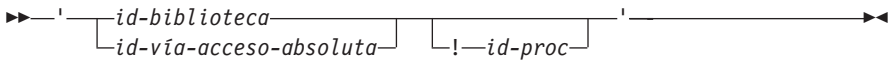
Si no se especifica la cláusula NAME, se supone "NAME *nombre-procedimiento*". Si la cláusula NAME no tiene un formato correcto, se devuelve un error (SQLSTATE 42878).

NAME *'serie'*
Esta cláusula identifica el nombre del código escrito por el usuario que implanta el procedimiento que se está definiendo.

La opción 'serie' es una constante de tipo serie con un máximo de 254 bytes. El formato que se utiliza para la serie depende de la opción LANGUAGE especificada.

- Para LANGUAGE C:

  La *serie* especificada constituye el nombre de la biblioteca y el procedimiento de la biblioteca que el gestor de bases de datos invoca para ejecutar el procedimiento que ha de crearse (CREATE). Al ejecutar la sentencia CREATE PROCEDURE, no es preciso que exista la biblioteca (ni el procedimiento dentro de la biblioteca). Sin embargo, cuando se llama el procedimiento, deben existir la biblioteca y el procedimiento en la biblioteca y deben ser accesibles desde la máquina servidora de la base de datos.



El nombre debe especificarse entre comillas simples. No está permitido especificar espacios en blanco adicionales.

id-biblioteca
Identifica el nombre de la biblioteca donde reside el procedimiento. El gestor de bases de datos buscará en la biblioteca de la manera siguiente:

– En los sistemas UNIX, si se ha especificado 'myfunc' como *library_id*, y el gestor de bases de datos se ejecuta desde /u/production, el gestor de bases de datos buscará el procedimiento en la biblioteca /u/production/sqllib/function/myproc si se ha especificado FENCED o en /u/production/sqllib/function/unfenced/myproc si se ha especificado NOT FENCED.

– En los sistemas operativos Windows, el gestor de bases de datos buscará la función en la vía de acceso del directorio que las variables de entorno LIBPATH o PATH especifican.

Los procedimientos almacenados en cualquiera de estos directorios no utilizan ninguno de los atributos registrados.

id-vía-acceso-absoluta
Identifica el nombre completo de vía de acceso del procedimiento.

En sistemas UNIX, por ejemplo, /u/jchui/mylib.systems haría que el gestor de bases de datos buscara el procedimiento myproc en /u/jchui/mylib.

En los sistemas operativos Windows, 'd:\mylib\myproc.dll' haría que el gestor de bases de datos cargara el archivo myproc.dll desde el directorio d:\mylib. Si se utiliza un ID de vía de acceso absoluta para identificar el cuerpo de la rutina, asegúrese de añadir la extensión .dll.

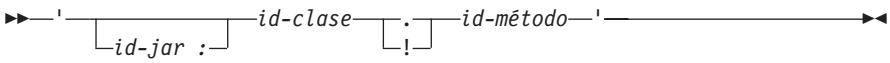
! id-proc
Identifica el nombre del punto de entrada del procedimiento que se debe invocar. El carácter de exclamación (!) sirve como delimitador entre el ID de biblioteca y el ID de procedimiento. '!proc8' indicaría al gestor de bases de datos que buscara la biblioteca en la ubicación especificada por el *id-vía-acceso-absoluta* y que utilizara el punto de entrada proc8 dentro de esa biblioteca.

Si la serie no se ha formado correctamente, se devuelve un error (SQLSTATE 42878).

El cuerpo de cada procedimiento debe encontrarse en un directorio que se haya montado y que esté disponible en todas las particiones de base de datos.

- Para LANGUAGE JAVA:

  La *serie* especificada contiene el identificador del archivo jar opcional, el identificador de clase y el identificador de método, que el gestor de bases de datos invoca para ejecutar el procedimiento que ha de crearse (CREATE). No es preciso que existan el identificador de clase y el identificador de método cuando se ejecuta la sentencia CREATE PROCEDURE. Si se especifica un *id_jar*, 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'.

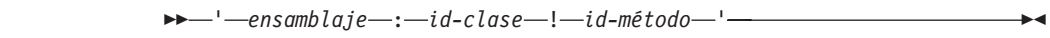
id-clase
Identifica el identificador de clase del objeto Java. Si la clase forma parte de un paquete, la parte del identificador de clase debe incluir el prefijo de paquete completo, por ejemplo, 'misPaqs.StoredProcs'. La máquina virtual Java buscará en el directorio '../myPacks/StoredProcs/' las clases. En los sistemas operativos Windows, la máquina virtual Java buscará en el directorio '..\myPacks\StoredProcs\'.

id-método
Identifica el nombre de método de la clase Java que se invoca.

- Para LANGUAGE CLR:

  La *serie* especificada representa el ensamblaje .NET (biblioteca o ejecutable), la clase de ese ensamblaje y el método dentro de la clase que el gestor de bases de datos invoca para ejecutar el procedimiento que se está creando. No es necesario que existan el módulo, la clase y el método cuando se ejecute la sentencia CREATE PROCEDURE. Sin embargo, cuando se llama al procedimiento, el módulo, la clase y el método deben existir y poderse acceder desde la máquina de servidor de bases de datos, de lo contrario se devuelve un error (SQLSTATE 42284).

  Las rutinas C++ que se compilan con la opción de compilador '/clr' para indicar que incluyen extensiones de código gestionado deben estar catalogadas como 'LANGUAGE CLR' y no como 'LANGUAGE C'. DB2 debe saber que se utiliza la infraestructura .NET en un procedimiento, para tomar las decisiones necesarias en tiempo de ejecución. Todos los procedimientos que utilicen la infraestructura .NET deben estar catalogados como 'LANGUAGE CLR'.



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 de función de la vía de acceso de instancia de DB2 (por ejemplo, c:\DB2\function). Si el archivo reside en un subdirectorio del directorio de función de la instancia, se puede especificar el subdirectorio antes del nombre de archivo en lugar de especificar toda la vía de acceso. Por ejemplo, si el directorio de instancia es c:\DB2 y el archivo de ensamblaje es c:\DB2\function\myprocs\mydotnet.dll, sólo es necesario especificar 'myprocs\mydotnet.dll' para el ensamblaje. La sensibilidad a las mayúsculas y minúsculas de este parámetro es igual a la del sistema de archivos.

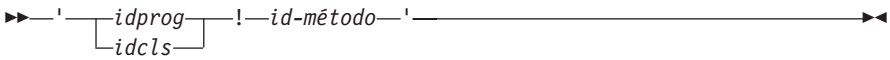
id-clase
Especifica el nombre de la clase del ensamblaje proporcionado en el que reside el método que se debe invocar. Si la clase reside en un espacio de nombres, se debe facilitar el espacio de nombres completo además de la clase. Por ejemplo, si la clase EmployeeClass está en el espacio de nombres MyCompany.ProcedureClasses, se debe especificar MyCompany.ProcedureClasses.EmployeeClass para la

clase. Tenga en cuenta que los compiladores para algunos lenguajes .NET añadirán el nombre del proyecto como espacio de nombres para la clase y el comportamiento puede diferir según se utilice el compilador de la línea de mandatos o el compilador de la GUI. Este parámetro es sensible a las mayúsculas y minúsculas.

id-método
Especifica el método de la clase que se debe invocar. Este parámetro es sensible a las mayúsculas y minúsculas.

- Para LANGUAGE OLE:

  La serie especificada es el identificador de programa OLE (*idprog*) o el identificador de clase (*idcls*) y el identificador de método (*id_método*), que el gestor de bases de datos invoca para ejecutar el procedimiento que la sentencia crea. No es necesario que el identificador de programa o de clase ni el identificador de método existan cuando se ejecuta la sentencia CREATE PROCEDURE. Sin embargo, cuando el procedimiento se utiliza en una sentencia CALL, el identificador de método debe existir y ser accesible desde la máquina del servidor de bases de datos; de lo contrario se produce un error (SQLSTATE 42724).



  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 del 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 el procedimiento se considera "seguro" (NOT FENCED) o no (FENCED) para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos.

Si un procedimiento se registra como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios de datos) contra el acceso del procedimiento. Todos los procedimientos tienen la opción de ejecutarse como FENCED o como NOT FENCED. En general, un procedimiento que se ejecute como FENCED no funcionará tan bien como otro de iguales características que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para los procedimientos que no se han comprobado de forma adecuada puede comprometer la integridad de DB2. DB2 toma algunas precauciones frente a muchos de los tipos de errores involuntarios más habituales que pueden producirse, pero no garantiza la integridad completa cuando se utilizan procedimientos NOT FENCED.

Necesita la autorización SYSADM, la autorización DBADM o una autorización especial (CREATE_NOT_FENCED) para registrar un procedimiento como NOT FENCED. Sólo puede especificarse FENCED para un procedimiento con LANGUAGE OLE o NOT THREADSAFE.

No podrán crearse procedimientos LANGUAGE CLR cuando se especifique la cláusula NOT FENCED (SQLSTATE 42601).

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

Si 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 el LANGUAGE es JAVA o CLR. Para todos los demás lenguajes, NOT THREADSAFE es el valor por omisión. Si el procedimiento se ha definido con LANGUAGE OLE, THREADSAFE no puede especificarse (SQLSTATE 42613).

Para los procedimientos NOT FENCED, THREADSAFE es el valor por omisión. No puede especificarse NOT THREADSAFE (SQLSTATE 42613).

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si el procedimiento realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). El valor por omisión es EXTERNAL ACTION. Si

se especifica NO EXTERNAL ACTION, el sistema puede utilizar determinadas optimizaciones que suponen que el procedimiento no tiene ningún impacto externo.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación.

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

PARAMETER STYLE
Esta cláusula sirve para especificar los convenios utilizados para pasar parámetros a los procedimientos y devolver el valor de los mismos.

DB2GENERAL
Significa que el procedimiento utilizará un convenio de pase de parámetros definido para su uso con métodos Java. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

DB2SQL
Además de los parámetros existentes en la sentencia CALL, se pasan los argumentos siguientes al procedimiento:

- Un vector que contiene un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento
- El nombre específico del procedimiento
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR o OLE.

GENERAL
Significa que el procedimiento utilizará un mecanismo de pase de parámetros por medio del cual el procedimiento recibirá los parámetros especificados en la sentencia CALL. Los parámetros se envían directamente, tal como espera el lenguaje; la estructura de la SQLDA no se utiliza. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

Los indicadores nulos *no* se envían directamente al programa.

GENERAL WITH NULLS
Además de los parámetros de la sentencia CALL que ha especificado bajo GENERAL, se pasa otro argumento al procedimiento. Este argumento adicional es un vector de indicadores nulos, uno para cada parámetro de la sentencia CALL. En C, sería una matriz de enteros cortos. Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL o CLR.

JAVA
Significa que el procedimiento utilizará un convenio para el envío de parámetros que se ajusta a la especificación para el lenguaje Java y las Rutinas SQLJ. Los parámetros IN/OUT y OUT se pasarán como matrices

de entrada para facilitar los valores de retorno. Esto sólo puede especificarse cuando se utiliza LANGUAGE JAVA.

Los procedimientos PARAMETER STYLE JAVA no soportan las cláusulas DBINFO o PROGRAM TYPE.

SQL
Además de los parámetros existentes en la sentencia CALL, se pasan los argumentos siguientes al procedimiento:

- Un indicador nulo para cada parámetro de la sentencia CALL
- El SQLSTATE que debe devolverse a DB2
- El nombre calificado del procedimiento
- El nombre específico del procedimiento
- La serie de caracteres de diagnóstico de SQL que debe devolverse a DB2

Esto sólo se puede especificar cuando se utiliza LANGUAGE C, COBOL, CLR o OLE.

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031). Cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es la página de códigos de la base de datos.

UNICODE
Especifica que los datos de serie están codificados en Unicode. Si la base de datos es Unicode, los datos de caracteres están en UTF-8 y los datos gráficos están en UCS-2. Si la base de datos no es Unicode, los datos de caracteres están en UTF-8. En los dos casos, cuando se invoca el procedimiento, la página de códigos de la aplicación para el procedimiento es 1208.

Si la base de datos no es una base de datos Unicode y se crea un procedimiento con PARAMETER CCSID UNICODE, el procedimiento no puede tener ningún tipo gráfico, tipo XML ni tipos definidos por el usuario (SQLSTATE 560C1). Los procedimientos PARAMETER CCSID UNICODE sólo se pueden llamar desde un cliente DB2 Versión 8.1 o posterior (SQLSTATE 42997).

Si la base de datos no es Unicode y se ha especificado el orden de clasificación alternativo en la configuración de la base de datos, se pueden crear procedimientos con PARAMETER CCSID ASCII o PARAMETER CCSID UNICODE. Todos los datos que se pasan al procedimiento o desde él se convertirán en la página de códigos adecuada.

Esta cláusula no se puede especificar con LANGUAGE OLE, LANGUAGE JAVA ni LANGUAGE CLR (SQLSTATE 42613).

PROGRAM TYPE
Especifica si el procedimiento espera parámetros del estilo de una rutina principal o una subrutina. El valor por omisión es SUB.

SUB
El procedimiento espera que los parámetros se pasen como argumentos por separado.

MAIN
El procedimiento espera que los parámetros se pasen como un contador de argumentos y como un vector de argumentos (argc, argv). El nombre del procedimiento que debe invocarse también debe ser "main". Los procedimientos almacenados de este tipo deben seguir incorporándose de la misma forma que para una biblioteca compartida, en lugar de como se haría para un ejecutable autónomo. PROGRAM TYPE MAIN sólo es válido cuando la cláusula LANGUAGE especifica C, COBOL o CLR.

DBINFO o NO DBINFO
Indica si se pasa información específica conocida por DB2 al procedimiento cuando se invoca como argumento en tiempo de invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613). Tampoco recibe soporte para PARAMETER STYLE JAVA o DB2GENERAL.

Si se especifica DBINFO, se pasa al procedimiento una estructura que contiene la información siguiente:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de la aplicación - ID de autorización del usuario que se ha conectado a la base de datos (el registro especial SYSTEM_USER).
- Página de códigos - identifica la página de códigos de la base de datos.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos que invoca el procedimiento.
- Plataforma - contiene el tipo de plataforma del servidor.

La estructura DBINFO es común para todas las rutinas externas y contiene campos adicionales que no están relacionados con los procedimientos.

Si modifica su ID de autorización de sesión (el registro especial SESSION_USER) mediante la sentencia SET SESSION AUTHORIZATION, el ID de autorización de la aplicación continúa devolviendo el valor del registro especial SYSTEM_USER.

Notas

- La creación de un procedimiento con un nombre de esquema que todavía no existe dará como resultado la creación implícita de este esquema, siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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.
- Los parámetros de XML sólo están soportados en los procedimientos externos de LANGUAGE JAVA si se especifica la cláusula PARAMETER STYLE DB2GENERAL.

- *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 del privilegio EXECUTE para cualquiera de los paquetes que el procedimiento utiliza.
- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 para 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.

Ejemplos

Ejemplo 1: Cree la definición de procedimiento para un procedimiento, grabado en Java, al que se pasa un número de pieza y que devuelve el coste de la pieza y la cantidad disponible actualmente.

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
    OUT COST DECIMAL(7,2),
    OUT QUANTITY INTEGER)
EXTERNAL NAME 'parts.onhand'
LANGUAGE JAVA PARAMETER STYLE JAVA
```

Ejemplo 2: Cree la definición de procedimiento para un procedimiento, escrito en C, al que se pasa un número de ensamblaje y que devuelve el número de piezas que componen el ensamblaje, el coste total de las piezas y un conjunto de resultados en el que se listan los números de pieza, la cantidad y el coste unitario de cada pieza.

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,
    OUT NUM_PARTS INTEGER,
    OUT COST DOUBLE)
  EXTERNAL NAME 'parts!assembly'
  DYNAMIC RESULT SETS 1 NOT FENCED
  LANGUAGE C PARAMETER STYLE GENERAL
```

CREATE PROCEDURE (con fuente)

La sentencia CREATE PROCEDURE (con fuente) registra un procedimiento (el *procedimiento con fuente*) que está basado en otro procedimiento (el *procedimiento fuente*). En sistemas federados, se entiende por *procedimiento federado* un procedimiento con fuente cuyo procedimiento fuente se encuentra en una fuente de datos soportada.

Invocación

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

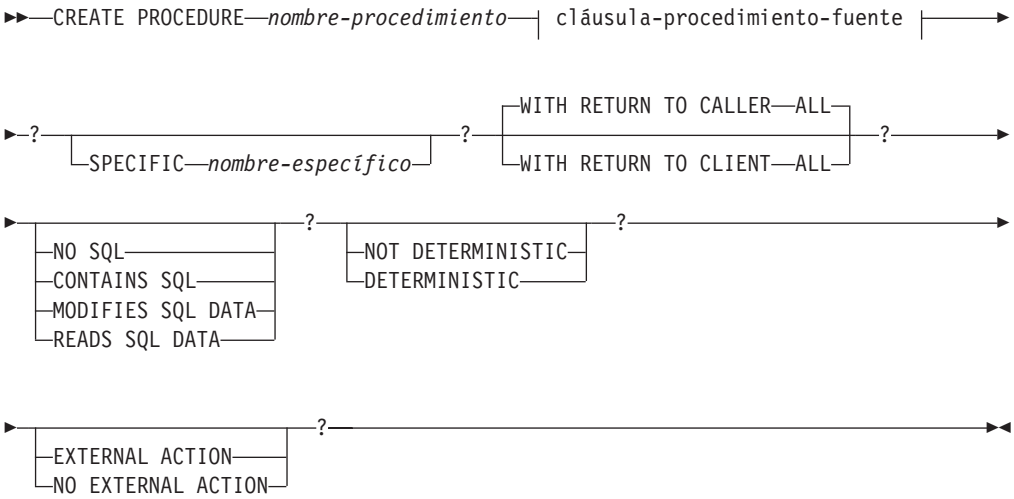
Autorización

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

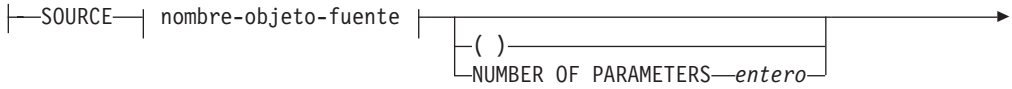
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del procedimiento no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente
- Autorización SYSADM o DBADM

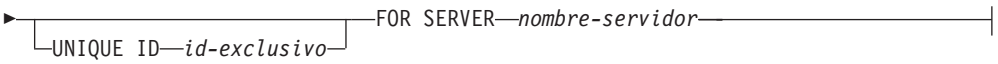
Para los orígenes de datos que requieren una correlación de usuarios, los privilegios contenidos en el ID de autorización en la fuente de datos deben incluir el privilegio para seleccionar datos de las tablas del catálogo remoto.

Sintaxis

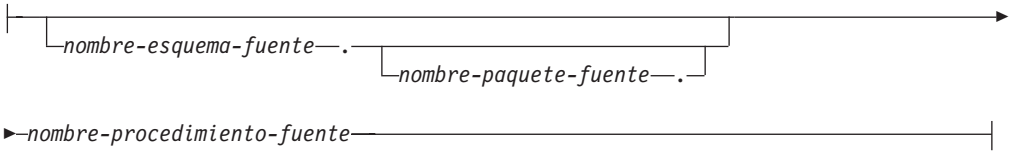


cláusula-procedimiento-fuente:





nombre-objeto-fuente:



Descripción

nombre-procedimiento
Los nombres del procedimiento con fuente 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 sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

El nombre, incluidos los calificadores implícitos o explícitos, junto con el número de parámetros, no deben identificar un procedimiento que se haya descrito en el catálogo (SQLSTATE 42723). No es necesario que el nombre no calificado, junto con el número de parámetros, sea exclusivo en los esquemas.

Si se especifica un nombre de dos partes, el *nombre-esquema* no puede empezar por 'SYS' (SQLSTATE 42939).

En un sistema federado, un *nombre-procedimiento* es el nombre del procedimiento del servidor federado.

SOURCE *nombre-objeto-fuente*
Especifica el procedimiento fuente que utiliza el procedimiento que se está definiendo. En sistemas federados, el procedimiento fuente es un procedimiento que se encuentra en una fuente de datos soportada.

nombre-esquema-fuente
Identifica el nombre de esquema del procedimiento fuente. Si se utiliza un nombre de esquema para identificar el procedimiento fuente, el *nombre-esquema-fuente* debe especificarse en la sentencia CREATE PROCEDURE (con fuente). Si el *nombre-esquema-fuente* contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

nombre-paquete-fuente
Identifica el nombre de paquete del procedimiento fuente. El *nombre-paquete-fuente* sólo se aplica a los orígenes de datos Oracle. Si se utiliza un nombre de esquema para identificar el procedimiento fuente, el *nombre-paquete-fuente* debe especificarse en la sentencia CREATE PROCEDURE (con fuente). Si el *nombre-paquete-fuente* contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

nombre-procedimiento-fuente
Identifica el nombre de procedimiento del procedimiento fuente. Si el

nombre-procedimiento-fuente contiene cualquier carácter especial o caracteres en minúsculas, se debe encerrar entre comillas dobles.

( ) Indica que el número de parámetros es cero.

NUMBER OF PARAMETERS *entero*
Especifica el número de parámetros para el procedimiento fuente. El valor mínimo para *entero* es 0 y el valor máximo es 32.767.

UNIQUE ID *constante-serie*
Proporciona un modo exclusivo de identificar de modo exclusivo el procedimiento fuente cuando hay varios procedimientos en la fuente de datos con el mismo nombre, esquema y número de parámetros. El valor de la *constante-serie* que tiene una longitud máxima de 128, se interpreta exclusivamente mediante cada fuente de datos.

FOR SERVER *nombre-servidor*
Especifica una definición de servidor que se ha registrado utilizando la sentencia CREATE SERVER.

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia del procedimiento con fuente que se está definiendo. Este nombre específico puede utilizarse al eliminar el procedimiento con fuente o asignarle un comentario. Este nombre no puede utilizarse nunca para invocar el procedimiento con fuente. El formato no calificado del *nombre-específico* es un identificador SQL con una longitud máxima de 18. El formato calificado del *nombre-específico* es un *nombre-esquema* seguido de un punto y un identificador SQL. El valor del *nombre-específico*, incluido el calificador implícito o explícito, no debe identificar otra instancia de función que exista en el servidor de aplicaciones; de lo contrario se produce un error (SQLSTATE 42710).

El *nombre-específico* puede ser igual a un *nombre-procedimiento* existente.

Si no se especifica ningún calificador, se emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, 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: 'SQL*aammddhhmmssxxx*'.

WITH RETURN TO CALLER o WITH RETURN TO CLIENT
Indica si se ha manejado el conjunto de resultados del procedimiento fuente. Sólo se devuelve un conjunto de resultados al cliente o al que llama. Si el procedimiento fuente se ha codificado para que devuelva más de un conjunto de resultados, sólo se devuelve el primer conjunto de resultados al cliente o al que llama. El valor por omisión es WITH RETURN TO CALLER.

WITH RETURN TO CALLER ALL
Especifica que se devuelve el conjunto de resultados del procedimiento fuente al que llama.

WITH RETURN TO CLIENT ALL
Especifica que se devuelve el conjunto de resultados del procedimiento fuente directamente a la aplicación de cliente. Sólo es aplicable cuando el valor de los conjuntos de resultados dinámicos en la fuente de datos es mayor que 0 y en el caso de una llamada de procedimiento anidada.

NO SQL, CONTAINS SQL, MODIFIES SQL DATA, READS SQL DATA
Indica el nivel de acceso a datos para las sentencias de SQL que se han

incluido en el procedimiento con fuente. Debido a que el procedimiento fuente del procedimiento con fuente no está situado en el servidor federado, el nivel especificado no se aplica durante la ejecución del procedimiento fuente en la fuente de datos. Si hay alguna discrepancia entre lo que se especifica para el procedimiento con fuente y lo que hace realmente el procedimiento con fuente en la fuente de datos, es posible que se produzcan incoherencias de datos. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es MODIFIES SQL DATA. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

DETERMINISTIC o NOT DETERMINISTIC
Especifica si el procedimiento con fuente devuelve siempre los mismos resultados para los valores de argumentos determinados (DETERMINISTIC) o si el procedimiento con fuente depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Un procedimiento con fuente DETERMINISTIC debe siempre devolver el mismo resultado si se invoca de forma sucesiva con entradas de datos idénticas. Esta cláusula no afecta actualmente el proceso del procedimiento. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es NOT DETERMINISTIC. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

EXTERNAL ACTION o NO EXTERNAL ACTION
Especifica si el procedimiento con fuente realiza alguna acción que cambia el estado de un objeto que el gestor de bases de datos no gestiona (EXTERNAL ACTION) o no (NO EXTERNAL ACTION). Si se especifica la cláusula NO EXTERNAL ACTION, la base de datos federada utiliza la optimización que presupone que el procedimiento con fuente no tiene un impacto externo. Si esta opción no se especifica explícitamente, se utiliza el valor del procedimiento fuente. Si no está disponible esta opción en la fuente de datos, el valor por omisión es EXTERNAL ACTION. Si se especifica esta opción explícitamente pero no coincide con el valor del procedimiento fuente, se devuelve un error (SQLSTATE 428GS).

Normas

- Si *nombre-objeto-fuente*, junto con las cláusulas NUMBER OF PARAMETERS y UNIQUE ID no identifican un procedimiento en una fuente de datos, se devuelve un error (SQLSTATE 42883). Si se identifica más de un procedimiento, se devuelve un error (SQLSTATE 42725).
- Si se especifica la cláusula UNIQUE ID y la fuente de datos no da soporte a los ID exclusivos, se devuelve un error (SQLSTATE 42883).

Notas

- Para poder registrar un procedimiento federado para una fuente de datos, el servidor federado debe estar configurado para acceder a esa fuente de datos. Esta configuración incluye la creación de un derivador para la fuente de datos, la definición del servidor para el servidor para la fuente de datos y la creación de las correlaciones de usuarios entre el servidor federado y el servidor fuente de datos para los orígenes de datos que requieren correlaciones de usuarios.
- A diferencia de los procedimientos de SQL y externos definidos en el servidor federado, los procedimientos federados no heredan los registros especiales del

que llama, incluso aquellos cuyos *nombre-objeto-remoto* hace referencia a un procedimiento de una fuente de datos DB2.

- Si se cambia la definición del procedimiento fuente (por ejemplo, se modifica un tipo de datos de parámetro), se debe desactivar y volver a crear el procedimiento federado; de lo contrario, se pueden producir errores cuando se invoque el procedimiento federado.
- Si la longitud del parámetro de procedimiento fuente es mayor de 128, el nombre de parámetro del procedimiento federado se trunca en 128 bytes.
- ***Compatibilidades***
  - La sintaxis de DataJoiner para crear apodos de procedimientos almacenados no está soportada. En la sintaxis de la nueva versión 9, la correlación de tipos de parámetros se maneja de modo similar a los apodos, ya que una búsqueda de catálogo determina el tipo de datos remoto. El tipo de parámetro local se determina mediante la correlación de tipos hacia adelante.

Ejemplos

Ejemplo 1: Crear un procedimiento federado cuyo nombre es FEDEMPLOYEE para un procedimiento Oracle cuyo nombre es EMPLOYEE, utilizando el nombre de esquema remoto USER1, el nombre de paquete remoto P1 en el servidor federado S1, y devolviendo el conjunto de resultados al cliente.

```
CREATE PROCEDURE FEDEMPLOYEE SOURCE USER1.P1.EMPLOYEE
  FOR SERVER S1 WITH RETURN TO CLIENT ALL
```

CREATE PROCEDURE (SQL)

La sentencia CREATE PROCEDURE (SQL) define un procedimiento de SQL en el servidor actual.

Invocación

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

Autorización

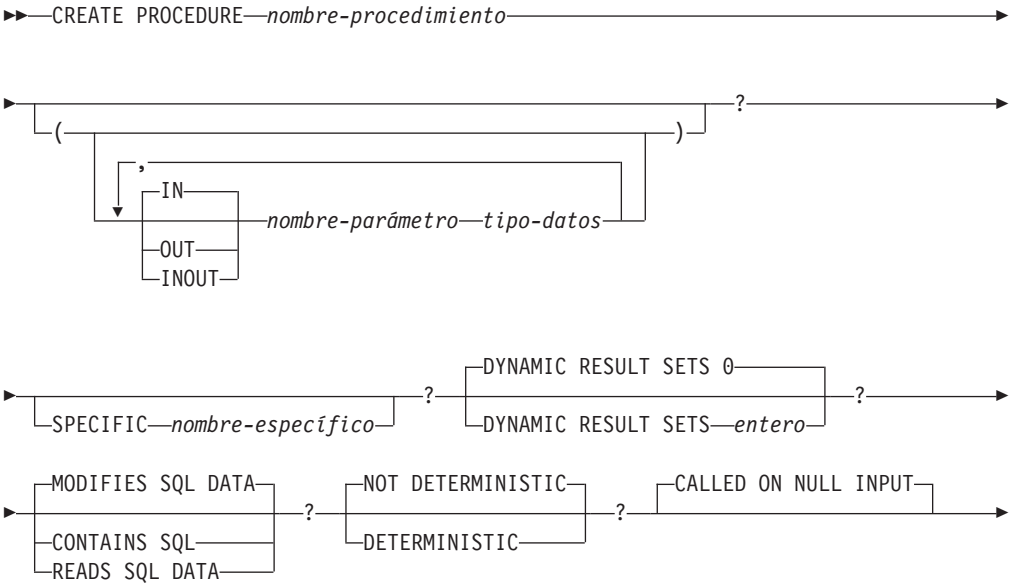
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

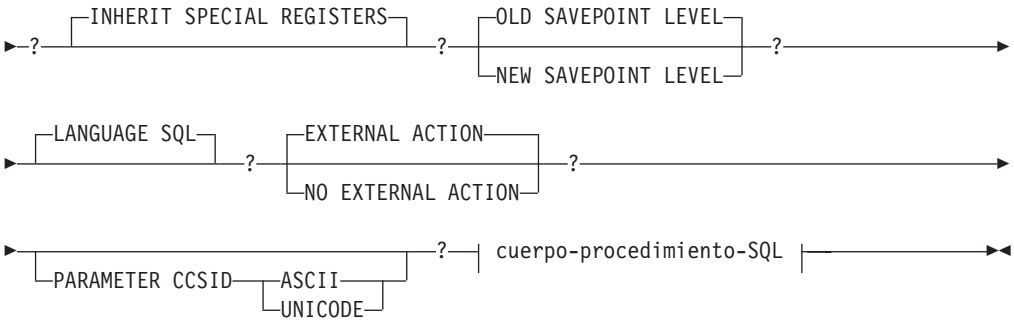
- Privilegio BINDADD para la base de datos, y uno de los siguientes:
  - Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito del procedimiento no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema del procedimiento hace referencia a un esquema existente
- Autorización SYSADM o DBADM

Si el definidor de un procedimiento sólo puede crear el procedimiento porque el definidor tiene la autorización SYSADM, se le otorgará la autorización DBADM implícita con el propósito de crear el procedimiento.

Si el ID de autorización de la sentencia no tiene la autorización SYSADM o DBADM, los privilegios del ID de autorización de la sentencia deben incluir también todos los privilegios necesarios para invocar las sentencias de SQL especificadas en el cuerpo del procedimiento.

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 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 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.

Si el procedimiento devuelve un error, los parámetros OUT no se definen y los parámetros INOUT no cambian.

IN Identifica el parámetro como un parámetro de entrada para el procedimiento. Los cambios que se realicen en el parámetro dentro del procedimiento no estarán disponibles para la aplicación de SQL que realiza la llamada cuando se devuelva el control. El valor por omisión es IN.

OUT Identifica el parámetro como un parámetro de salida para el procedimiento.

INOUT
Identifica el parámetro como parámetro de entrada y de salida para el procedimiento.

nombre-parámetro
Especifica el nombre del parámetro. El nombre del parámetro debe ser exclusivo para el procedimiento (SQLSTATE 42734).

tipo-datos
Especifica el tipo de datos del parámetro.

- 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, 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 emplea el que se haya utilizado para el *nombre-procedimiento*. Si se especifica un calificador, éste deberá ser el mismo que el calificador explícito o implícito del *nombre-procedimiento* o se generará un error (SQLSTATE 42882).

Si no se especifica el *nombre-específico*, el gestor de bases de datos genera un nombre exclusivo. El nombre exclusivo será 'SQL' seguido de una indicación de fecha y hora expresada en forma de caracteres: 'SQL*aammddhhmmssxxx*'.

DYNAMIC RESULT SETS *entero*
Indica el límite superior estimado de los conjuntos de resultados devueltos para el procedimiento.

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA
Indica el nivel de acceso a datos para las sentencias de SQL que se han incluido en el procedimiento.

CONTAINS SQL
Indica que el procedimiento puede ejecutar sentencias de SQL que no lean ni modifiquen datos de SQL (SQLSTATE 38004 ó 42985). Es posible que las sentencias que no reciban soporte en los procedimientos devuelvan otro error (SQLSTATE 38003 o 42985).

READS SQL DATA
Indica que en el procedimiento pueden incluirse algunas sentencias de SQL que no modifiquen datos de SQL (SQLSTATE 38002 ó 42985). Es posible que las sentencias que no reciban soporte en los procedimientos devuelvan otro error (SQLSTATE 38003 o 42985).

MODIFIES SQL DATA
Indica que el procedimiento puede ejecutar cualquier sentencia de SQL excepto las que no están soportadas en ningún procedimiento (SQLSTATE 38003 ó 42985).

Si se utiliza la cláusula BEGIN ATOMIC en un procedimiento de SQL compuesto, sólo se podrá crear el procedimiento si está definido como MODIFIES SQL DATA.

DETERMINISTIC o NOT DETERMINISTIC
Esta cláusula especifica si el procedimiento devuelve siempre los mismos resultados para unos valores argumento determinados (DETERMINISTIC) o si el procedimiento depende de algunos valores de estado que afectan a los resultados (NOT DETERMINISTIC). Es decir, un procedimiento DETERMINISTIC debe siempre devolver el mismo resultado ante invocaciones sucesivas con entradas de datos idénticas.

Actualmente, esta cláusula no afecta al proceso del procedimiento.

CALLED ON NULL INPUT
CALLED ON NULL INPUT siempre se aplica a los procedimientos. Esto significa que el procedimiento recibe llamada independientemente de si algún argumento es nulo. Cualquier parámetro OUT o INOUT puede devolver un valor nulo o un valor normal (no nulo). Corresponde al procedimiento comprobar si hay valores de argumento nulos.

INHERIT SPECIAL REGISTERS
Esta cláusula opcional especifica que los registros especiales que pueden actualizarse del procedimiento heredarán sus valores iniciales del entorno de la sentencia que realiza la invocación. Para una rutina que se invoca en un objeto anidado (por ejemplo, un activador o una vista), los valores iniciales se heredan del entorno de ejecución (no se heredan de la definición del objeto).

Al proceso que realiza la llamada al procedimiento no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Los registros especiales que no pueden actualizarse como, por ejemplo, los registros especiales de indicación de fecha y hora, reflejan una propiedad de la sentencia que está ejecutándose actualmente y, por lo tanto, se establecen en sus valores por omisión.

OLD SAVEPOINT LEVEL o NEW SAVEPOINT LEVEL
Especifica si este procedimiento establece o no un nuevo nivel de punto de salvaguarda para los nombres y efectos de punto de salvaguarda. OLD SAVEPOINT LEVEL es el comportamiento por omisión. Para obtener más información sobre los niveles de punto de salvaguarda, consulte “Normas” en “SAVEPOINT”.

LANGUAGE SQL
Esta cláusula se utiliza para especificar que el cuerpo de procedimiento se ha escrito en el lenguaje SQL.

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

PARAMETER CCSID
Especifica el esquema de codificación que se debe utilizar para todos los datos de serie pasados al procedimiento y desde él. Si no se especifica la cláusula PARAMETER CCSID, el valor por omisión es PARAMETER CCSID UNICODE para las bases de datos Unicode y PARAMETER CCSID ASCII para todas las demás bases de datos.

ASCII
Especifica que los datos de serie están codificados en la página de códigos de la base de datos. Si la base de datos es Unicode, no se puede especificar PARAMETER CCSID ASCII (SQLSTATE 56031).

UNICODE
Especifica que los datos de caracteres están en UTF-8, y que los datos gráficos están en UCS-2. Si la base de datos no es Unicode, no se puede especificar PARAMETER CCSID UNICODE (SQLSTATE 56031).

cuerpo-procedimiento-SQL
Especifica la sentencia de SQL que forma el cuerpo del procedimiento de SQL. Se pueden especificar varias sentencias de procedimiento de SQL dentro de una sentencia compuesta de procedimiento. Consulte *sentencia-procedimiento-SQL* en “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

- 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.

- *Compatibilidades*
  - Para mantener la compatibilidad con DB2 para 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.

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
```

CREATE ROLE

La sentencia CREATE ROLE define una función en el servidor actual.

Invocación

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

Autorización

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

Sintaxis

Descripción

nombre-función
Da nombre a la función. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar una función existente en el servidor actual (SQLSTATE 42710). El nombre no puede empezar por los caracteres 'SYS' y no debe ser 'DBADM', 'NONE', 'NULL', 'PUBLIC' ni 'SECADM' (SQLSTATE 42939).

Ejemplo

Cree una función denominada DOCTOR.

```
CREATE ROLE DOCTOR
```

CREATE SCHEMA

La sentencia CREATE SCHEMA define un esquema. También es posible crear algunos objetos y otorgar privilegios en los objetos dentro de la sentencia.

Invocación

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

Autorización

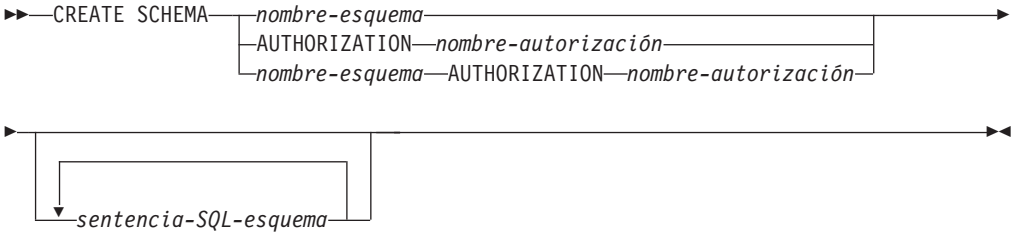
Un ID de autorización que tenga la autorización SYSADM o DBADM puede crear un esquema con cualquier *nombre-esquema* o *nombre-autorización* válido.

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 una *sentencia-SQL-esquema*, los privilegios que mantiene el *nombre-autorización* (el cual, si no se especifica, toma por omisión el ID de autorización de la sentencia) deben 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 incorporado se compilan en momentos diferentes, los resultados dependen del momento en que se define la tabla temporal declarada. Estas cuestiones no se plantean cuando no se utiliza el nombre de esquema "SESSION" para definir una tabla permanente, vista o alias.

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) )
```

CREATE SECURITY LABEL COMPONENT

La sentencia CREATE SECURITY LABEL COMPONENT define un componente que se utilizará como parte de una política de seguridad.

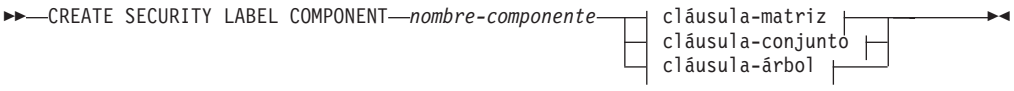
Invocación

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

Autorización

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

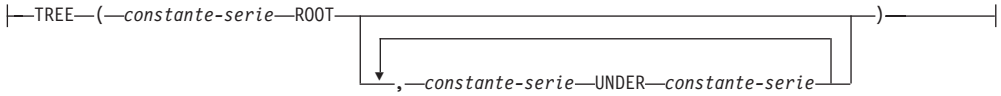
Sintaxis



cláusula-matriz:

cláusula-conjunto:

cláusula-árbol:



Descripción

nombre-componente
Asigna un nombre al componente de etiqueta de seguridad. Este nombre consta de una sola parte. El nombre no debe identificar un componente de etiqueta de seguridad existente en el servidor actual (SQLSTATE 42710).

ARRAY
Especifica un conjunto ordenado de elementos.

constante-serie,...
Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad. Es

importante el orden en que aparecen los elementos de la matriz. El primer elemento tiene una clasificación superior al segundo. El segundo elemento tiene una clasificación superior al tercero, y así sucesivamente.

SET
Especifica un conjunto no ordenado de elementos.

*constante-serie***,...**
Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad. El orden de los elementos no es importante.

TREE
Especifica una estructura de árbol de elementos de nodos.

constante-serie
Uno o más valores de constante de tipo serie que forman el conjunto de valores válidos para este componente de etiqueta de seguridad.

ROOT
Especifica que la *constante-serie* que sigue a la palabra clave es el elemento del nodo raíz del árbol.

UNDER
Especifica que la *constante-serie* antes de la palabra clave **UNDER** es hija de la *constante-serie* que sigue a la palabra clave **UNDER**. Se debe definir un elemento como elemento raíz o como el hijo de otro elemento antes de que se pueda utilizar como padre; de lo contrario, se devuelve un error (SQLSTATE 42704).

Normas

Estas normas se aplican a los tres tipos de componentes (ARRAY, SET y TREE):

- Los nombres de elementos no pueden contener ninguno de estos caracteres:
  - Paréntesis de apertura - (
  - Paréntesis de cierre -)
  - Coma - ,
  - Dos puntos - :
- Un nombre de elemento no puede tener más de 32 bytes (SQLSTATE 42622).
- Si un componente de etiqueta de seguridad es un conjunto o un árbol, no podrán formar parte de este componente más de 64 elementos.
- Una sentencia CREATE SECURITY LABEL COMPONENT puede especificar como máximo 65.535 para un componente de etiqueta de seguridad de tipo de matriz.
- Ningún nombre de elemento se puede utilizar más de una vez en el mismo componente (SQLSTATE 42713).

Ejemplos

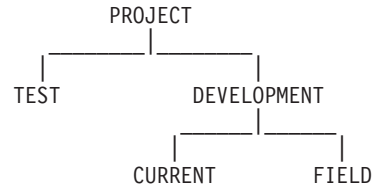
Ejemplo 1: Crear un componente de etiqueta de seguridad del tipo ARRAY llamado LEVEL. El componente tiene los cuatro elementos siguientes, enumerados en orden de clasificación decreciente: Top Secret, Secret, Classified y Unclassified.

```
CREATE SECURITY LABEL COMPONENT LEVEL
  ARRAY ['Top Secret', 'Secret', 'Classified', 'Unclassified']
```

Ejemplo 2: Crear un componente de etiqueta de seguridad del tipo SET llamado COMPARTMENTS. El componente tiene los tres elementos siguientes: Research, Analysis y Collection.

```
CREATE SECURITY LABEL COMPONENT COMPARTMENTS
  SET {'Collection', 'Research', 'Analysis'}
```

Ejemplo 3: Crear un componente de etiqueta de seguridad del tipo TREE llamado GROUPS. GROUPS tiene cinco elementos: PROJECT, TEST, DEVELOPMENT, CURRENT y FIELD. El diagrama siguiente muestra la relación de estos elementos entre sí:




```
CREATE SECURITY LABEL COMPONENT GROUPS
  TREE (
    'PROJECT' ROOT,
    'TEST' UNDER 'PROJECT',
    'DEVELOPMENT' UNDER 'PROJECT',
    'CURRENT' UNDER 'DEVELOPMENT',
    'FIELD' UNDER 'DEVELOPMENT'
  )
```

CREATE SECURITY LABEL

La sentencia CREATE SECURITY LABEL define una etiqueta de seguridad.

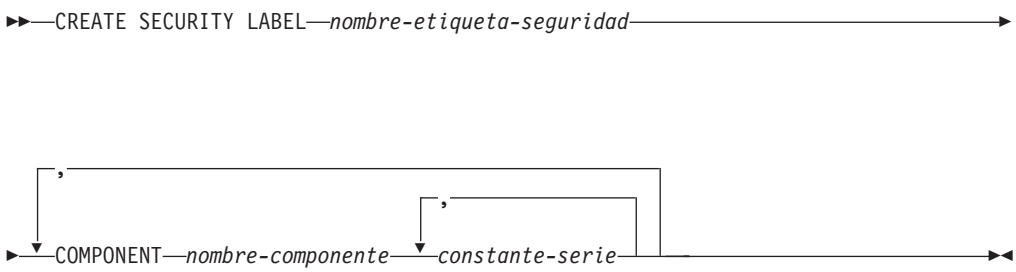
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-etiqueta-seguridad
Asigna un nombre a la etiqueta de seguridad. El nombre debe calificarse con una política de seguridad (SQLSTATE 42704) y no debe identificar una etiqueta de seguridad existente para esta política de seguridad (SQLSTATE 42710).

COMPONENT *nombre-componente*
Especifica el nombre de un componente de etiqueta de seguridad. Si el componente no forma parte de la política de seguridad *nombre-política-seguridad*, se devuelve un error (SQLSTATE 4274G). Si un componente se especifica dos veces en la misma sentencia, se devuelve un error (SQLSTATE 42713).

*constante-serie***,...**
Especifica un elemento válido para el componente de seguridad. Un elemento válido es el que se especificó cuando se creó el componente de seguridad. Si el elemento no es válido, se devuelve un error (SQLSTATE 4274F).

Ejemplos

Ejemplo 1: Crear una etiqueta de seguridad llamada EMPLOYEESECLABEL que forme parte de la política de seguridad DATA_ACCESS y que tenga el elemento Top Secret para el componente LEVEL, así como los elementos Research y Analysis para el componente COMPARTMENTS.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research', 'Analysis'
```

Ejemplo 2: Crear una etiqueta de seguridad llamada EMPLOYEESECLABELREAD que tenga el elemento Top Secret para el componente LEVEL y el elemento Research para el componente COMPARTMENTS.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research'
```

Ejemplo 3: Crear una etiqueta de seguridad llamada EMPLOYEESECLABELWRITE que tenga el elemento Analysis para el componente COMPARTMENTS y un valor nulo para el componente LEVEL. Suponer que la política de seguridad llamada DATA_ACCESS es la misma política de seguridad que la que se utiliza en los ejemplos 1 y 2.

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
  COMPONENT COMPARTMENTS 'Analysis'
```

Ejemplo 4: Crear una etiqueta de seguridad llamada BEGINNER que forme parte de una política de seguridad CLASSPOLICY existente, y que tenga el elemento Trainee para el componente TRUST y el elemento Morning para el componente SECTIONS.

```
CREATE SECURITY LABEL CLASSPOLICY.BEGINNER
  COMPONENT TRUST 'Trainee',
  COMPONENT SECTIONS 'Morning'
```

CREATE SECURITY POLICY

La sentencia CREATE SECURITY POLICY define una política de seguridad.

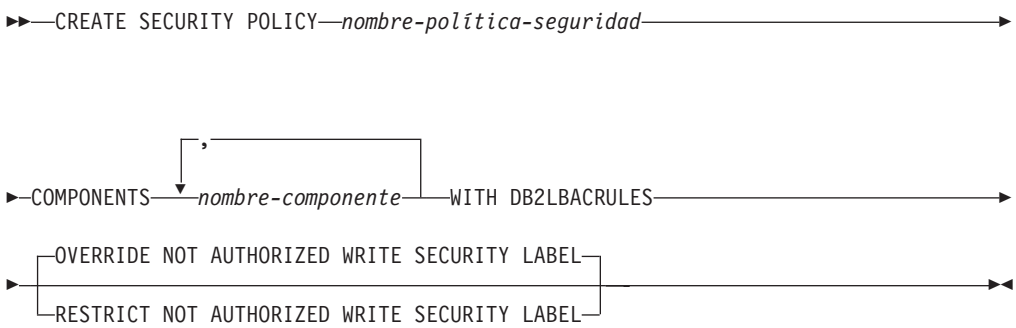
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-política-seguridad
Asigna un nombre a la política de seguridad. Este nombre consta de una sola parte. El nombre no debe identificar una política de seguridad existente en el servidor actual (SQLSTATE 42710).

COMPONENTS *nombre-componente*,...
Identifica un componente de etiqueta de seguridad. El nombre debe identificar un componente de etiqueta de seguridad que ya existe en el servidor actual (SQLSTATE 42704). No se debe especificar el mismo componente de seguridad más de una vez para la política de seguridad (SQLSTATE 42713). No se pueden especificar más de 16 componentes de etiqueta de seguridad para una política de seguridad (SQLSTATE 54062).

WITH DB2LBACRULES
Indica qué conjunto de normas se utilizará al comparar las etiquetas de seguridad que forman parte de esta política de seguridad. Actualmente sólo hay un conjunto de normas: DB2LBACRULES.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL o RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Especifica la acción que se llevará a cabo cuando un usuario no tenga autorización para grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia INSERT o UPDATE emitida sobre una tabla que está protegida con esta política de seguridad. Una etiqueta de seguridad

del usuario y las credenciales de exención determinan la autorización del usuario para grabar una etiqueta de seguridad proporcionada explícitamente. El valor por omisión es OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL.

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
Indica que el valor de la etiqueta de seguridad del usuario se usará para el acceso de grabación durante una operación de inserción o de actualización, en lugar de la etiqueta de seguridad especificada explícitamente.

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
Indica que la operación de inserción o actualización no será satisfactoria si el usuario no está autorizado a grabar la etiqueta de seguridad especificada explícitamente que se proporciona en la sentencia INSERT o UPDATE (SQLSTATE 42519).

Notas

- ***Conjunto de normas DB2LBACRULES:*** DB2LBACRULES es un conjunto de normas predefinido que incluye las siguientes normas: DB2LBACREADARRAY, DB2LBACREADSET, DB2LBACREADTREE, DB2LBACWRITEARRAY, DB2LBACWRITESET, DB2LBACWRITETREE.
- Las autorizaciones de función y de grupo no se tienen en cuenta por omisión cuando se crea una política de seguridad. Utilice la sentencia ALTER SECURITY POLICY para cambiar este comportamiento y tenerlas en cuenta.

Ejemplos

Ejemplo 1: Crear una política de seguridad llamada DATA_ACCESS que utilice el conjunto de normas DB2LBACRULES y que tenga dos componentes: LEVEL y COMPARTMENTS, en este orden. Suponer que los dos componentes ya existen.

```
CREATE SECURITY POLICY DATA_ACCESS
  COMPONENTS LEVEL, COMPARTMENTS
  WITH DB2LBACRULES
```

Ejemplo 2: Crear una política de seguridad llamada CONTRIBUTIONS que tenga los componentes MEMBER y BADGE, que se supone que ya existen.

```
CREATE SECURITY POLICY CONTRIBUTIONS
  COMPONENTS MEMBER, BADGE
  WITH DB2LBACRULES
```

CREATE SEQUENCE

La sentencia CREATE SEQUENCE define una secuencia en el servidor de aplicaciones.

Invocación

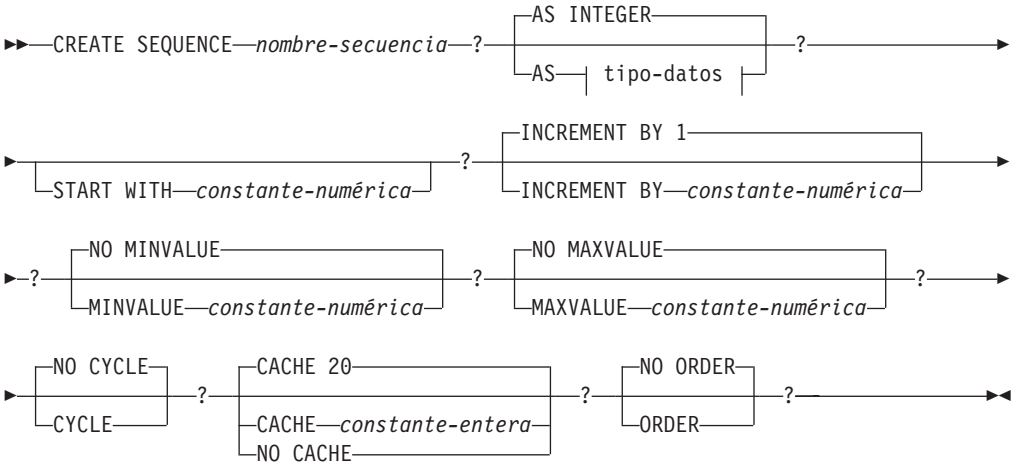
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

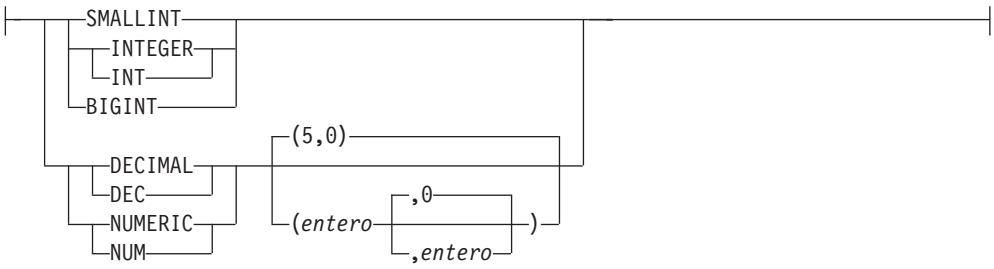
- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la secuencia no existe
- Privilegio CREATEIN en el esquema, si el nombre de esquema de la secuencia hace referencia a un esquema existente
- Autorización SYSADM o DBADM

Sintaxis



tipo-datos:

tipo-incorporado:



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. Éste es el valor por omisión.

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

MAXVALUE *constante-numérica*
Especifica la constante numérica que es el valor máximo. Este valor puede ser cualquier valor positivo o negativo que pueda asignarse a una columna del tipo de datos 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. Éste es el valor por omisión.

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

CACHE *constante-entera*
Especifica el número máximo de valores de secuencia que se preasignan y se mantienen en memoria. La preasignación y el almacenamiento de valores en la antememoria 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. Éste es el valor por omisión.

Notas

- Es posible definir una secuencia constante, es decir, una que devuelva siempre un valor constante. Esto puede hacerse especificando un valor cero para INCREMENT y un valor para START WITH que no exceda el valor de MAXVALUE o bien especificando el mismo valor para START WITH, MINVALUE y MAXVALUE. Para una secuencia constante, cada vez que se invoque NEXT VALUE para la secuencia, se devolverá el mismo valor. Una secuencia constante puede utilizarse como variable global numérica. ALTER SEQUENCE puede utilizarse para ajustar los valores que se generarán para una secuencia constante.
- El ciclo de una secuencia puede especificarse manualmente utilizando la sentencia ALTER SEQUENCE. Si se especifica NO CYCLE de forma implícita o explícita, se puede reiniciar o ampliar la secuencia utilizando la sentencia ALTER SEQUENCE para hacer que los valores continúen generándose una vez que se haya alcanzado el valor máximo o mínimo para la secuencia.
- Una secuencia puede definirse explícitamente para que ejecute un ciclo especificando la palabra clave CYCLE. Al definir una secuencia, utilice la opción CYCLE para indicar que los valores generados deben ejecutar un ciclo cuando se

haya alcanzado el límite. Cuando se ha definido una secuencia para la ejecución automática de un ciclo (es decir, CYCLE se ha especificado explícitamente), el valor máximo o mínimo generado para una secuencia puede que no sea el valor MAXVALUE o MINVALUE real que se ha especificado, si el incremento es un valor distinto de 1 ó -1. Por ejemplo, la secuencia que se ha definido con `START WITH=1, INCREMENT=2, MAXVALUE=10` generará un valor máximo de 9 y no generará el valor 10. Al definir una secuencia con CYCLE, considere detenidamente el efecto que pueden tener los valores para MINVALUE, MAXVALUE y START WITH.

- El almacenamiento en antememoria de los números de secuencia implica que un rango de números de secuencia puede mantenerse en la memoria para acceder a ellos rápidamente. Cuando una aplicación accede a una secuencia que puede asignar el siguiente número de secuencia desde la antememoria, la asignación de números de secuencia se produce rápidamente. Sin embargo, si una aplicación accede a una secuencia que no puede asignar el siguiente número de secuencia desde la antememoria, puede que para asignar el número de secuencia sea necesario esperar a que se realicen las operaciones de E/S en el almacenamiento permanente. La elección del valor para CACHE debe realizarse teniendo en cuenta los cambios de requisitos del rendimiento y de la aplicación.
- Al usuario que define una secuencia se le otorgan los privilegios ALTER y USAGE y el privilegio WITH GRANT OPTION. El propietario de la secuencia puede eliminar la secuencia.
- ***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.

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
```

CREATE SERVICE CLASS

La sentencia CREATE SERVICE CLASS define una clase de servicio.

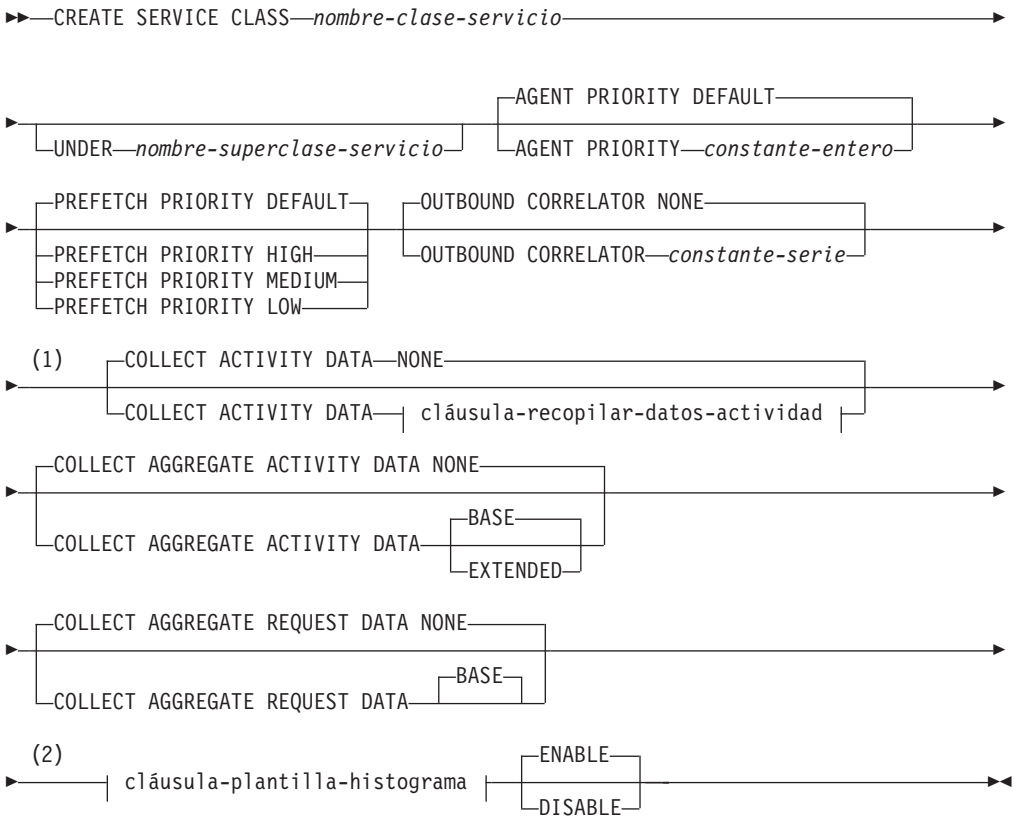
Invocación

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

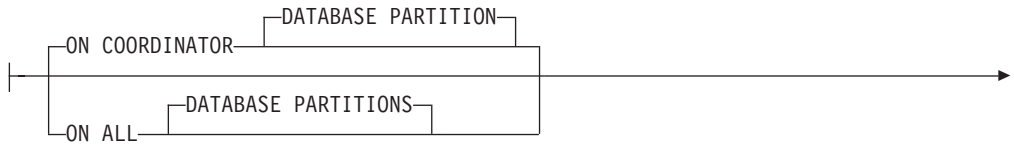
Autorización

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

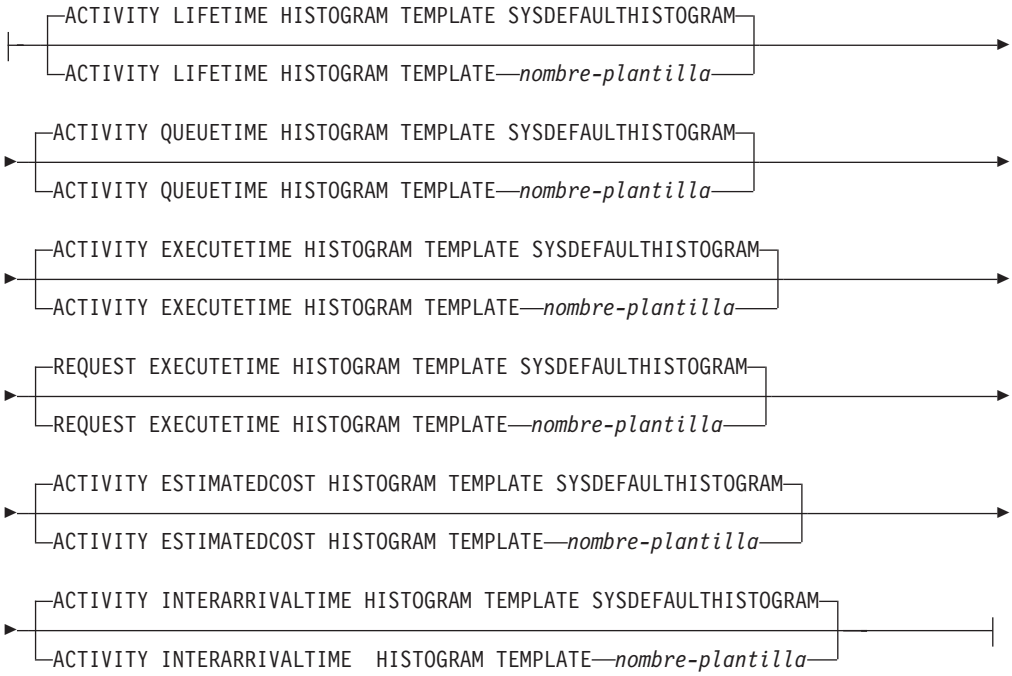
Sintaxis



cláusula-recopilar-datos-actividad:



cláusula-plantilla-histograma:



Notas:

1 Las cláusulas COLLECT sólo son válidas para una subclase de servicio.

2 Las cláusulas HISTOGRAM TEMPLATE sólo son válidas para una subclase de servicio.

Descripción

nombre-clase-servicio
Indica el nombre de la clase de servicio. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). Si la clase de servicio es una superclase de servicio, el *nombre-clase-servicio* no debe identificar ninguna superclase de servicio que ya exista en el catálogo (SQLSTATE 42710). Si la clase de servicio es una subclase de servicio, el *nombre-clase-servicio* no debe identificar ninguna subclase de servicio que ya exista bajo la superclase de servicio (SQLSTATE 42710). Si la clase de servicio es una subclase de servicio, el *nombre-clase-servicio* no debe ser el mismo que su superclase de servicio (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

UNDER *nombre-superclase-servicio*
Especifica que la clase de servicio es una subclase de la superclase de servicio *nombre-superclase-servicio*. Si no se especifica UNDER, la clase de servicio es una superclase de servicio. El *nombre-superclase-servicio* debe identificar una

superclase de servicio que exista para la base de datos (SQLSTATE 42704). La superclase de servicio no puede ser una clase de servicio por omisión (SQLSTATE 5U029).

AGENT PRIORITY DEFAULT o AGENT PRIORITY *constante-entero*
Especifica la prioridad del sistema operativo (delta) relativa de los agentes que se ejecutan en la clase de servicio o la prioridad normal de las hebras que se ejecutan en DB2. El valor por omisión es DEFAULT. Cuando se establece en DEFAULT, no se adopta ninguna acción especial y los agentes de la clase de servicio se planifican con arreglo a la prioridad normal con la que el sistema operativo planifica todas las hebras de DB2. Cuando se establece este parámetro en un valor que no sea DEFAULT, los agentes se establecen en una prioridad que sea igual a la prioridad normal más AGENT PRIORITY cuando comienza la siguiente actividad. Por ejemplo, si la prioridad normal es 20 y AGENT PRIORITY se establece en -10, la prioridad de los agentes de la clase de servicio se establece en 20 – 10 = 10.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

OUTBOUND CORRELATOR NONE o OUTBOUND CORRELATOR *constante-serie*
Especifica si asociar o no hebras de esta clase de servicio a una clase de servicio de gestor de carga de trabajo externa.

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

OUTBOUND CORRELATOR NONE
Para una superclase de servicio, especifica que no hay ninguna asociación de clase de servicio de gestor de carga de trabajo externa con esta clase de servicio y, para una subclase de servicio, especifica que la asociación de clase de servicio de gestor de carga de trabajo externa es la misma que la de su padre.

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

COLLECT ACTIVITY DATA
Especifica que la información sobre cada una de las actividades que se ejecuta en esta clase de servicio se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. El valor por omisión es COLLECT ACTIVITY DATA NONE. La cláusula COLLECT ACTIVITY DATA sólo es válida para una subclase de servicio.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que se ejecutan en esta clase de servicio.

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Sin embargo, los valores y detalles de actividad sólo se recopilarán en la partición de la base de datos del coordinador.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades que se ejecuta en esta clase de servicio se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza su ejecución. El entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que los datos de entorno de compilación y sentencia se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

COLLECT AGGREGATE ACTIVITY DATA
Especifica que los datos de actividades agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. Cuando no se especifica COLLECT AGGREGATE ACTIVITY DATA, el valor por omisión es COLLECT AGGREGATE ACTIVITY DATA NONE. Cuando se especifica COLLECT AGGREGATE ACTIVITY DATA, el valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. La cláusula COLLECT AGGREGATE ACTIVITY DATA sólo es válida para una subclase de servicio.

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

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo de cola de la actividad
- Histograma de tiempo de ejecución de la actividad

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

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

- Histograma de tiempo de llegada de DML de actividad

NONE
Especifica que no debe capturarse ningún dato de actividad agregados para esta clase de servicio.

COLLECT AGGREGATE REQUEST DATA
Especifica que los datos de peticiones agregados deben capturarse para esta clase de servicio y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo especificado por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE REQUEST DATA NONE. La cláusula COLLECT AGGREGATE REQUEST DATA sólo es válida para una subclase de servicio.

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

NONE
Especifica que no debe capturarse ningún dato de petición agregado para esta clase de servicio.

cláusula-plantilla-histograma
Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades que se ejecutan en la clase de servicio. La cláusula HISTOGRAM TEMPLATE sólo es válida para una subclase de servicio.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos de actividades de DB2 en ejecución en la clase de servicio durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, en que las actividades de DB2 en ejecución en la clase de servicio están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, en que las actividades de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma únicamente en la partición de base de datos del coordinador. El tiempo no incluye el tiempo de inactividad. El tiempo de inactividad es el tiempo entre la ejecución de peticiones que pertenecen a la misma actividad cuando no se efectúa ningún trabajo. Un ejemplo de tiempo de inactividad es el tiempo entre que se acaba de abrir un cursor y comienza la captación desde dicho cursor. El valor por omisión es

SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, en que las peticiones de DB2 en ejecución en la clase de servicio están ejecutándose durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la petición se recopila en este histograma en cada una de las particiones de base de datos en las que se ejecuta la petición. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE REQUEST DATA con la opción BASE.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML en ejecución en la clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED. Sólo se toman en consideración las actividades en el nivel de anidamiento 0 para su inclusión en el histograma.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ENABLE o DISABLE
Especifica si pueden correlacionarse o no conexiones y actividades con la clase de servicio. El valor por omisión es ENABLE.

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

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

Normas

- El número máximo de subclases de servicio que pueden crearse bajo una superclase de servicio es 61 (SQLSTATE 5U027).
- El número máximo de superclases de servicio que pueden crearse para una base de datos es 64 (SQLSTATE 5U027).

- Una sentencia de SQL exclusiva de gestión de carga de trabajo (WLM) debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U027). Las sentencias de SQL exclusivas de WLM son:
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
  - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Una subclase por omisión, SYSDEFAULTSUBCLASS, se crea automáticamente para cada superclase de servicio.
- Sólo se permite una sentencia de SQL exclusiva de WLM no confirmada a la vez entre todas las particiones. Si se ejecuta una sentencia de SQL exclusiva de WLM sin confirmar, las siguientes sentencias de SQL exclusivas de WLM esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva de XML actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta después de una sentencia COMMIT, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Cree una superclase de servicio denominada PETSALES. La subclase por omisión para PETSALES se crea automáticamente.

```
CREATE SERVICE CLASS PETSALES
```

Ejemplo 2: Cree una subclase de servicio denominada DOGSALES debajo de la superclase de servicio PETSALES. Establezca la clase de servicio DOGSALES como inhabilitada.

```
CREATE SERVICE CLASS DOGSALES UNDER PETSALES DISABLE
```

Ejemplo 3: Cree una superclase de servicio denominada BARNSALES con una prioridad de captación previa de LOW. La subclase por omisión para BARNSALES se crea automáticamente. Las peticiones de captación previa que someten los agentes de la clase de servicio BARNSALES irán a la cola de captación previa de prioridad baja.

```
   CREATE
SERVICE CLASS BARNSALES PREFETCH PRIORITY LOW
```

CREATE SERVER

La sentencia CREATE SERVER define una fuente de datos para una base de datos federada. En esta sentencia, el término SERVER y los nombres de parámetro que terminan por *-servidor* sólo hacen referencia a fuentes de datos de un sistema federado. No hacen referencia al servidor federado de ese sistema ni a los servidores de aplicaciones DRDA.

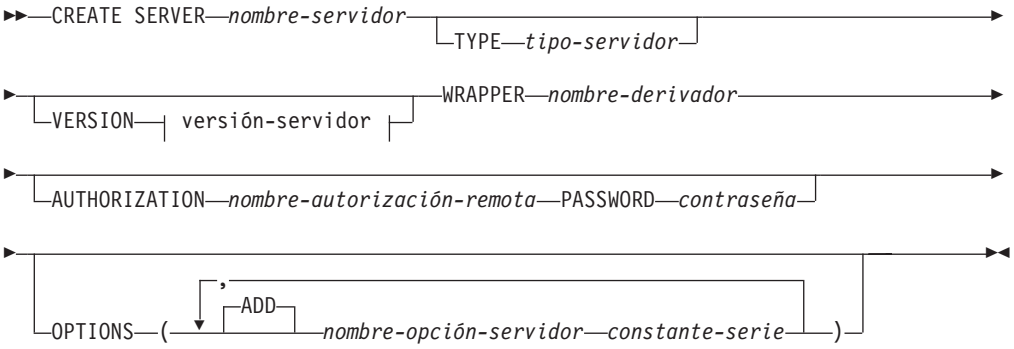
Invocación

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

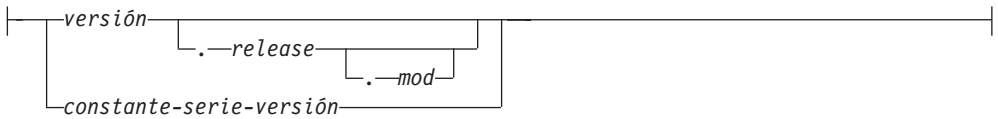
Autorización

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

VERSION
Especifica la versión de la fuente de datos indicada por *nombre-servidor*. Este parámetro es necesario para algunos derivadores.

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

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

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

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

WRAPPER *nombre-derivador*
Nombra el derivador que el servidor federado utiliza para interactuar con el objeto de servidor especificado por *nombre-servidor*.

AUTHORIZATION *nombre-autorización-remota*
Sólo se requiere para las fuentes de datos de la familia DB2. Especifica el ID de autorización bajo el cual se realiza cualquier acción necesaria en la fuente de datos cuando se procesa la sentencia CREATE SERVER. Este ID de autorización no se utiliza al establecer conexiones posteriores con el servidor.

Este ID debe tener la autorización (BINDADD o su equivalente) que las acciones necesarias necesitan. Si se especifica el *nombre-autorización-remota* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota contiene nombres de autorización sensibles a las mayúsculas y minúsculas), el *nombre-autorización-remota* se debe especificar entre comillas dobles.

PASSWORD *contraseña*
Sólo se requiere para las fuentes de datos de la familia DB2. Especifica la contraseña asociada al ID de autorización representado por *nombre-autorización-remota.* Si se especifica la *contraseña* en minúsculas o mezclando caracteres en mayúsculas y minúsculas (y la fuente de datos remota tiene contraseñas sensibles a las mayúsculas y minúsculas), la *contraseña* se debe especificar entre comillas dobles.

OPTIONS
Indica las opciones que se habilitan cuando se crea la definición de servidor. Las opciones de servidor se utilizan para configurar la definición de servidor.

Algunas opciones de servidor pueden utilizarse a fin de crear la definición de servidor para cualquier fuente de datos. Algunas opciones de servidor son específicas para una fuente de datos en particular.

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.
- DB2 no comprueba que la versión del servidor especificada coincida con la versión del servidor remoto. Si se especifica una versión de servidor incorrecta se pueden provocar errores de SQL cuando se accede a apodos que pertenecen a la definición del servidor DB2. Esto puede producirse con mucha probabilidad cuando se especifica una versión de servidor posterior a la versión del servidor remoto. En ese caso, al acceder a apodos que pertenecen a la definición del servidor, puede que DB2 envíe SQL que el servidor remoto no reconoce.

Ejemplos

Ejemplo 1: Registre una definición de servidor para acceder a la fuente de datos DB2 para z/OS y OS/390, versión 7.1. CRANDALL es el nombre asignado a la definición del servidor DB2 para z/OS y OS/390. DRDA es el nombre del derivador utilizado para acceder a esta fuente de datos. Además, especifique que:

- GERALD y drowssap son el ID de autorización y la contraseña bajo los cuales se vinculan los paquetes en CRANDALL cuando se procesa esta sentencia.
- El alias para la base de datos DB2 para z/OS y OS/390 que se ha especificado con la sentencia CATALOG DATABASE es CLIENTS390.
- Los ID de autorización y las contraseñas bajo los que se puede acceder a CRANDALL se deben enviar a CRANDALL en mayúsculas.
- CLIENTS390 y la base de datos federada utilizan el mismo orden de clasificación.

```
CREATE SERVER CRANDALL
  TYPE DB2/ZOS
  VERSION 7.1
  WRAPPER DRDA
  AUTHORIZATION "GERALD"
  PASSWORD drowssap
  OPTIONS
```

```
      (DBNAME 'CLIENTS390',
      FOLD_ID 'U',
      FOLD_PW 'U',
      COLLATING_SEQUENCE 'Y')
```

Ejemplo 2: Registre una definición de servidor para acceder a una fuente de datos Oracle 9. CUSTOMERS es el nombre asignado a la definición de servidor Oracle. NET8 es el nombre del derivador utilizado para acceder a esta fuente de datos. Además, especifique que:

- ABC es el nombre del nodo en el que reside el servidor de bases de datos Oracle.
- La CPU para el servidor federado se ejecuta el doble de rápido que la CPU que soporta CUSTOMERS.
- Los dispositivos de E/S del servidor federado procesan los datos 1,5 veces más rápido que los dispositivos de E/S de CUSTOMERS.

```
CREATE SERVER CUSTOMERS
  TYPE ORACLE
  VERSION 9
  WRAPPER NET8
  OPTIONS
      (NODE 'ABC',
      CPU_RATIO '2.0',
      IO_RATIO '1.5')
```

Ejemplo 3: Registre una definición de servidor para el derivador Excel. La definición de servidor es necesaria para conservar la jerarquía de objetos federados. BIOCHEM_LAB es el nombre asignado a la definición de servidor Excel. EXCEL_2000_WRAPPER es el nombre del derivador utilizado para acceder a esta fuente de datos.

```
CREATE SERVER BIOCHEM_DATA
  WRAPPER EXCEL_2000_WRAPPER
```

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 derivador 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 derivador 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')
```

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 a través del uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

- Autorización CREATETAB en la base de datos y privilegio USE en el espacio de tablas, así como uno de los elementos siguientes:
  - Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito de la tabla no existe
  - Privilegio CREATEIN para el esquema, si el nombre de esquema de la tabla hace referencia a un esquema existente
- Autorización SYSADM o DBADM

Si se define una subtabla, el ID de autorización debe ser el mismo que el propietario de la tabla raíz de la jerarquía de tablas.

Para definir una clave foránea, los privilegios del ID de autorización de la sentencia deben incluir uno de los siguientes en la tabla padre:

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

Para definir una tabla de consulta materializada (utilizando una selección completa), los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes para cada tabla o vista identificada en la selección completa (excluyendo los privilegios de grupo):

- Privilegio SELECT para la tabla o vista, y privilegio ALTER (incluyendo los privilegios de grupo) si se especifica REFRESH DEFERRED o REFRESH IMMEDIATE
- Privilegio CONTROL sobre 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 en la tabla de consultas materializadas:

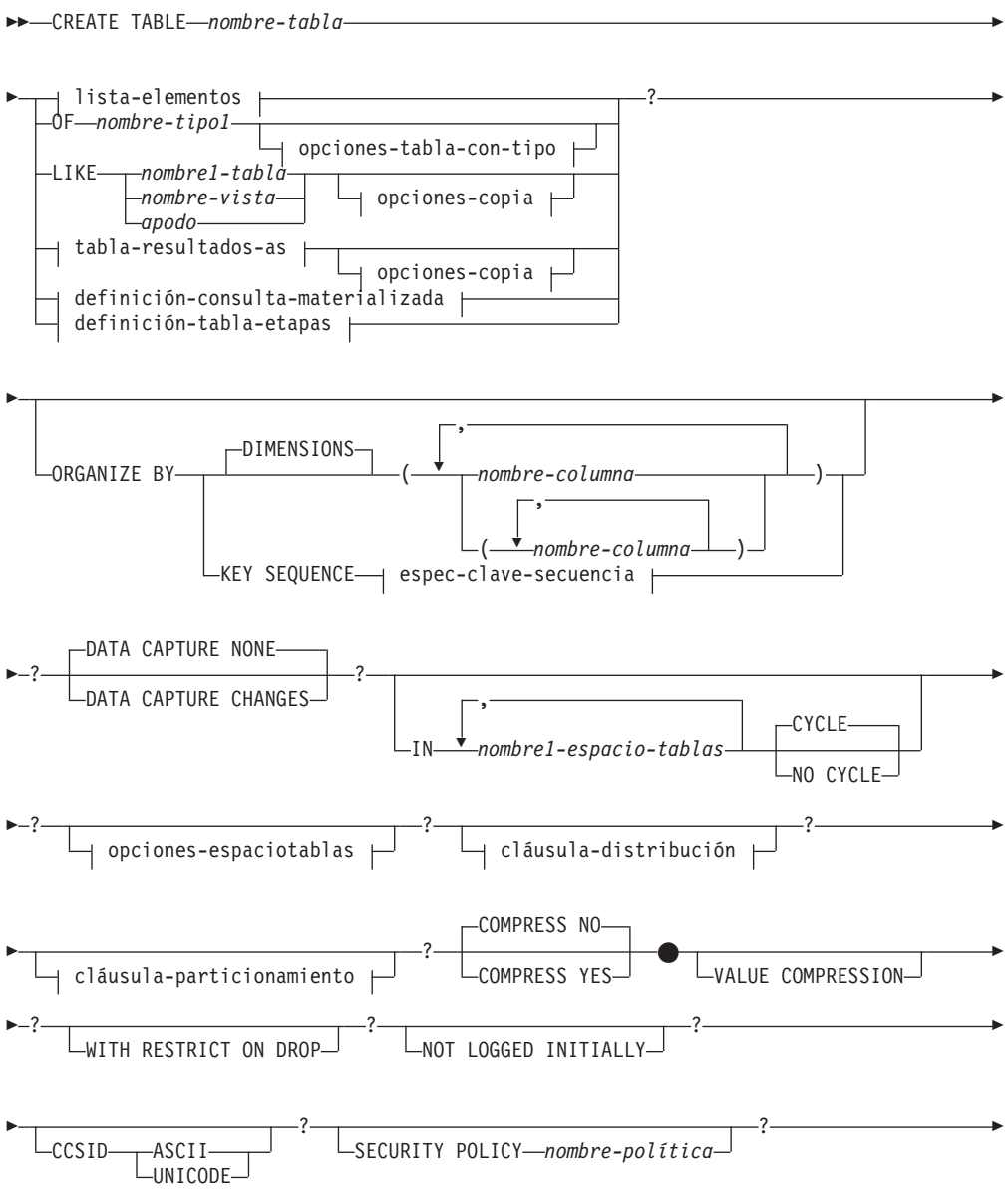
- Privilegio ALTER y SELECT

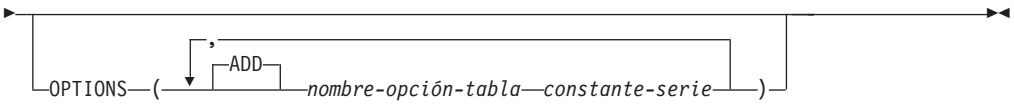
- Privilegio CONTROL
- Autorización SYSADM o DBADM

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 (excluyendo los privilegios de grupo):

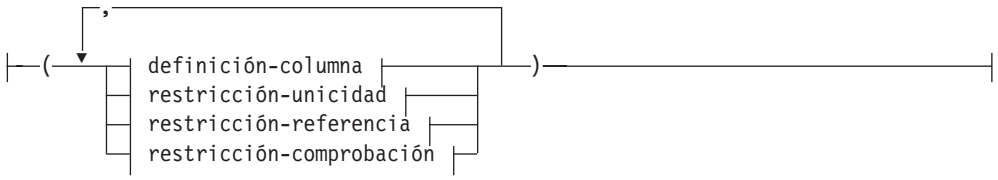
- Privilegio SELECT y privilegio ALTER (incluyendo los privilegios de grupo) para la tabla o vista
- Privilegio CONTROL sobre la tabla o la vista
- Privilegio SELECT para la tabla o vista y privilegio ALTERIN (incluyendo los privilegios de grupo) para el esquema de la tabla o vista
- Autorización SYSADM o DBADM

Sintaxis

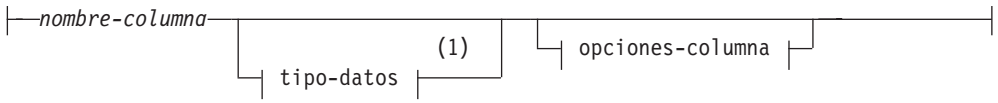




lista-elementos:

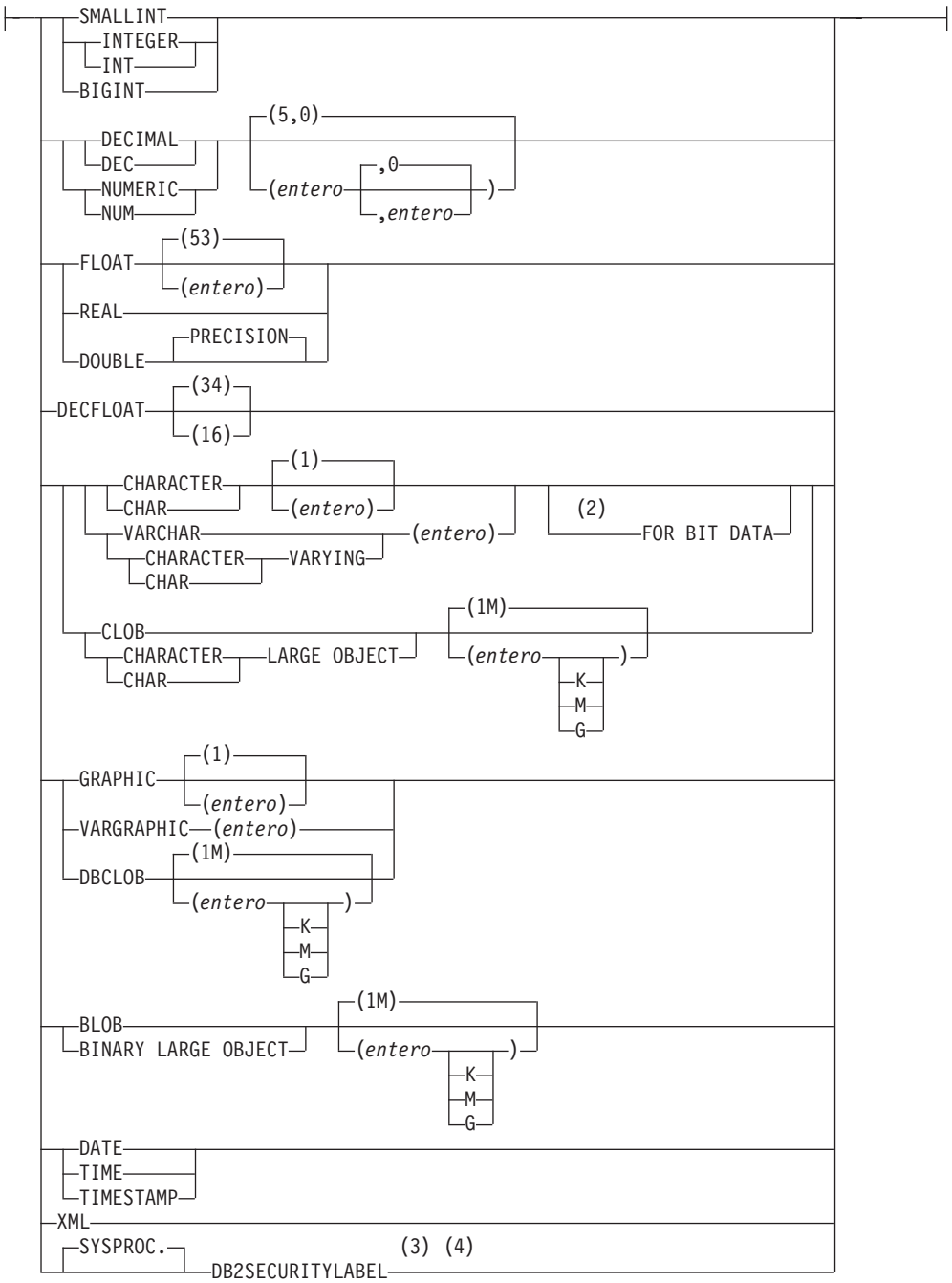


definición-columna:

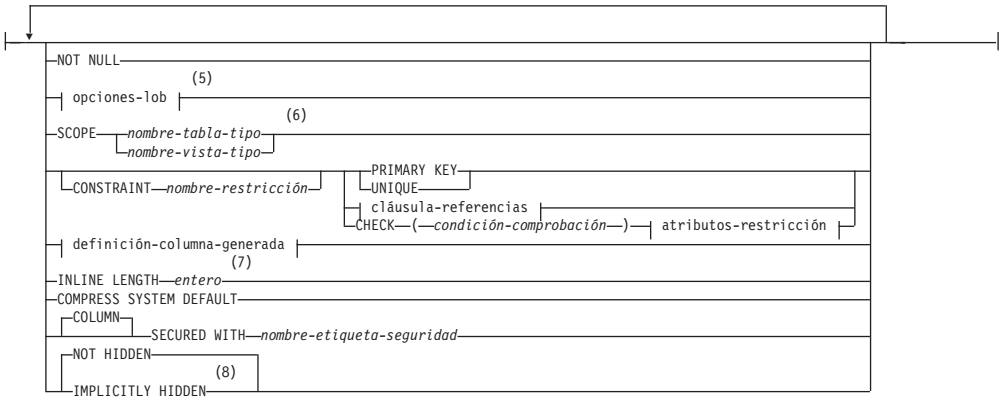


tipo-datos:

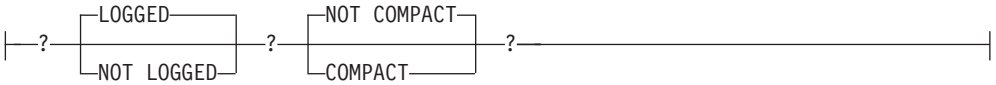
tipo-incorporado:



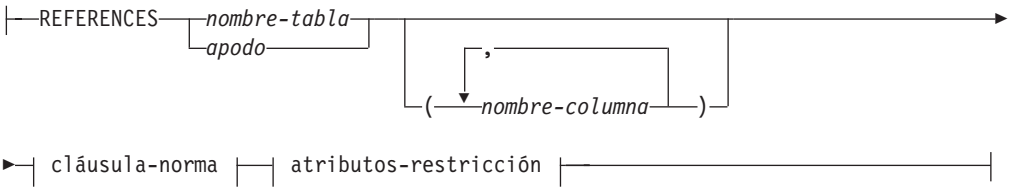
opciones-columna:



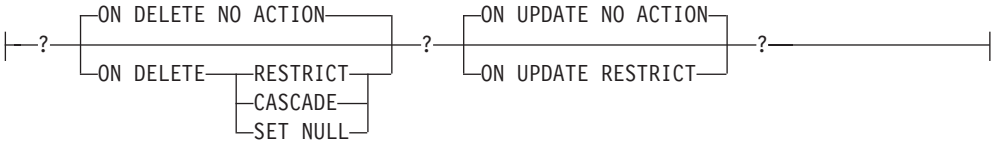
opciones-lob:



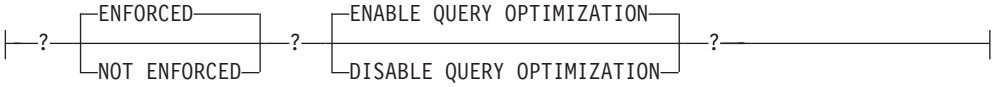
cláusula-referencias:



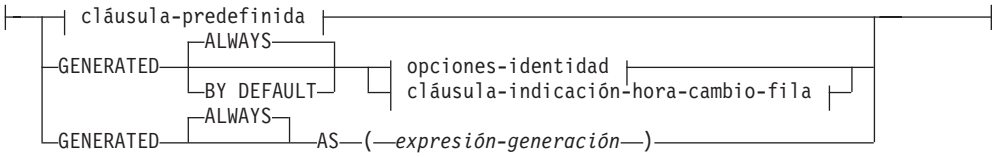
cláusula-norma:



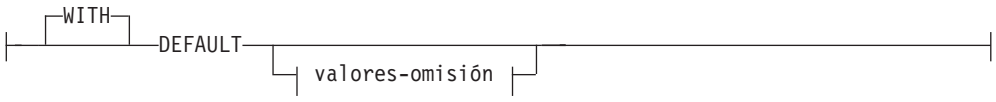
atributos-restricción:



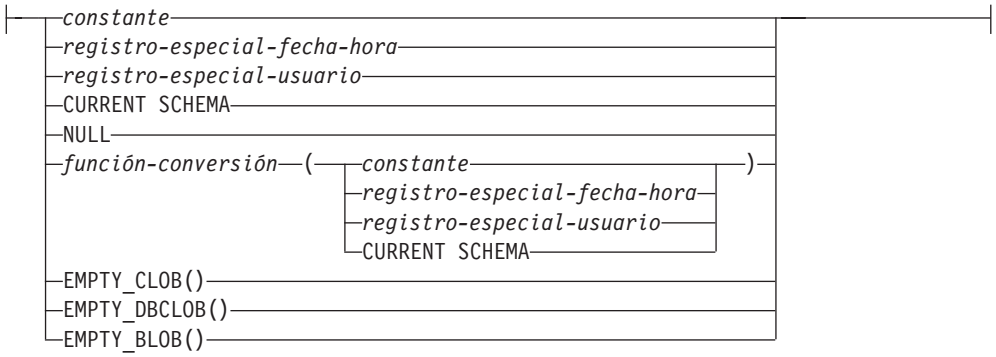
definición-columna-generada:



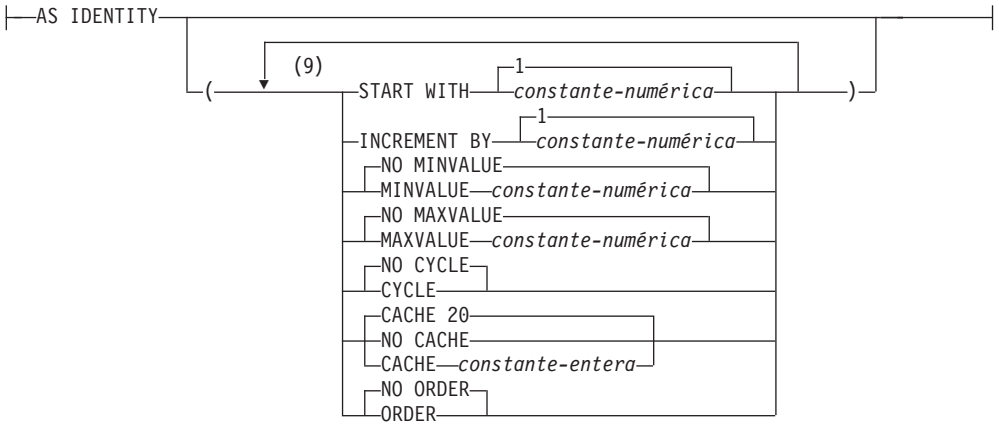
cláusula-predefinida:



valores-omisión:

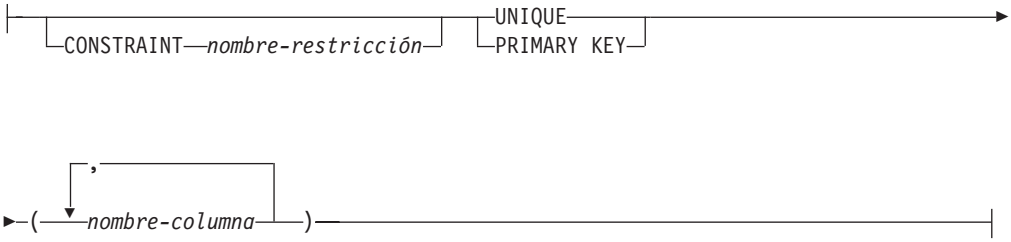


opciones-identidad:

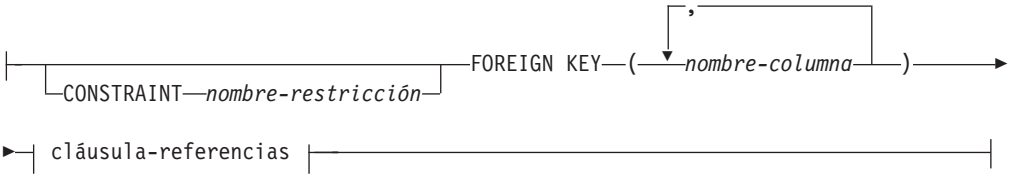


cláusula-indicación-hora-cambio-fila:

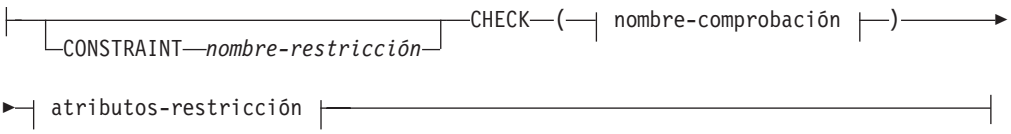
restricción-unicidad:



restricción-referencia:

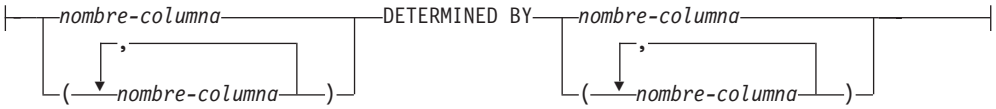


restricción-comprobación:

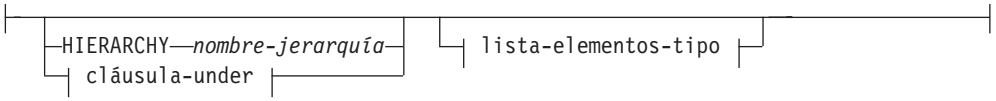


condición-error:

dependencia-funcional:

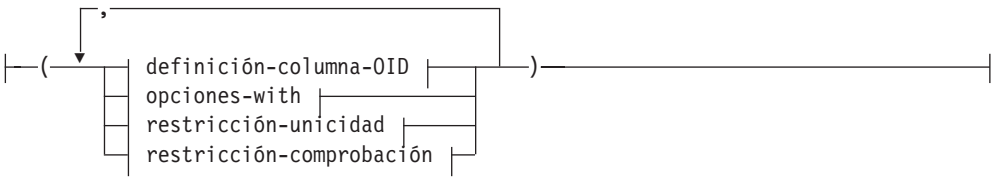


opciones-tabla-con-tipo:



cláusula-under:

lista-elementos-tipo:



definición-columna-OID:

opciones-with:

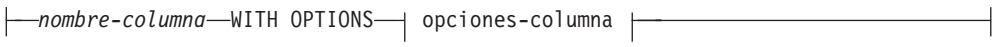
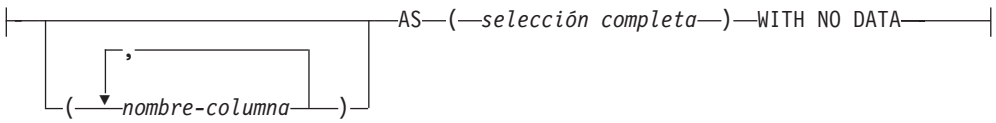
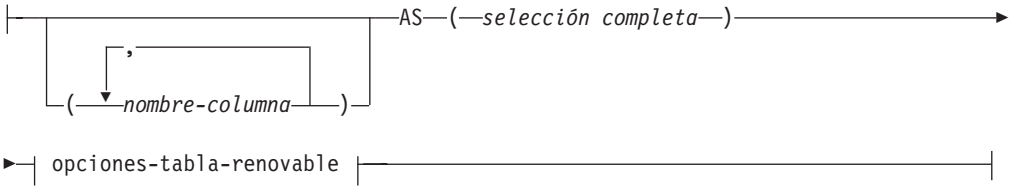


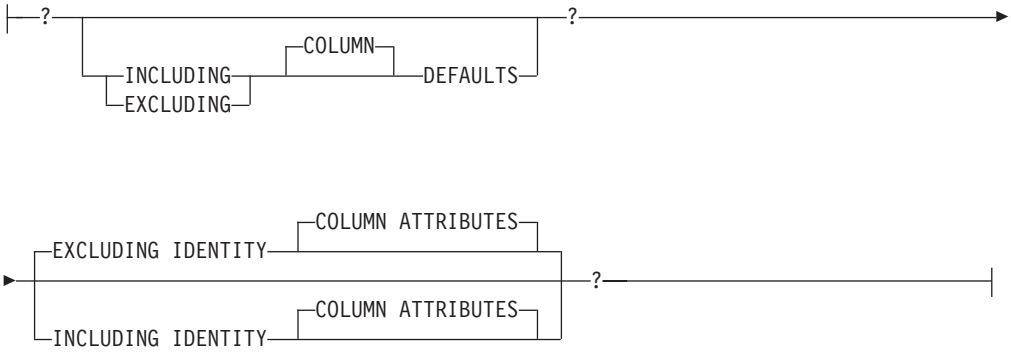
tabla-resultados-as:



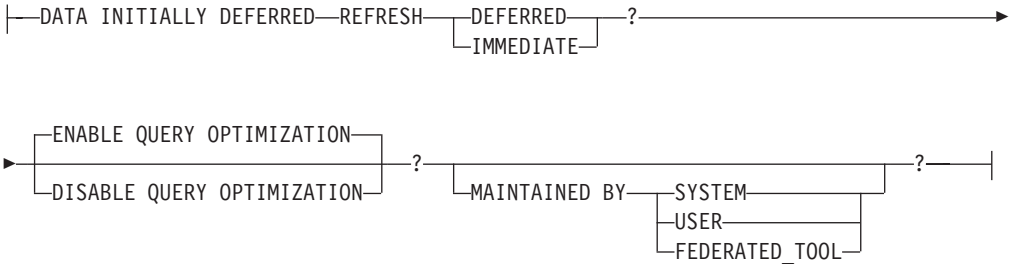
definición-consulta-materializada:



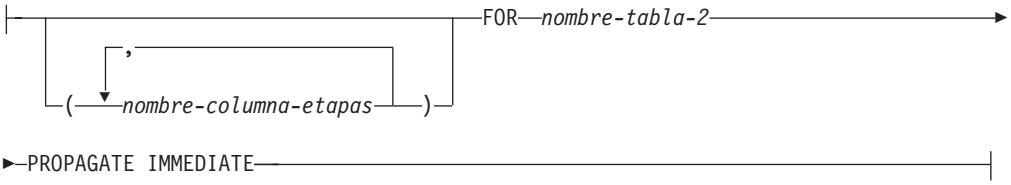
opciones-copia:



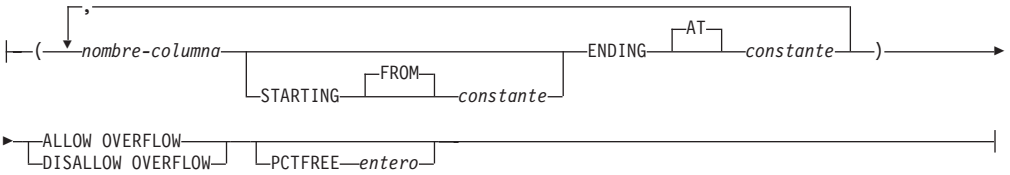
opciones-tabla-renovable:



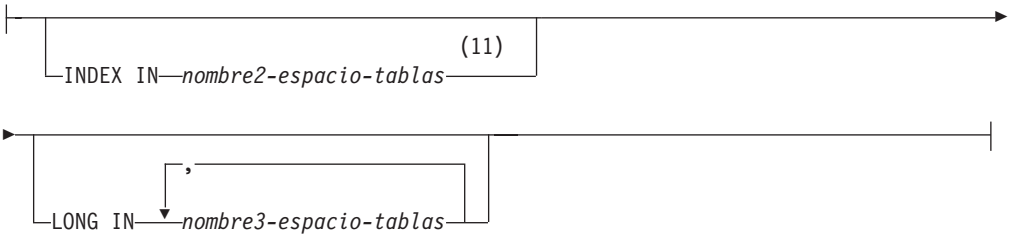
definición-tabla-etapas:



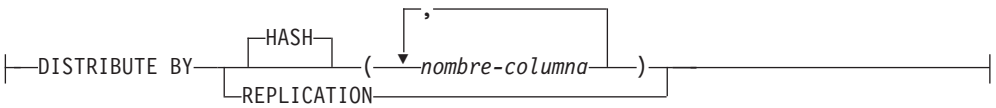
espec-clave-secuencia:



opciones-espaciotablas:

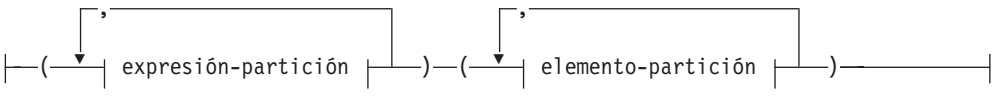


cláusula-distribución:



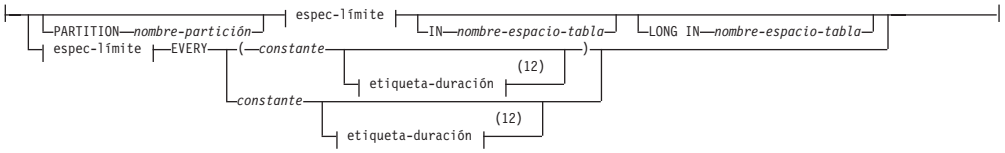
cláusula-particionamiento:

espec-partición-rango:



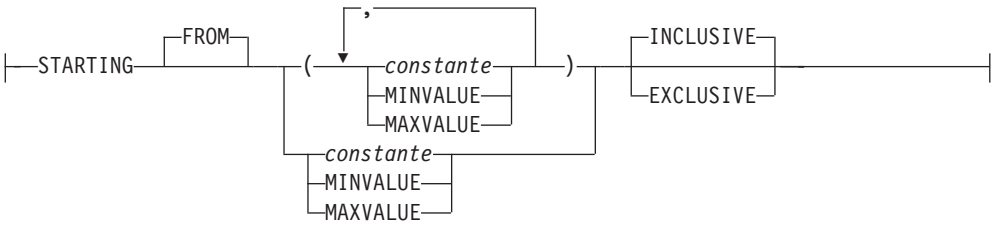
expresión-partición:

elemento-partición:

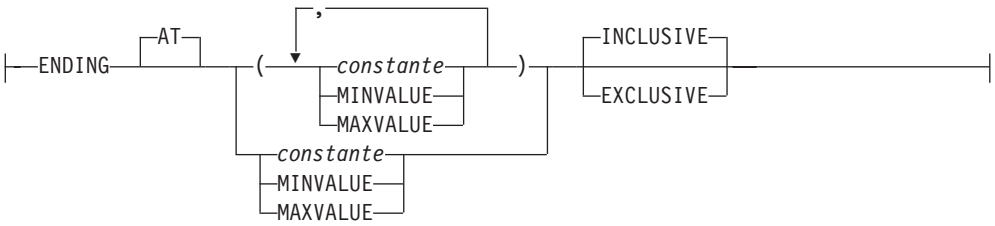


espec-límite:

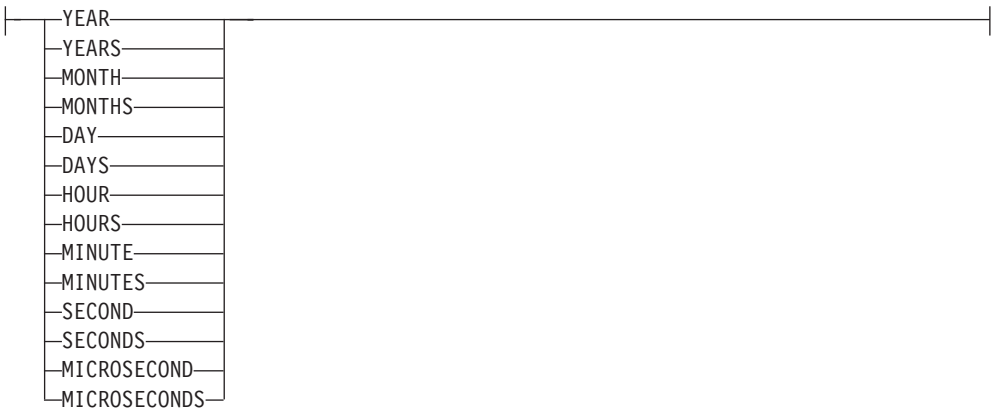
cláusula-inicial:



cláusula-final:



etiqueta-duración:



Notas:

1 Si la primera opción-columna elegida es una definición-columna-generada con una expresión-generación, el tipo-datos puede omitirse. Éste se determinará a partir del tipo de datos resultante de la expresión-generación.

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

3 DB2SECURITYLABEL es el tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida.

4 Para una columna de tipo DB2SECURITYLABEL, NOT NULL WITH DEFAULT está implícito y no se puede especificar explícitamente (SQLSTATE 42842). El valor por omisión de una columna de tipo DB2SECURITYLABEL es la etiqueta de seguridad del ID de autorización de sesión correspondiente al acceso de grabación.

5 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.

6 La cláusula SCOPE sólo se aplica al tipo REF.

7 INLINE LENGTH sólo es aplicable a columnas definidas como tipos estructurados. o XML.

8 IMPLICITLY HIDDEN sólo se puede especificar si también se especifica ROW CHANGE TIMESTAMP.

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

10 El tipo de dato es opcional para una columna de indicación de fecha y hora de cambio de fila.

11 Se puede especificar qué espacio de tablas contendrá los índices de una tabla al crearse la tabla. Si la tabla es una tabla particionada de rango, el espacio de tablas de índices se puede especificar con la cláusula IN de la sentencia CREATE INDEX.

12 Esta sintaxis de un elemento-partición es válida si hay una sola expresión-partición con un tipo de datos numérico o de fecha y hora.

13 El primer elemento-partición debe contener una cláusula-inicial y el último elemento-partición debe incluir una cláusula-final.

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).

lista-elementos
Define los elementos de una tabla. Esto incluye la definición de las columnas y las restricciones de la tabla.

definición-columna
Define los atributos de una columna.

nombre-columna
Es el nombre de una columna de la tabla. El nombre no puede estar calificado y no puede utilizarse el mismo nombre para más de una columna de la tabla (SQLSTATE 42711).

Una tabla puede tener las características siguientes:

- Un tamaño de página de 4K con un máximo de 500 columnas, donde el número total de bytes de las columnas no debe ser superior a 4.005.
- Un tamaño de página de 8K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 8.101.
- Un tamaño de página de 16K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 16.293.
- Un tamaño de página de 32K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 32.677.

Para obtener más detalles, consulte el apartado Tamaño de fila.

tipo-datos
Especifica el tipo de datos de la columna.

tipo-incorporado
Para los tipos incorporados, utilice uno de los tipos siguientes.

SMALLINT
Para especificar un entero pequeño.

INTEGER o INT
Para especificar un entero grande.

BIGINT
Para especificar un entero superior.

DECIMAL(*entero-precisión, entero-escala)* **o DEC**(*entero-precisión, entero-escala)*
Para especificar un número decimal. El primer entero es la precisión del número, es decir, el número total de dígitos; los valores permitidos son del 1 al 31. El segundo entero es la escala del número (es decir, el número de dígitos situados a la derecha de la coma decimal); su valor varía entre 0 y la precisión del número.

Si no se especifican la precisión ni la escala, se emplean los valores por omisión 5,0. Las palabras **NUMERIC** y **NUM** pueden utilizarse como sinónimos de **DECIMAL** y **DEC**.

FLOAT(*entero***)**
Para especificar un número de coma flotante de precisión simple o doble, en función del valor del *entero*. El valor del entero debe estar en el rango 1-53. Los valores 1 a 24 indican precisión simple y los valores 25 a 53 indican precisión doble.

También puede especificar:

REAL Para especificar un valor de coma flotante de precisión simple.

DOUBLE
Para especificar coma flotante de precisión doble.

DOUBLE PRECISION
Para especificar coma flotante de precisión doble.

FLOAT
Para especificar coma flotante de precisión doble.

DECFLOAT*(entero-precisión)*
Para especificar un número de coma flotante decimal. El valor de *entero-precisión* es la precisión del número; es decir, el número total de dígitos, los cuales pueden ser 16 ó 34.

Si no se especifica la precisión, se utilizará un valor por omisión de 34.

CHARACTER*(entero)* **o CHAR***(entero)* **o CHARACTER o CHAR**
Para una serie de caracteres de longitud fija de 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.

FOR BIT DATA
Especifica que el contenido de la columna se tratará como datos de bit (binarios). Durante el intercambio de datos con otros sistemas, no se efectúan conversiones de página de códigos. Las comparaciones se efectúan en binario, sin tener en cuenta el orden de clasificación de la base de datos.

CLOB o CHARACTER (CHAR) LARGE OBJECT*(entero [K | M | G])*
Para una serie de gran objeto de caracteres de la longitud máxima especificada en bytes.

El significado de *entero K | M | G* es el mismo que para BLOB.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 (1 megabyte).

No es posible especificar la cláusula FOR BIT DATA para las columnas CLOB. Sin embargo, una serie CHAR FOR BIT DATA puede asignarse a una columna CLOB y una serie CHAR FOR BIT DATA puede concatenarse con una serie CLOB.

GRAPHIC*(entero)*
Para una serie gráfica de longitud fija de longitud *entero*, que puede ir de 1 a 127. Si se omite la especificación de longitud, se supone una longitud de 1.

VARGRAPHIC*(entero)*
Para una serie gráfica de longitud variable de *entero* de longitud máxima, que puede estar en el rango de 1 a 16.336.

DBCLOB*(entero [K | M | G])*
Para una serie de objeto grande de caracteres de doble byte de la longitud máxima especificada en caracteres de doble byte.

El significado de *entero K | M | G* es parecido al de BLOB. Las diferencias radican en que el número especificado es el número de caracteres de doble byte y que el tamaño máximo es 1.073.741.823 caracteres de doble byte.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 caracteres de doble byte.

BLOB o BINARY LARGE OBJECT*(entero [K | M | G])*
Para una serie de objeto grande binario de la longitud máxima especificada en bytes.

La longitud puede estar en el rango de 1 byte a 2.147.483.647 bytes.

Si el *entero* ya está especificado por sí solo, se entiende que esa es la longitud máxima.

Si se especifica *entero K* (ya sea en mayúsculas o en minúsculas), la longitud máxima es *entero* multiplicado por 1.024. El valor máximo para *entero* es 2.097.152.

Si se especifica *entero M*, la longitud máxima es *entero* multiplicado por 1.048.576. El valor máximo para *entero* es 2.048.

Si se especifica *entero G*, la longitud máxima es *entero* multiplicado por 1.073.741.824. El valor máximo de *entero* es 2.

Si se especifica un múltiplo de K, M o G cuyo cálculo da 2.147.483.648, el valor real que se utiliza es 2.147.483.647 (o 2 gigabytes menos 1 byte), que es la longitud máxima para una columna LOB.

Si se omite la especificación de longitud, se supone que la longitud es 1.048.576 (1 megabyte).

Está permitido especificar cualquier número de espacios entre el entero y K, M o G; por otra parte, no es obligatorio especificar un espacio. Por ejemplo, todos los valores siguientes son válidos:

```
BLOB(50K)    BLOB(50 K)    BLOB (50   K)
```

DATE
Para la fecha.

TIME
Para la hora.

TIMESTAMP
Para la indicación horaria.

XML
Para un documento XML. Solo se pueden insertar documentos

XML con el formato correcto en una columna XML. Las columnas sólo pueden ser de tipo XML si la instancia de base de datos tiene una única partición de base de datos (SQLSTATE 42997).

Una columna XML tiene las restricciones siguientes:

- La columna no puede formar parte de ningún índice excepto un índice sobre datos XML. Por lo tanto, no puede incluirse como columna de una clave primaria ni de una restricción de unicidad (SQLSTATE 42962).
- La columna no puede ser una clave foránea de una restricción de referencia (SQLSTATE 42962).
- No se puede especificar un valor por omisión (WITH DEFAULT) para la columna (SQLSTATE 42613). Si la columna es anulable, el valor por omisión de la columna es el valor nulo.
- No se puede emplear la columna en una tabla con una clave de distribución (SQLSTATE 42997).
- No se puede emplear la columna en una tabla de clúster de rango (SQLSTATE 429BG).
- No se puede emplear la columna en una tabla particionada de rango (SQLSTATE 42997).
- No se puede hacer referencia a la columna en una restricción de comprobación excepto en un predicado VALIDATED (SQLSTATE 42621).

Cuando se crea una columna de tipo XML, se crea un índice de vías de acceso XML en esa columna. También se crea un índice de regiones XML en el nivel de tabla cuando se crea la primera columna de tipo XML. El nombre de estos índices es 'SQL' seguido de una indicación de fecha y hora de caracteres (*aammddhhmmssxxx*). El nombre de esquema es SYSIBM.

SYSPROC.DB2SECURITYLABEL
Es un tipo diferenciado incorporado que debe utilizarse para definir la columna de etiqueta de seguridad de fila de una tabla protegida. El tipo de datos subyacente de una columna del tipo diferenciado incorporado DB2SECURITYLABEL es VARCHAR(128) FOR BIT DATA. Una tabla puede tener como máximo una columna de tipo DB2SECURITYLABEL (SQLSTATE 428C1).

nombre-tipo-diferenciado
Para un tipo definido por el usuario que es un tipo diferenciado. Si se especifica un nombre de tipo diferenciado sin un nombre de esquema, el nombre del tipo diferenciado se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico).

Si se define una columna con un tipo diferenciado, el tipo de datos de la columna es el tipo diferenciado. La longitud y la escala de la columna son, respectivamente, la longitud y la escala del tipo de fuente del tipo diferenciado.

Si una columna definida con un tipo diferenciado es la clave externa de una restricción de referencia, el tipo de datos de la columna correspondiente de la clave primaria debe tener el mismo tipo diferenciado.

nombre-tipo-estructurado
Para especificar un tipo definido por el usuario que es un tipo estructurado. Si se especifica un nombre de tipo estructurado sin un nombre de esquema, el nombre de tipo estructurado se resuelve buscando en los esquemas especificados en la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el SQL estático y por el registro CURRENT PATH en el SQL dinámico).

Si se define una columna utilizando un tipo estructurado, el tipo de datos estáticos de la columna es el tipo estructurado. La columna puede contener valores con un tipo dinámico que es un subtipo de *nombre-tipo-estructurado*.

Una columna definida utilizando un tipo estructurado no se puede utilizar en una clave primaria, restricción de unicidad, clave foránea, clave de índice o clave de distribución (SQLSTATE 42962).

Si se define una columna utilizando un tipo estructurado, y la columna contiene un atributo de tipo de referencia, a cualquier nivel de anidamiento, ese atributo no tiene ámbito. Para utilizar un atributo de esa clase en una operación de eliminación de referencia, es necesario especificar explícitamente un ámbito (SCOPE), utilizando una especificación CAST.

REF (*nombre2-tipo***)**
Para una referencia a una tabla con tipo. Si se especifica *nombre-tipo2* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso de SQL estático y por el registro CURRENT PATH en el caso del SQL dinámico). El tipo de datos subyacente de la columna se basa en el tipo de datos de representación que se ha especificado en la cláusula REF USING de la sentencia CREATE TYPE para *nombre-tipo2* o el tipo raíz de la jerarquía de tipos de datos que incluye a *nombre-tipo2*.

opciones-columna
Define opciones adicionales relacionadas con las columnas de la tabla.

NOT NULL
Evita que la columna contenga valores nulos.

Si no se especifica NOT NULL, la columna puede contener valores nulos, y su valor por omisión es un valor nulo o bien el valor proporcionado por la cláusula WITH DEFAULT.

NOT HIDDEN o IMPLICITLY HIDDEN
Especifica si hay que definir la columna como oculta o no. El atributo oculto determina si la columna se incluye como referencia implícita a la tabla o si se le puede hacer referencia de manera explícita en sentencias de SQL. El valor por omisión es NOT HIDDEN.

NOT HIDDEN
Especifica que la columna se incluye como referencia implícita a la tabla y que se puede hacer referencia a la columna de manera explícita.

IMPLICITLY HIDDEN
Especifica que la columna no es visible en sentencias de SQL a menos que se haga referencia a la columna de manera explícita por el nombre. Por ejemplo, asumiendo que una tabla incluye una columna definida con la cláusula IMPLICITLY HIDDEN, el

resultado de SELECT * no incluye la columna implícitamente oculta. Sin embargo, el resultado de SELECT que haga referencia de manera explícita al nombre de una columna implícitamente oculta incluirá dicha columna en la tala de resultados.

IMPLICITLY HIDDEN sólo debe especificarse para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42867). La expresión ROW CHANGE TIMESTAMP FOR *designador-tabla* se resolverá en una columna IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP.

No debe especificarse IMPLICITLY HIDDEN para todas las columnas de la tabla (SQLSTATE 428GU).

opciones-lob
Especifica opciones para los tipos de datos LOB.

LOGGED
Especifica que los cambios efectuados en la columna deben registrarse en el archivo de anotaciones cronológicas. Acto seguido, los programas de utilidad de la base de datos (como RESTORE DATABASE) pueden recuperar los datos de estas columnas. LOGGED es el valor por omisión.

NOT LOGGED
Especifica que los cambios efectuados en la columna no se van a anotar cronológicamente.

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. Éste es el valor por omisión.

SCOPE
Identifica el ámbito de la columna de tipo de referencia.

Debe especificarse un ámbito para cualquier columna que vaya a utilizarse como operando izquierdo de un operador de eliminación de referencia o como argumento de la función DEREF. La especificación del ámbito de una columna de tipo de referencia puede diferirse a una

sentencia ALTER TABLE subsiguiente para permitir que se defina la tabla de destino, normalmente en el caso de tablas que se hacen referencia mutuamente.

nombre-tabla-tipo
El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores asignados a *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción. Un *nombre-restricción* no debe identificar a ninguna restricción que ya esté especificada en la misma sentencia CREATE TABLA (SQLSTATE 42710).

Si se omite esta cláusula, el sistema genera un identificador de 18 bytes que es exclusivo entre los identificadores de las restricciones existentes que se han definido en la tabla. (El identificador se compone de la palabra "SQL" seguida de una secuencia de 15 caracteres numéricos que genera una función basada en la indicación de la hora.)

Si se utiliza con una restricción PRIMARY KEY o UNIQUE, el *nombre-restricción* se puede utilizar como nombre de un índice creado para dar soporte a la restricción.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria compuesta de una sola columna. De este modo, si se especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si se especifica la cláusula PRIMARY KEY(C) como cláusula separada.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

No puede utilizarse una columna ROW CHANGE TIMESTAMP como parte de una clave primaria (SQLSTATE 429BV).

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

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

No puede especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

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

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

Consulte el apartado *cláusula-referencias* en *restricción-referencia*, que encontrará más adelante.

CHECK (*condición-comprobación***)**
Proporciona un método abreviado de definir una restricción de comprobación que se aplica a una sola columna. Vea CHECK (*condición-error*) más adelante.

definición-columna-generada
Especifica un valor generado para la columna.

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

WITH
Palabra clave opcional.

DEFAULT
Proporciona un valor por omisión en el caso de que no se suministre ningún valor en INSERT o se especifique uno como DEFAULT en INSERT o UPDATE. Si no se especifica un valor por omisión a continuación de la palabra clave DEFAULT, el valor por omisión depende del tipo de datos de la columna, tal como se muestra en "ALTER TABLE".

Si una columna se define como XML, no se puede especificar un valor por omisión (SQLSTATE 42613). El único valor por omisión posible es NULL.

Si la columna está basada en una columna de una tabla con tipo, debe especificarse un valor por omisión específico cuando se defina un valor por omisión. No se puede especificar un valor por omisión para la columna de identificador de objeto de una tabla con tipo (SQLSTATE 42997).

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

Si una columna se define utilizando un tipo estructurado, no puede especificarse la *cláusula-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
- no ser una constante de coma flotante a menos que la columna esté definida con un tipo de datos de coma flotante
- ser una constante numérica o un valor especial de coma flotante decimal si el tipo de datos de la columna es de coma flotante decimal. Las constantes de coma flotante se interpretan en primer lugar como DOUBLE y, a continuación, se convierten a coma flotante decimal, si la columna de destino es DECFLOAT. Para las columnas DECFLOAT(16), las constantes decimales que tengan una precisión de más de 16 dígitos se redondearán utilizando las modalidades de redondeo especificadas por el registro especial CURRENT DECFLOAT ROUNDING MODE.
- no tener dígitos 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 bytes, incluyendo los caracteres de comillas, cualquier carácter prefijo como la X para una constante hexadecimal, los caracteres del nombre de función completamente calificados y paréntesis, cuando la constante es el argumento de una *función-conversión*

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

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

CURRENT SCHEMA
Especifica el valor que tenía el registro especial CURRENT

SCHEMA en el momento de ejecutarse INSERT, UPDATE o LOAD como valor por omisión de la columna. Si se especifica CURRENT SCHEMA, el tipo de datos de la columna debe ser una serie de caracteres con una longitud superior o igual al atributo de longitud del registro especial CURRENT SCHEMA.

NULL
Especifica NULL como valor por omisión para la columna. Si se ha especificado NOT NULL, puede especificarse DEFAULT NULL en la misma definición de columna, pero se producirá un error si se intenta establecer la columna en el valor por omisión.

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

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

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

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

EMPTY_CLOB(), EMPTY_DBCLOB() o EMPTY_BLOB()
Especifica una serie de longitud cero como valor por omisión para la columna. La columna debe tener el tipo de datos que corresponden al tipo de datos de resultado de la función.

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

GENERATED
Indica que DB2 genera valores para la columna. Debe especificarse GENERATED si la columna va a considerarse una columna IDENTITY o una columna ROW CHANGE TIMESTAMP.

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 las columnas generadas.

BY DEFAULT
Especifica que DB2 generará un valor para la columna cuando se inserte una fila o se actualice mediante la especificación de la cláusula DEFAULT, a menos que se especifique un valor explícito. BY DEFAULT es el valor recomendado cuando se utiliza la propagación de datos o se realiza una operación de descarga o recarga.

Aunque no se solicita explícitamente, defina un índice de una sola columna exclusivo en las columnas IDENTITY generadas para garantizar la exclusividad de los valores.

AS IDENTITY
Especifica que la columna va a ser la columna de identidad para esta tabla. Una tabla 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 descendentes. Este valor no es necesariamente el valor del que se realizaría un ciclo después de alcanzar el valor máximo o mínimo para la columna de identidad. Se puede utilizar la cláusula START WITH para iniciar la generación de valores 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 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. Éste 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. 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 se pueda asignar a esta columna (SQLSTATE 42815), sin dígitos distintos de cero a la derecha de la coma decimal (SQLSTATE 428FA), pero el valor debe ser superior o igual al valor mínimo (SQLSTATE 42815).

NO CYCLE o CYCLE
Especifica si esta columna de identidad debe o no seguir generando valores tras la generación de su valor máximo o de su valor mínimo.

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

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

Cuando CYCLE está en vigor, DB2 puede generar valores duplicados para una columna de identidad. Aunque no se solicita explícitamente, debe definirse un índice de una sola columna, exclusivo, en la columna generada para garantizar la existencia de valores exclusivos, si se 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. Éste es el valor por omisión.

ORDER
Especifica que los valores deben generarse en el orden en el que se solicitan.

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
Especifica que la columna es una columna de indicación de fecha y hora para la tabla. Se genera un valor para la columna en cada fila que se inserta y para cualquier fila de cualquier columna que se actualiza. El valor que se genera para una columna ROW CHANGE TIMESTAMP es una indicación de fecha y hora que corresponde a la hora de inserción o actualización de la fila. Si se insertan o actualizan varias filas con una sentencia única, el valor de la columna ROW CHANGE TIMESTAMP puede ser diferente para cada fila.

Una tabla sólo puede tener una columna ROW CHANGE TIMESTAMP (SQLSTATE 428C1). Si se especifica *tipo-datos*, debe ser TIMESTAMP (SQLSTATE 42842). Una columna ROW CHANGE TIMESTAMP no puede tener una cláusula DEFAULT (SQLSTATE 42623). Debe especificarse NOT NULL para una columna ROW CHANGE TIMESTAMP (SQLSTATE 42831).

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 ninguno de los elementos siguientes (SQLSTATE 42621):

- Subconsultas
- Expresiones XMLQUERY o XMLEXISTS
- 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 que no estén definidas con NO SQL
- Variables del lenguaje principal o marcadores de parámetros
- Registros especiales
- Variables globales
- Referencias a columnas definidas más adelante en la lista de columnas
- Referencias a otras columnas generadas
- Referencias a columnas de tipo XML

El tipo de datos de la columna se basa en el tipo de datos resultante de la *expresión-generación*. Se puede utilizar una especificación CAST para forzar un tipo de datos determinado y proporcionar un ámbito (para un tipo de referencia solamente). Si se especifica *tipo-datos*, se asignarán valores a la columna de acuerdo con las normas de asignación correspondientes. Implícitamente se considera que una columna generada puede contener nulos, a menos que se especifique la opción NOT NULL

para columnas.El tipo de datos de una columna generada y el tipo de datos de resultado de *expresión-generación* debe tener definida la igualdad definida (consulte "Asignaciones y comparaciones"). Esto excluye las columnas y las expresiones de generación de tipos de datos LOB,XML, los tipos estructurados y los tipos diferenciados basados en cualquiera de estos tipos (SQLSTATE 42962).

INLINE LENGTH *entero*
Esta opción sólo es válida para una columna que se ha definido utilizando un tipo XML o estructurado (SQLSTATE 42842).

Para una columna de un tipo XML, *entero* indica el tamaño máximo en bytes de la representación interna de un documento XML que debe almacenarse en la fila de la tabla base. Los documentos de XML que tengan una representación interna más grande se almacenarán de modo independiente de la fila de tabla base en un objeto de almacenamiento auxiliar. Esta operación se realiza automáticamente. No hay longitud en línea por omisión para las columnas del tipo XML.

Para una columna de tipo estructurado, *entero* indica el tamaño máximo en bytes de una instancia de un tipo estructurado que debe almacenarse en línea con el resto de los valores de la fila. Las instancias de tipos estructurados que no se pueden almacenar en 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 longitud en línea de la columna.

Nota: Las longitudes en línea de los subtipos no se cuentan en la longitud en línea por omisión, lo que significa que las instancias de los subtipos pueden no caber en línea a menos que, al crear la tabla, se especifique explícitamente un valor INLINE LENGTH para tener en cuenta los subtipos actuales y futuros.

El valor INLINE LENGTH explícito debe ser igual a 292 como mínimo y no ser mayor que 32673 (SQLSTATE 54010).

COMPRESS SYSTEM DEFAULT
Especifica que los valores por omisión del sistema se deben almacenar utilizando el mínimo espacio. Si no se especifica la cláusula VALUE COMPRESSION, se devuelve un aviso (SQLSTATE 01648) y los valores por omisión del sistema no se almacenan utilizando el mínimo espacio.

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

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

COLUMN SECURED WITH *nombre-etiqueta-seguridad*
Identifica una etiqueta de seguridad existente para la política de

seguridad asociada a la tabla. El nombre no debe ser calificado (SQLSTATE 42601). La tabla debe tener asociada una política de seguridad (SQLSTATE 55064).

restricción-unicidad
Define una restricción de clave primaria o de unicidad. Si la tabla tiene una clave de distribución, cualquier clave primaria o de unicidad debe ser un superconjunto de la clave de distribución. No puede especificarse una restricción de unicidad o de clave primaria para una tabla que sea una subtabla (SQLSTATE 429B3). Las claves primarias o exclusivas no pueden ser subconjuntos de dimensiones (SQLSTATE 429BE). Si la tabla es una tabla raíz, la restricción se aplica a la tabla y a todas sus subtablas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de clave primaria o de unicidad.

UNIQUE (*nombre-columna***,...)**
Define una clave exclusiva compuesta por las columnas identificadas. Las columnas identificadas deben estar definidas como NOT NULL. Cada *nombre-columna* debe identificar una columna de la tabla y la misma columna no debe estar identificada más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar como parte de una clave exclusiva ningún LOB, XML, tipo diferenciado basado en uno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave exclusiva no puede ser el mismo que el conjunto de columnas de la clave primaria u otra clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

No puede especificarse una restricción de unicidad si la tabla es una subtabla (SQLSTATE 429B3) porque las restricciones de unicidad se heredan de la supertabla.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave de unicidad y su índice de unicidad. Se creará automáticamente un índice bidireccional exclusivo, que permite las exploraciones directas e inversas, para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre de índice entra en conflicto, el nombre será SQL seguido de una indicación de fecha y hora de caracteres (*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 de la tabla, y la misma columna no debe identificarse más de una vez.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar como parte de una clave primaria ningún LOB, XML,tipo diferenciado basado en uno de estos tipos o tipo estructurado, aunque el atributo de longitud de la columna sea suficientemente pequeño para caber en el límite de longitud de la clave de índice del tamaño de página (SQLSTATE 54008).

El conjunto de columnas de la clave primaria no puede ser el mismo que el conjunto de columnas de una clave exclusiva (SQLSTATE 01543). (Si LANGLEVEL es SQL92E o MIA, se devuelve un error, SQLSTATE 42891.)

En una tabla sólo se puede definir una clave primaria.

No puede especificarse una clave primaria si la tabla es una subtabla (SQLSTATE 429B3) porque la clave primaria se hereda de la supertabla.

La descripción de la tabla, tal como está registrada en el catálogo, incluye la clave primaria y su índice principal. Se creará automáticamente un índice bidireccional exclusivo, que permite las exploraciones directas e inversas, para las columnas en la secuencia especificada por orden ascendente para cada columna. El nombre del índice será el mismo que el *nombre-restricción* si no entra en conflicto con un índice existente en el esquema donde se ha creado la tabla. Si el nombre de índice entra en conflicto, el nombre será SQL seguido de una indicación de fecha y hora de caracteres (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema.

Si la tabla tiene una clave de distribución, las columnas de una *restricción-unicidad* deben ser un superconjunto de las columnas de clave de distribución; el orden de las columnas no es importante.

restricción-referencia
Define una restricción de referencia.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de referencia.

FOREIGN KEY (*nombre-columna***,...)**
Define una restricción de referencia con el *nombre-restricción* especificado.

T1 indicará la tabla de objetos de la sentencia. La clave foránea de la restricción de referencia se compone de las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de T1, y no debe identificarse la misma columna en más de una ocasión.

El número de columnas identificadas no debe ser superior a 64 y la suma de sus longitudes almacenadas no debe ser superior al límite de la longitud de la clave de índice para el tamaño de página. Para las longitudes almacenadas de columnas, consulte el apartado Número total de bytes. Para los límites de longitud de clave, consulte "Límites de SQL". No se puede utilizar una columna LOB, XML, 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 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 la cláusula es una especificación implícita de ON DELETE NO ACTION.

Si T1 está conectada para supresión con T2 mediante varias vías de acceso, no se permitirá definir dos normas SET NULL con definiciones de claves foráneas solapadas. Por ejemplo: T1 (i1, i2, i3). No se permite utilizar la Norma1 con la clave foránea (i1, i2) y la Norma2 con la clave foránea (i2, i3).

El orden de aplicación de las normas es el siguiente:

1. RESTRICT
2. SET NULL OR CASCADE
3. NO ACTION

Si dos normas distintas afectan a cualquier fila de T1, se producirá un error y no se suprimirá ninguna fila.

No se puede definir una restricción de referencia si hará que una tabla se suprima-conecte a sí misma mediante un ciclo en el que intervienen dos o más tablas y donde una de las normas de supresión es RESTRICT o SET NULL (SQLSTATE 42915).

Se puede definir una restricción de referencia que haga que una tabla se suprima-conecte a sí misma o a otra tabla mediante varias vías de acceso, excepto en los siguientes casos (SQLSTATE 42915):

- Una tabla no debe ser una tabla dependiente en una relación de tipo CASCADE (referencia a sí misma o referencia a otra tabla) y no debe tener una relación de auto referencia en la que la norma de supresión es RESTRICT o SET NULL.
- Una clave solapa otra clave cuando al menos una columna de una clave está en la misma columna que la otra clave. Cuando una tabla se suprime-conecta a otra tabla a través de varias relaciones con claves foráneas de solapamiento, estas relaciones deben tener la misma norma de supresión y ninguna de las normas de supresión puede ser SET NULL.
- Cuando una tabla está conectada por supresión a otra tabla a través de varias relaciones, y al menos una de estas relaciones se especifica con la norma de supresión SET NULL, las definiciones de clave foránea de estas relaciones no deben contener ninguna clave de distribución ni columna de clave MDC (clúster multidimensional).
- Cuando dos tablas están conectadas por supresión con la misma tabla a través de relaciones de tipo CASCADE, las dos tablas no

deben estar conectadas por supresión entre sí si la norma de supresión de la última relación de cada una de las vías de acceso conectadas por supresión es RESTRICT o SET NULL.

Si alguna fila de T1 se ve afectada por dos normas de supresión distintas, el resultado sería el efecto de todas las acciones especificadas por estas normas. Los activadores AFTER y las restricciones CHECK sobre T1 también sufrirán el efecto de todas las acciones. Un ejemplo de esto es una fila que constituye el destino de anulación a través de una vía de acceso de supresión-conexión con una tabla progenitora y que es el destino de una supresión por parte de una segunda vía de acceso de supresión-conexión con la misma tabla progenitora. El resultado sería la supresión de la fila. Los activadores AFTER DELETE sobre esta tabla descendiente se activaría, pero los activadores AFTER UPDATE no.

Cuando se aplican las normas anteriores a las restricciones de referencia, en que la tabla padre o la tabla dependiente es un miembro de una jerarquía de tablas con tipo, se tienen en cuenta todas las restricciones de referencia aplicables a cualquier tabla de sus respectivas jerarquías.

ON UPDATE
Especifica la acción que se debe emprender en las tablas dependientes al actualizar una fila de la tabla padre. La cláusula es opcional. ON UPDATE NO ACTION es el valor por omisión y ON UPDATE RESTRICT es la única alternativa.

La diferencia entre NO ACTION y RESTRICT se describe en el apartado "Notas".

restricción-comprobación
Define una restricción de comprobación. Una *restricción-comprobación* es una *condición-búsqueda* que se debe evaluar como no falsa o una dependencia funcional que se define entre columnas.

CONSTRAINT *nombre-restricción*
Indica el nombre de la restricción de comprobación.

CHECK (*condición-comprobación***)**
Define una restricción de comprobación. La *condición-búsqueda* debe ser verdadera o desconocida para cada fila de la tabla.

condición-búsqueda
La *condición-búsqueda* tiene las restricciones siguientes:

- Una referencia de columna debe hacer referencia a una columna de la tabla que está creándose.
- La *condición-búsqueda* no puede contener un predicado TYPE.
- La *condición-búsqueda* no puede contener ninguno de los elementos siguientes (SQLSTATE 42621):
  - Subconsultas
  - Expresiones XMLQUERY o XMLEXISTS
  - Operaciones de eliminación de referencia o funciones DEREF donde el argumento de referencia con ámbito no es el correspondiente a la columna de identificador de objeto (OID)
  - Especificaciones CAST con una cláusula SCOPE

- Funciones de columna
- Funciones que no sean deterministas
- Funciones definidas para que exista una acción externa
- Funciones definidas por el usuario definidas con CONTAINS SQL o READS SQL DATA
- Variables del lenguaje principal
- Marcadores de parámetro
- Registros especiales
- Variables globales
- Referencias a columnas generadas que no correspondan a la columna de identidad
- Referencias a columnas de tipo XML (excepto en un predicado VALIDATED)
- Una *expresión-tabla-anidada* tolerante a errores

dependencia-funcional
Define una dependencia funcional entre columnas.

nombre-columna **DETERMINED BY** *nombre-columna* **o**
(*nombre-columna***,...) DETERMINED BY (***nombre-columna***,...)**
El conjunto de columnas padre contiene las columnas identificadas que preceden inmediatamente a la cláusula DETERMINED BY. El conjunto de columnas hijo contiene las columnas identificadas que siguen inmediatamente a la cláusula DETERMINED BY. Todas las restricciones de la *condición-búsqueda* se aplican a las columnas de los conjuntos padre e hijo y sólo están permitidas las referencias de columnas simples en el conjunto de columnas (SQLSTATE 42621). La misma columna no se debe identificar más de una vez en la dependencia funcional (SQLSTATE 42709). El tipo de datos de la columna no puede ser un tipo de datos LOB, un tipo diferenciado basado en un tipo de datos LOB, un tipo de datos XML ni un tipo estructurado (SQLSTATE 42962). No puede utilizarse una columna ROW CHANGE TIMESTAMP como parte de una clave primaria (SQLSTATE 429BV). Ninguna columna del conjunto de columnas hijo puede ser una columna 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.

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
La restricción se supone que es verdadera y se puede utilizar para la optimización de consulta.

DISABLE QUERY OPTIMIZATION
No se puede utilizar la restricción para la optimización de consulta.

OF *nombre1-tipo*
Especifica que las columnas de la tabla están basadas en los atributos del tipo estructurado identificado por *nombre-tipo1*. Si se especifica *nombre-tipo1* sin un nombre de esquema, el nombre de tipo se resuelve efectuando una búsqueda en los esquemas de la vía de acceso de SQL (definida por la opción de preproceso FUNCPATH en el caso del SQL estático y por el registro CURRENT PATH en el caso de SQL dinámico). El nombre de tipo debe ser el nombre de un tipo existente definido por el usuario (SQLSTATE 42704) y debe ser un tipo estructurado del que se pueda crear una instancia (SQLSTATE 428DP) con al menos un atributo (SQLSTATE 42997).

Si no se especifica UNDER, debe especificarse una columna de identificador de objeto (consulte *definición-columna-OID*). Esta columna de identificador de objeto es la primera columna de la tabla. La columna de ID de objeto va seguida de columnas basadas en los atributos de *nombre-tipo1*.

HIERARCHY *nombre-jerarquía*
Indica la tabla de jerarquía asociada con la jerarquía de 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. Este nombre está compuesto por el nombre de la tabla que se crea, seguido de un sufijo exclusivo, de tal modo que el identificador será exclusivo entre los identificadores de las tablas, vistas y apodos existentes.

UNDER *nombre-supertabla*
Indica que la tabla es una subtabla de *nombre-supertabla*. La supertabla debe ser una tabla existente (SQLSTATE 42704) y la tabla debe estar definida mediante un tipo estructurado que sea el supertipo inmediato de *nombre-tipo1* (SQLSTATE 428DB). El nombre de esquema de *nombre-tabla* y *nombre-supertabla* debe ser el mismo (SQLSTATE 428DQ). La tabla identificada por *nombre-supertabla* no debe tener ninguna subtabla existente ya definida mediante *nombre-tipo1* (SQLSTATE 42742).

Las columnas de la tabla incluyen la columna de identificador de objeto de la supertabla con su tipo modificado para que sea REF(*nombre-tipo1*), seguida de columnas basadas en los atributos de *nombre-tipo1* (recuerde que el tipo incluye los atributos de su supertipo). Los atributos no pueden tener el mismo nombre que la columna de OID (SQLSTATE 42711).

No pueden especificarse otras opciones de tabla como el espacio de tablas, la captura de datos, NOT LOGGED INITIALLY y la clave de distribución. Estas opciones se heredan de la supertabla (SQLSTATE 42613).

INHERIT SELECT PRIVILEGES
Cualquier usuario o grupo que sostenga un privilegio SELECT sobre la supertabla recibirá un privilegio equivalente sobre la subtabla recién creada. Se considera que el definidor de la subtabla es el encargado de otorgar este privilegio.

lista-elementos-tipo
Define los elementos adicionales de una tabla con tipo. Esto incluye las opciones adicionales para las columnas, la adición de una columna de identificador de objeto (sólo la tabla raíz) y las restricciones de la tabla.

definición-columna-OID
Define la columna de identificador de objeto para la tabla con tipo.

REF IS *nombre-columna-OID* **USER GENERATED**
Especifica que en la tabla se define una columna de identificador de objeto (OID) como la primera columna. Se necesita un OID para la tabla raíz de una jerarquía de tablas (SQLSTATE 428DX). La tabla debe ser una tabla con tipo (debe estar presente la cláusula OF) que no sea una subtabla (SQLSTATE 42613). El nombre de la columna se define como *nombre-columna-OID* y no puede ser el mismo que el nombre de cualquier atributo del tipo estructurado *nombre-tipo1* (SQLSTATE 42711). La columna se define con el tipo REF(*nombre-tipo1*), NOT NULL y se genera un índice de unicidad requerido por el sistema (con un nombre de índice por omisión). Esta columna viene referida como la *columna de identificador de objeto* o *columna de OID*. Las palabras clave USER GENERATED indican que el usuario debe proporcionar el valor

inicial de la columna de OID cuando inserte una fila. Una vez que se haya insertado una fila, la columna de OID no podrá actualizarse (SQLSTATE 42808).

opciones-with
Define opciones adicionales que se aplican a las columnas de una tabla con tipo.

nombre-columna
Especifica el nombre de la columna para la que se especifican las opciones adicionales. El *nombre-columna* debe corresponder al nombre de una columna de la tabla que no sea además una columna de una supertabla (SQLSTATE 428DJ). Sólo puede aparecer un nombre de columna en una cláusula WITH OPTIONS de la sentencia (SQLSTATE 42613).

Si ya está especificada una opción como parte de la definición de tipo (en CREATE TYPE), las opciones especificadas aquí alteran temporalmente las opciones de CREATE TYPE.

WITH OPTIONS *opciones-columna*
Define opciones para la columna especificada. Consulte el valor *opciones-columna* descrito anteriormente. Si la tabla es una subtabla, no pueden especificarse restricciones de unicidad ni de clave primaria (SQLSTATE 429B3).

LIKE *nombre1-tabla* **o** *nombre-vista* **o** *apodo*
Especifica que las columnas de la tabla tienen exactamente el mismo nombre y descripción que las columnas de la tabla (*nombre-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 identificada (incluidas las columnas implícitamente ocultas), vista o apodo. También se definirá como implícitamente oculta una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla existente.

- Si se identifica una tabla, la definición implícita incluye el nombre de columna, tipo de datos, atributo oculto y la característica de poder contener nulos de cada una de las columnas de *nombre1-tabla*. Si no se especifica EXCLUDING COLUMN DEFAULTS, también se incluye el valor por omisión de la columna.
- Si se designa una vista, la definición implícita incluye el nombre de columna, tipo de datos y posibilidad de contener nulos de cada columna 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*.
- Si se identifica una tabla protegida en la cláusula LIKE, la tabla nueva hereda la misma política de seguridad y las mismas columnas protegidas que la tabla identificada.

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.

Cuando una tabla se identifica en la cláusula LIKE y dicha tabla contiene una columna ROW CHANGE TIMESTAMP, la columna correspondiente de la nueva tabla hereda sólo el tipo de datos de la columna ROW CHANGE TIMESTAMP. La nueva columna no se considera una columna generada.La tabla identificada en la cláusula LIKE no debe contener una columna ROW CHANGE TIMESTAMP que también esté definida como IMPLICITLY HIDDEN (SQLSTATE 42867).

opciones-copia
Estas opciones especifican si deben copiarse atributos adicionales de la definición de la tabla de resultados fuente (tabla, vista o selección completa).

INCLUDING COLUMN DEFAULTS
Especifica que deben copiarse los valores por omisión de cada columna actualizable contenida en la definición de la tabla de resultados fuente. Las columnas que no son actualizables no tienen un valor por omisión definido en la correspondiente columna de la tabla creada.

Si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica una tabla base o una tabla temporal declarada, INCLUDING COLUMN DEFAULTS es el valor por omisión. Si se especifica LIKE *nombre-tabla* y *nombre-tabla* identifica un apodo, INCLUDING COLUMN DEFAULTS no tendrá ningún efecto y no se copiarán los valores por omisión de la columna.

EXCLUDING COLUMN DEFAULTS
Especifica que no deben copiarse los valores por omisión de las columnas de la tabla de resultados fuente.

Esta es la cláusula por omisión, excepto si se especifica LIKE *nombre-tabla* y *nombre-tabla* 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.

tabla-resultados-as

nombre-columna
Designa las columnas de la tabla. Si se especifica una lista de nombres de columna, ésta debe constar de tantos nombres como columnas haya en la tabla de resultados de la *selección completa*. Cada *nombre-columna* debe ser exclusivo y no calificado. Si no se especifica una lista de nombres de columnas, las columnas de la tabla heredan los nombres de las columnas de la tabla de resultados de la *selección completa*.

Debe especificarse una lista de nombres de columna si la tabla de resultados de la selección completa tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es una columna derivada de una constante, función, expresión u operación de conjuntos que no se designa utilizando la cláusula AS de la lista de selección.

AS
Introduce la consulta que se utiliza para la definición de la tabla.

selección completa
Define la consulta en la que se basa la tabla. Las definiciones de columna resultantes son las mismas que las de una vista definida con la misma consulta. Una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla base a la que se hace referencia en la *selección completa* no se considera oculta en la nueva tabla.

Todos los elementos de la lista de selección deben tener un nombre (utilice la cláusula AS para las expresiones). La *tabla-resultados-as* define atributos de la tabla.

La *selección completa* no puede incluir una cláusula *referencia-tabla-cambio-datos* (SQLSTATE 428FL).

Se puede especificar cualquier *selección completa* válida que no haga referencia a una tabla con tipo ni a una vista con tipo.

WITH NO DATA
La consulta sólo se utiliza para definir la tabla. La tabla no se rellena con los resultados de la consulta.

Las columnas de la tabla se definen basándose en las definiciones de las columnas producidas por la *selección completa*. Si la *selección completa* hace referencia a una tabla individual de la cláusula FROM, los elementos de la lista de selección que son columnas de esa tabla se definen utilizando el nombre de columna, el tipo de datos y la posibilidad de contener nulos de la tabla referenciada.

definición-consulta-materializada

nombre-columna
Designa las columnas de la tabla. Si se especifica una lista de nombres de columna, 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.

Se debe especificar una lista de nombres de columna si la tabla de resultados de la *selección completa* tiene nombres de columna duplicados o una columna sin nombre (SQLSTATE 42908). Una columna sin nombre es

una columna derivada de una constante, función, expresión u operación de conjuntos que no se designa utilizando la cláusula AS de la lista de selección.

AS
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. No se considera oculta en la tabla nueva una columna de la tabla nueva que se corresponde a una columna implícitamente oculta de la tabla base a la que se hace referencia en la selección completa.

Todos los elementos de la lista de selección deben tener un nombre (utilice la cláusula AS para las expresiones). La *definición-consulta-materializada* define los atributos de la tabla de consultas materializadas. La opción elegida también define el contenido de la selección completa de la manera siguiente.

La selección completa no puede incluir una cláusula *referencia-tabla-cambio-datos* (SQLSTATE 428FL).

Cuando se especifica REFRESH DEFERRED o REFRESH IMMEDIATE, la selección completa no puede incluir (SQLSTATE 428EC):

- 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 diferenciado basado en este tipo)
- Funciones que tengan cualquiera de los atributos siguientes:
  - EXTERNAL ACTION
  - LANGUAGE SQL
  - CONTAINS SQL
  - READS SQL DATA
  - MODIFIES SQL DATA
- Funciones que dependan de características físicas (por ejemplo, DBPARTITIONNUM, HASHEDVALUE, RID_BIT, RID)
- Una expresión ROW CHANGE o referencia a una columna ROW CHANGE TIMESTAMP de la fila
- Referencias de tabla o vista a objetos del sistema (tampoco se deben especificar tablas Explain)
- Expresiones que sean un tipo estructurado, un tipo LOB (o un tipo diferenciado basado en un tipo LOB) o un tipo XML
- Referencias a una tabla protegida o un apodo protegido

Cuando se especifica DISTRIBUTE BY REPLICATION, se aplican las siguientes restricciones:

- No está permitida la cláusula GROUP BY.
- La tabla de 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
  - Variables globales
  - SELECT DISTINCT
  - Una *expresión-tabla-anidada* tolerante a errores
- Si la cláusula FROM hace referencia a más de una tabla o vista, sólo puede definir una unión interna sin utilizar la sintaxis INNER JOIN explícita.
- Cuando se especifica una cláusula GROUP BY, se aplican las siguientes consideraciones:
  - 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 distribución 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 las siguientes restricciones relacionadas con la exclusividad:
    - Todos los elementos de GROUP BY se deben incluir en la lista de selección.
    - Cuando GROUP BY contiene GROUPING SETS, CUBE o ROLLUP, los elementos de GROUP BY y las funciones de columna GROUPING de la lista de selección deben formar una clave exclusiva del conjunto de resultados. Por lo tanto, deberán satisfacerse las restricciones siguientes:
      - No se pueden repetir conjuntos de agrupación. Por ejemplo, `ROLLUP(X,Y),X` no está permitido, porque es equivalente a `GROUPING SETS((X,Y),(X),(X))`.
      - Si X es un elemento 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 REFRESH DEFERRED:

- Si 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.
- Si la consulta es recursiva, la tabla de consultas materializadas no se utiliza para optimizar el proceso de las consultas.

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.

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.

Cuando se especifica esta opción, la cláusula select de la sentencia CREATE TABLE no puede contener una referencia a una tabla base (SQLSTATE 428EC).

definición-tabla-etapas
Define la consulta que recibe el soporte de la tabla de etapas indirectamente por medio de una tabla de 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 19. 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 *nombre2-tabla*
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 incoherente, su contenido, en cualquier momento, son los cambios delta para la tabla subyacente desde la última renovación de la tabla de consultas materializadas.

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. La tabla cuya definición especifica esta cláusula se conoce como tabla de clúster multidimensional (MDC).

Se mantiene automáticamente un índice de bloques de clúster para cada dimensión especificada y se mantiene un índice de bloques, compuesto de

todas las columnas que se utilizan en la cláusula, si ninguno de los índices de bloques de clúster las incluyen. El conjunto de columnas que se utiliza en la cláusula ORGANIZE BY debe seguir las normas de la sentencia CREATE INDEX.

Cada nombre de columna especificado en la cláusula ORGANIZE BY debe estar definido para la tabla (SQLSTATE 42703). Una dimensión no puede aparecer más de una vez en la lista de dimensiones (SQLSTATE 42709). La dimensión no puede contener una columna ROW CHANGE TIMESTAMP (SQLSTATE 429BV). La tabla no puede contener ninguna columna con el tipo de datos XML (SQLSTATE 42601).

Las páginas de la tabla se organizan en bloques de igual tamaño, que es el tamaño de extensión del espacio de tablas, y todas las filas de cada bloque contienen la misma combinación de valores de dimensión.

Una tabla puede ser una tabla de clúster multidimensional (MDC) y una tabla particionada. Las columnas de una tabla de esta clase se pueden emplear tanto en la *espec-partición-rango* como en la clave MDC. Tenga presente que el particionamiento de tablas es multicolumna, no multidimensional.

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 los apartados Tamaño de fila y Número total de bytes).

nombre-columna

Especifica una columna de la tabla que se incluye en la clave exclusiva que determina la secuencia de la tabla 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 ser superior a 64 (SQLSTATE 54008).

Se creará una entrada de índice exclusiva automáticamente en el catálogo para las columnas de la secuencia de clave especificadas con orden ascendente para cada columna. El nombre del índice será SQL, seguido de una indicación de fecha y hora de caracteres (*aammddhhmmssxxx*), con SYSIBM como nombre de esquema. Un objeto de índice real no se crea en almacenamiento, porque la organización de la tabla se ordena por esta clave. Si se define una clave principal o una restricción exclusiva en las mismas columnas que la clave de secuencia de 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 ALLOW OVERFLOW, el nombre de la restricción de comprobación será RCT_OFLOW y no se aplica la restricción de comprobación.

STARTING FROM *constante*
Especifica el valor constante en el extremo inferior del rango para *nombre-columna*. Los valores menores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821). Si no se especifica ninguna constante inicial, el valor por omisión es 1.

ENDING AT *constante*
Especifica el valor constante en el extremo superior del rango para *nombre-columna*. Los valores mayores que la constante especificada sólo se permiten si se especifica la opción ALLOW OVERFLOW. El valor de la constante final debe ser mayor que la constante inicial. Si *nombre-columna* es una columna SMALLINT o INTEGER, la constante debe ser una constante INTEGER. Si *nombre-columna* es una columna BIGINT, la constante debe ser una constante INTEGER o BIGINT (SQLSTATE 42821).

ALLOW OVERFLOW
Especifica que la tabla 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 que intervienen estas filas de desbordamiento son menos eficaces que las operaciones en filas que tienen valores de clave dentro del rango definido.

DISALLOW OVERFLOW
Especifica que la tabla 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.

PCTFREE *entero*
Especifica el porcentaje de cada página que debe dejarse como espacio libre. La primera fila de cada página se añade sin restricciones. Cuando se añaden filas adicionales a una página, al menos *entero* por ciento de la página se deja como espacio libre. El valor de *entero* entra en un rango que va de 0 a 99. El valor -1 de PCTFREE en el catálogo del sistema (SYSCAT.TABLES) se interpreta como el valor por omisión. El valor de PCTFREE por omisión para una página de tabla es 0.

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).

La captura de datos es incompatible con la compresión de filas (SQLSTATE 42997).

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

CHANGES
Indica que en el archivo de anotaciones cronológicas se registrará información adicional referente a los cambios de SQL efectuados en esta

tabla. Esta opción es necesaria para duplicar la tabla y cuando se utiliza el programa Capture para capturar los cambios contenidos en el archivo de anotaciones para esta tabla.

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

IN *nombre1-espacio-tablas***,...**
Identifica los espacios de tablas en que se va a crear la tabla. Los espacios de tablas deben existir, deben encontrarse en el mismo grupo de particiones de base de datos y deben ser todos espacios de tablas DMS normales o todos DMS grandes o todos SMS (SQLSTATE 42838) en los que el ID de autorización de la sentencia posee el privilegio USE.

Se permite como máximo una cláusula IN en el nivel de tabla. Todos los espacios de tablas que utiliza una tabla deben tener el mismo tamaño de página y el mismo tamaño de extensión. Si no tienen todos el mismo tamaño de captación previa, se devuelve un aviso. Si todos los espacios de tablas tienen el tamaño de captación previa AUTOMATIC, no se devuelve ningún aviso.

Si tan solo se especifica un espacio de tablas, todas las partes de la tabla se almacenan en este espacio de tablas. No puede especificarse esta cláusula cuando se crea una subtabla (SQLSTATE 42613), porque el espacio de tablas se hereda de la tabla raíz de la jerarquía de tablas. Si no se especifica esta cláusula, el 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 tamaño de página
    THEN elegirlo
ELSE IF un espacio de tablas (para el que el usuario tiene privilegio USE)
  existe con suficiente tamaño de página (vea más abajo cuando
  están capacitados varios espacios de tablas)
    THEN elegirlo
ELSE devolver un error (SQLSTATE 42727)
```

Si la condición ELSE IF identifica más de un espacio de tablas, elija el espacio de tablas con el tamaño de página suficiente inferior. Si es apto más de un espacio de tablas, elija el espacio de tablas de acuerdo con el orden de preferencia siguiente, según el usuario a quien se haya otorgado el privilegio USE:

1. El ID de autorización
2. Un grupo al que pertenezca el ID de autorización
3. PUBLIC

Si todavía existe más de un espacio de tablas que puede elegirse, el gestor de bases de datos toma la decisión final.

La determinación del espacio de tablas puede cambiar si:

- Se descartan o crean espacios de tablas
- Se otorgan o revocan los privilegios USE

Las tablas particionadas pueden tener las particiones de datos repartidas por varios espacios de tablas. Cuando se especifican varios espacios de tablas, todos los espacios de tablas deben existir y todos deben ser espacios de tablas SMS, DMS normales o DMS grandes (SQLSTATE 42838). El ID de autorización de la sentencia debe poseer el privilegio USE para todos los espacios de tablas especificados.

El tamaño de página suficiente de una tabla está determinado por el número de bytes de la fila o por el número de columnas. Para obtener más información, consulte el apartado Tamaño de fila.

Si una tabla se coloca en un espacio de tablas grande:

- La tabla puede ser mayor que una tabla de un espacio de tablas normal. Para obtener detalles sobre los límites de espacio de tablas y de tabla, consulte el apartado "Límites de SQL".
- La tabla puede dar soporte a más de 255 filas por página de datos, lo que puede mejorar la utilización del espacio de las páginas de datos.
- Los índices definidos en la tabla requerirán una entrada de dos bytes por fila adicional, a diferencia de los índices definidos en una tabla que reside en un espacio de tablas normal.

CYCLE o NO CYCLE
Especifica si el número de particiones de datos sin ningún espacio de tablas explícito puede exceder el número de particiones de datos especificadas.

CYCLE
Especifica que, si el número de particiones de datos sin ningún espacio de tablas explícito excede el número de particiones de datos especificadas, los espacios de tablas se asignan a las particiones de datos de modo rotativo.

NO CYCLE
Especifica que el número de particiones de datos sin ningún espacio de tablas explícito no puede exceder el número de particiones de datos especificadas (SQLSTATE 428G1). Esta opción evita la asignación rotativa de los espacios de tablas a las particiones de datos.

opciones-espaciotablas
Especifica el espacio tablas en que se almacenarán los valores de los índices o las columnas largas. Para obtener detalles sobre los tipos de espacios de tablas, consulte "CREATE TABLESPACE".

INDEX IN *nombre2-espacio-tablas*
Identifica el espacio de tablas donde se crearán todos los índices de la tabla. El espacio de tablas especificado debe existir; debe ser un espacio de tablas DMS si la tabla tiene datos en espacios de tablas DMS o un espacio de tablas SMS si la tabla particionada tiene datos en espacios de tablas SMS; debe ser un espacio de tablas en que el ID de autorización de la sentencia posea el privilegio USE y debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tablas1* (SQLSTATE 42838).

Se puede especificar qué espacio de tablas contendrá los índices al crear una tabla o, en el caso de las tablas particionadas, al especificar la cláusula IN de la sentencia CREATE INDEX. La comprobación del privilegio USE para el espacio de tablas se realiza al crear la tabla, no al crear posteriormente un índice.

LONG IN *nombre3-espacio-tablas*
Identifica los espacios de tablas donde se almacenarán los valores de todas las columnas largas. Las columnas largas incluyen las de tipos de datos LOB, tipo XML, tipos diferenciados con cualquiera de estos tipos como fuente o cualquier columna definida con tipos estructurados

definidos por el usuario cuyos valores no se pueden almacenar en línea. Esta opción sólo está permitida si la cláusula IN identifica un espacio de tablas DMS.

El espacio de tablas especificado debe existir. Puede ser un espacio de tablas normal si es el mismo espacio de tablas en el que están almacenados los datos; de lo contrario, debe ser un espacio de tablas DMS grande en el que el ID de autorización de la sentencia tenga el privilegio USE. También debe encontrarse en el mismo grupo de particiones de base de datos que *nombre-espacio-tablas1* (SQLSTATE 42838).

Se puede especificar qué espacio de tablas contendrá las columnas largas, LOB o XML al crear una tabla. La comprobación del privilegio USE se realiza al crear la tabla, no al añadir posteriormente una columna larga o LOB.

Para obtener información sobre las normas que rigen el uso de la cláusula LONG IN con tablas particionadas, consulte el apartado "Comportamiento de objetos grandes en tablas particionadas".

cláusula-distribución

Especifica el particionamiento de base de datos o el modo en que se distribuyen los datos en diversas particiones de base de datos.

DISTRIBUTE BY HASH (*nombre-columna***,...)**

Especifica la utilización de la función de hash por omisión en las columnas especificadas (denominada *clave de distribución*) como método de distribución en las particiones de base de datos. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No se puede utilizar como parte de una clave de distribución ninguna columna cuyo tipo de datos sea BLOB, CLOB, DBCLOB, XML, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado (SQLSTATE 42962). La clave de distribución no puede contener una columna ROW CHANGE TIMESTAMP (SQLSTATE 429BV). No puede especificarse una clave de distribución para una tabla que sea una subtabla (SQLSTATE 42613) porque la clave de distribución se hereda de la tabla raíz de la jerarquía de tablas o una tabla con una columna que tenga el tipo de datos XML (SQLSTATE 42997). Si no se especifica esta cláusula y la tabla reside en un grupo de particiones de base de datos de varias particiones, la clave de distribución se define de la forma siguiente:

- Si la tabla es una tabla con tipo, la columna de identificador de objeto es la clave de distribución.
- Si se especifica una clave primaria, la primera columna de la clave primaria es la clave de distribución.
- En caso contrario, la primera columna con un tipo de datos válido para una clave de distribución se convierte en la clave de distribución.

Las columnas de la clave de distribución deben ser un subconjunto de las columnas que forman alguna restricción de unicidad.

Si ninguna columna cumple los requisitos de una clave de distribución por omisión, la tabla se crea sin ninguna. Estas tablas solo están permitidas en espacios de tablas definidos en grupos de particiones de base de datos de partición única.

Para las tablas de los espacios de tablas que están definidos en grupos de particiones de base de datos de partición única, se puede utilizar cualquier

conjunto de columnas con tipos de datos válidos para una clave de distribución a fin de definir la clave de distribución. Si no especifica esta cláusula, no se crea ninguna clave de distribución.

Para ver las restricciones relacionadas con la clave de distribución, consulte el apartado Normas.

DISTRIBUTE BY REPLICATION
Especifica que los datos almacenados en la tabla se duplican físicamente en cada partición de base de datos del grupo de particiones de base de datos de los espacios de tablas en que se define la tabla. Esto significa que existe una copia de todos los datos de la tabla en cada una de las particiones de base de datos. Esta opción sólo puede especificarse para una tabla de consultas materializadas (SQLSTATE 42997).

cláusula-particionamiento
Especifica cómo se particionan los datos en una partición de base de datos.

PARTITION BY RANGE *espec-partición-rango*
Especifica el esquema de particionamiento de rango de la tabla.

expresión-partición
Especifica los datos de clave en que se define el rango para determinar la partición de datos de destino de los datos.

nombre-columna
Identifica una columna de la clave de particionamiento de datos. El *nombre-columna* debe ser un nombre no calificado que identifica una columna de la tabla (SQLSTATE 42703). La misma columna no se puede identificar más de una vez (SQLSTATE 42709). No se puede utilizar como parte de una clave de particionamiento de tablas ninguna columna con un tipo de datos que sea BLOB, CLOB, DBCLOB, tipo diferenciado basado en cualquiera de estos tipos o tipo estructurado (SQLSTATE 42962).

Las normas para las literales numéricas gobiernan las literales numéricas utilizadas en la especificación del rango. Todas las literales numéricas (excepto los valores especiales de coma flotante decimal) utilizadas en los rangos correspondientes a las columnas numéricas se interpretan como constantes decimales, de coma flotante o entero, con arreglo a las normas especificadas para las constantes numéricas. En consecuencia, para las columnas de coma flotante decimal, el valor de constante numérica mínima y máxima que puede utilizarse en la especificación del rango de una partición de datos es el valor de DOUBLE más pequeño y el valor de DOUBLE más grande, respectivamente. Los valores especiales de coma flotante decimal pueden utilizarse en la especificación de rangos. Todos los valores especiales de coma flotante decimal se interpretan como superiores a MINVALUE e inferiores a MAXVALUE.

Las columnas de partición de rango no pueden contener una columna ROW CHANGE TIMESTAMP (SQLSTATE 429BV). El número de columnas identificadas no debe superar 16 (SQLSTATE 54008).

NULLS LAST
Indica que los valores nulos se comparan altos.

NULLS FIRST
Indica que los valores nulos se comparan bajos.

elemento-partición
Especifica los rangos de una clave de particionamiento de datos y el espacio de tablas donde se almacenarán las filas de la tabla del rango.

PARTITION *nombre-partición*
Indica el nombre de la partición de datos. El nombre no puede ser el mismo que el de ninguna otra partición de datos de la tabla (SQLSTATE 42710). Si no se especifica esta cláusula, el nombre será 'PART' seguido del formato de caracteres de un valor entero para que el nombre sea exclusivo para la tabla.

espec-límite
Especifica los límites de una partición de rango. La partición de rango inferior debe contener una cláusula-inicial y la partición de rango superior debe contener una cláusula-final (SQLSTATE 56016). Las particiones de rango entre la inferior y la superior pueden contener una cláusula-inicial, una cláusula-final o ambas. Si solo se especifica la cláusula-final, la partición de rango anterior también debe contener una cláusula-final (SQLSTATE 56016).

cláusula-inicial
Especifica el extremo inferior del rango de una partición de datos. Debe haber como mínimo un valor inicial, y el número máximo de valores es el número de columnas de la clave de particionamiento de datos (SQLSTATE 53038). Si hay especificados menos valores que el número de columnas, los demás valores son implícitamente MINVALUE.

STARTING FROM
Introduce la *cláusula-inicial*.

constante
Especifica un valor constante con un tipo de datos asignable al tipo de datos del *nombre-columna* al que corresponde (SQLSTATE 53045). El valor no puede estar dentro del rango de ninguna otra espec-límite de la tabla (SQLSTATE 56016).

MINVALUE
Especifica un valor inferior al valor mínimo posible para el tipo de datos del *nombre-columna* al que corresponde.

MAXVALUE
Especifica un valor superior al valor máximo posible para el tipo de datos del *nombre-columna* al que corresponde.

INCLUSIVE
Indica que los valores de rango especificados deben incluirse en la partición de datos.

EXCLUSIVE
Indica que los valores *constantes* especificados deben excluirse de la partición de datos. Esta especificación se omite cuando se especifica MINVALUE o MAXVALUE.

cláusula-final
Especifica el extremo superior del rango de una partición de datos. Debe haber como mínimo un valor inicial, y el número máximo de valores es el número de columnas de la clave de

particionamiento de datos (SQLSTATE 53038). Si hay especificados menos valores que el número de columnas, los demás valores son implícitamente MAXVALUE.

ENDING AT
Introduce la *cláusula-final*.

constante
Especifica un valor constante con un tipo de datos asignable al tipo de datos del *nombre-columna* al que corresponde (SQLSTATE 53045). El valor no puede estar dentro del rango de ninguna otra espec-límite de la tabla (SQLSTATE 56016).

MINVALUE
Especifica un valor inferior al valor mínimo posible para el tipo de datos del *nombre-columna* al que corresponde.

MAXVALUE
Especifica un valor superior al valor máximo posible para el tipo de datos del *nombre-columna* al que corresponde.

INCLUSIVE
Indica que los valores de rango especificados deben incluirse en la partición de datos.

EXCLUSIVE
Indica que los valores *constantes* especificados deben excluirse de la partición de datos. Esta especificación se omite cuando se especifica MINVALUE o MAXVALUE.

IN *nombre-espacio-tablas*
Especifica el espacio de tablas donde debe almacenarse la partición de datos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de autorización de la sentencia posea el privilegio USE. Si no se especifica esta cláusula, por omisión se asigna un espacio de tablas de modo rotativo a partir de la lista de espacios de tablas especificada para la tabla.

Si no se especifica la cláusula LONG IN en la sentencia CREATE TABLE, los objetos grandes se colocan en el mismo espacio de tablas que la partición de datos.

Si no se especifica la cláusula INDEX IN en la sentencia CREATE TABLE o CREATE INDEX, el índice se coloca en el mismo espacio de tablas que la primera partición visible o enlazada de la tabla.

LONG IN *nombre-espacio-tablas*
Especifica el espacio de tablas donde debe almacenarse la partición de datos que contiene datos de columna largos. El espacio de tablas especificado debe tener el mismo tamaño de página, debe estar en el mismo grupo de particiones de base de datos y debe gestionar el espacio del mismo modo que los demás espacios de tablas y particiones de datos de la tabla particionada (SQLSTATE 42838); debe ser un espacio de tablas en el que el ID de

autorización de la sentencia posea el privilegio USE. El tamaño de página y el tamaño de extensión del espacio de tablas especificado pueden ser distintos al tamaño de página y al tamaño de extensión de las otras particiones de datos de la tabla particionada.

EVERY (*constante***)**
Especifica el ancho de cada rango de partición de datos al utilizar el formato de la sintaxis generado automáticamente. Las particiones de datos se crearán comenzando por el valor STARTING FROM y con este número de valores en el rango. Este formato de la sintaxis solo está permitido para las tablas particionadas por una única columna numérica o de fecha y hora (SQLSTATE 53038).

Si la columna de clave de particionamiento es de tipo numérico, el valor inicial de la primera partición es el valor especificado en la cláusula-inicial. El valor final de la primera partición y todas las demás se calcula añadiendo el valor inicial de la partición al valor de incremento especificado como *constante* en la cláusula EVERY. El valor inicial de todas las demás particiones se calcula tomando el valor inicial de la partición anterior y añadiendo el valor de incremento especificado como *constante* en la cláusula EVERY.

Si la columna de clave de particionamiento es una fecha o una indicación de fecha y hora, el valor inicial de la primera partición es el valor especificado en la cláusula-inicial. El valor final de la primera partición y todas las demás se calcula añadiendo el valor inicial de la partición al valor de incremento especificado como duración etiquetada en la cláusula EVERY. El valor inicial de todas las demás particiones se calcula tomando el valor inicial de la partición anterior y añadiendo el valor de incremento especificado como duración etiquetada en la cláusula EVERY.

Para una columna de tipo numérico, el valor EVERY debe ser una constante numérica positiva; para una columna de fecha y hora, el valor EVERY debe ser una duración etiquetada (SQLSTATE 53045).

COMPRESS
Especifica si se aplica la compresión de datos a las filas de la tabla.

YES
Especifica que la compresión de filas de datos está habilitada. Las operaciones de inserción y actualización de la tabla estarán sujetas a compresión. Después de que la tabla se haya llenado de datos de forma suficiente, se crea automáticamente un diccionario de compresión y las filas están sujetas a compresión.

NO
Especifica que la compresión de filas de datos está inhabilitada.

VALUE COMPRESSION
Determina el formato de fila que debe utilizarse. Cada tipo de datos tiene un número total de bytes distinto según el formato de fila que se utilice. Para obtener más información, consulte el apartado Número total de bytes. Si la tabla es una tabla con tipo, esta opción sólo recibe soporte en la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR).

El valor NULL se almacena utilizando tres bytes. Es el mismo espacio o menor que cuando VALUE COMPRESSION no se ha activado para las columnas de todos los tipos de datos, con la excepción de CHAR(1). El hecho de definir o

no una columna como anulable no afecta al cálculo del tamaño de fila. Los valores de datos de longitud cero para las columnas cuyo tipo de datos es VARCHAR, VARGRAPHIC, LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB, BLOB o XML deben almacenarse utilizando dos bytes únicamente, que es un valor inferior al almacenamiento necesario cuando VALUE COMPRESSION no se ha activado. Cuando una columna se define con la opción COMPRESS SYSTEM DEFAULT, se permite también que el valor por omisión del sistema perteneciente a la columna se almacene utilizando tres bytes de almacenamiento total. El formato de fila utilizado para dar soporte a esto determina el número total de bytes de cada tipo de datos y tiende a causar la fragmentación de los datos al actualizar al valor NULL, a un valor de longitud cero o al valor por omisión del sistema o bien al actualizar desde estos valores.

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

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 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 tablas creadas con CCSID UNICODE en funciones de SQL o en métodos de SQL (SQLSTATE 560C0).
- Una sentencia de SQL que hace referencia a una tabla creada con CCSID UNICODE no puede invocar una función de SQL ni un método de SQL (SQLSTATE 53090).
- Los tipos gráficos, el tipo XML y los tipos definidos por el usuario no se pueden utilizar en tablas CCSID UNICODE (SQLSTATE 560C1).
- 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.

SECURITY POLICY
Indica el nombre de la política de seguridad que se asociará a la tabla.

nombre-política
Identifica una política de seguridad que ya existe en el servidor actual (SQLSTATE 42704).

OPTIONS (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* se debe especificar entre comillas simples.

El servidor remoto (el nombre de servidor que se ha especificado en la sentencia CREATE SERVER) se debe especificar en la cláusula OPTIONS. La cláusula OPTIONS también se puede utilizar para alterar temporalmente el esquema o el nombre no calificado de la tabla base remota que se crea.

Se recomienda especificar el nombre de esquema. Si no se especifica un nombre de esquema remoto, se utiliza el calificador para el nombre de tabla. Si el nombre de tabla no tiene calificador, se utiliza el ID de autorización de la sentencia.

Si no se especifica un nombre no calificado para la tabla base remota, se utiliza *nombre-tabla*.

Normas

- La suma de las cuentas de bytes de las columnas, incluidas las longitudes en línea de todas las columnas de tipo estructurado o XML no debe ser mayor que el límite de tamaño de la fila, que está basado en el tamaño de página del espacio de tablas (SQLSTATE 54010). Para obtener más información, consulte el apartado Número total de bytes. Para las tablas con tipo, el número total de bytes se aplica a las columnas de la tabla raíz de la jerarquía de 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 de una tabla no puede ser superior a 1.012 (SQLSTATE 54011). Para las tablas con tipo, el número total de atributos de los tipos de todas las subtablas de la jerarquía de tablas no puede sobrepasar la cantidad de 1010.
- Una columna de identificador de objeto de una tabla con tipo no puede actualizarse (SQLSTATE 42808).
- Cualquier restricción de clave primaria o de unicidad definida en la tabla debe ser un superconjunto de la clave de distribución (SQLSTATE 42997).
- Las normas siguientes solo se aplican a las bases de datos de varias particiones de base de datos.
  - Las tablas formadas sólo por columnas con tipos LOB, XML, un tipo diferenciado basado en uno de estos tipos o un tipo estructurado sólo se pueden crear en espacios de tablas que estén definidos en grupos de particiones de bases de datos de una sola partición.
  - La definición de clave de distribución de una tabla de un espacio de tablas que se ha definido en un grupo de particiones de base de datos de varias particiones no puede modificarse.

  - La columna de clave de distribución de una tabla con tipo debe ser la columna de OID.
  - No se pueden utilizar columnas de tipo XML.
  - No están permitidas las tablas de etapas particionadas.
- Se aplican las siguientes restricciones a las tablas de clúster de rango:
  - No se puede especificar una tabla de clúster de rango en una base de datos con varias particiones de base de datos (SQLSTATE 42997).
  - 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.
  - Las columnas no pueden ser de tipo XML.
- Una tabla no está protegida salvo que tenga asociada una política de seguridad y contenga una columna de tipo DB2SECURITYLABEL o una columna definida con la cláusula SECURED WITH. El primer caso indica que la tabla es una tabla protegida con **granularidad en el nivel de fila** y el segundo indica que la tabla es una tabla protegida con **granularidad en el nivel de columna**.
- La declaración de una columna de tipo DB2SECURITYLABEL falla si la tabla no tiene asociada una política de seguridad (SQLSTATE 55064).
- No se puede añadir una política de seguridad a una tabla con tipos (SQLSTATE 428DH), tabla de consultas materializadas o tabla de etapas (SQLSTATE 428FG).
- No se puede especificar una *expresión-tabla-anidada* tolerante a errores en la selección completa de una *definición-consulta-materializada* (SQLSTATE 428GG).

Notas

- La creación de una tabla con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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.
- Las columnas VARCHAR y VARGRAPHIC que son mayores de 4.000 y 2.000 respectivamente no deben utilizarse como parámetros de entrada en funciones en esquema SYSFUN. Se producirán errores si se invoca la función con un valor de argumento que excede estas longitudes (SQLSTATE 22001).
- La utilización de NO ACTION o RESTRICT como normas de supresión o actualización para restricciones de referencia determina cuándo se aplica la restricción. Una norma de supresión o actualización de RESTRICT se aplicará *antes* que las demás restricciones, incluidas las restricciones de referencia con normas de modificación como CASCADE o SET NULL. Una norma de supresión

o de actualización de NO ACTION se aplica *después* de otras restricciones de referencia. Un ejemplo en el que es evidente un comportamiento distinto implica la supresión de filas de una vista que se ha definido como UNION ALL de las tablas relacionadas.

```
La tabla T1 es una tabla padre de la tabla T3; norma de supresión, como
   se explica a continuación.
La tabla T2 es una tabla padre de la tabla T3; norma de supresión CASCADE.

CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2

DELETE FROM V1
```

Si la tabla T1 es una tabla padre de la tabla T3 que tiene la norma de supresión RESTRICT, se producirá una violación de restricción (SQLSTATE 23001) si existen filas hijas para las claves padre de T1 en T3.

Si la tabla T1 es una tabla padre de la tabla T3 que tiene la norma de supresión de NO ACTION, la norma de supresión de CASCADE puede suprimir las filas hijas al suprimirse las filas de T2, antes de que se aplique la norma de supresión de NO ACTION para las supresiones de T1. Si las supresiones de T2 no han tenido como resultado la supresión de todas las filas hijas para las claves padre de T1 en T3, se generará una violación de restricción (SQLSTATE 23504).

Observe que el SQLSTATE que se devuelve es diferente dependiendo de si la norma de supresión o actualización es RESTRICT o NO ACTION.

- Para las tablas de los espacios de tablas que se han definido en grupos de particiones de base de datos de varias particiones, en la elección de las claves de distribución debe tenerse en cuenta la colocación de la tabla. A continuación encontrará una lista de elementos que debe tener en cuenta:
  - Para la colocación, las tablas deben encontrarse en el mismo grupo de particiones de base de datos. Los espacios de tablas pueden ser distintos, pero deben haberse definido en el mismo grupo de particiones de base de datos.
  - Las claves de distribución de las tablas deben tener el mismo número de columnas, y las columnas de clave correspondientes deben tener particiones de base de datos compatibles para la colocación.
  - La elección de la clave de distribución también afecta al rendimiento de las uniones. Si una tabla se une con frecuencia a otra tabla, plantéese la conveniencia de unir las columnas como una clave de distribución para ambas tablas.
- La opción NOT LOGGED INITIALLY es útil para las situaciones en que se debe crear un conjunto de resultados grande con datos de una fuente alternativa (otra tabla o un archivo) y la recuperación de la tabla no es necesaria. La utilización de esta opción ahorrará la actividad general de anotar cronológicamente los datos. Las siguientes consideraciones se aplican cuando se especifica esta opción:
  - Cuando se confirma la unidad de trabajo, todos los cambios que se han realizado en la tabla durante la unidad de trabajo fluyen al disco.
  - Cuando ejecuta el programa de utilidad de avance (Rollforward) y éste detecta un registro de anotaciones cronológicas que indica que el programa de utilidad de carga (Load) ha llenado una tabla de la base de datos o que ésta se ha creado con la opción NOT LOGGED INITIALLY, la tabla se marcará como no disponible. El programa de utilidad de avance (Rollforward) 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 referencia de tabla
  - no debe incluir DISTINCT en ninguna parte dentro de la lista de selección
  - no debe incluir registros especiales
  - no debe incluir variables globales
  - 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 de carga o durante la ejecución de la sentencia SET INTEGRITY.
- Algunas operaciones no se pueden realizar en la tabla a la que hace referencia una tabla de 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 SQL. Esta posibilidad se conoce como *DDL transparente*. Para poder crear una tabla base remota en una fuente de datos, el servidor federado debe estar configurado para acceder a esa fuente de datos. Esta configuración incluye la creación de un derivador para la fuente de datos, el suministro de la definición del servidor para el servidor en el que se ubicará la tabla base remota y la creación de correlaciones de usuario entre el servidor federado y la fuente de datos.

  El DDL transparente impone algunas limitaciones en los elementos que pueden incluirse en la sentencia CREATE TABLE:

  – Sólo se pueden crear columnas y una clave primaria en la tabla base remota.
  – Entre las clases específicas que admite el DDL transparente se encuentran las siguientes:
    - *definición-columna* y *restricción-unicidad* en la cláusula *lista-elementos*
    - NOT NULL y PRIMARY KEY en la cláusula *opciones-columna*
    - OPTIONS
  – La fuente de datos remota debe dar soporte a:
    - Los tipos de datos de la columna remota con los que están correlacionados los tipos de datos de la columna DB2
    - La opción de clave primaria de la sentencia CREATE TABLE

    Según cómo responda la fuente de datos a las peticiones que no da soporte, puede que se devuelva un error o puede pasarse por alto la petición.

  Cuando se crea una tabla base remota utilizando un DDL transparente, se crea automáticamente un apodo para esta tabla base remota.
- Se puede definir una restricción de referencia de forma que la tabla padre o la tabla dependiente forme parte de una jerarquía de tablas. En este caso, el efecto de la restricción de referencia es el siguiente:
  1. Efectos de las sentencias INSERT, UPDATE y DELETE:
     – Si existe una restricción de referencia, en la que TP es una tabla padre y TD es una tabla dependiente, la restricción asegura que para cada fila de TD (o de cualquiera de sus subtablas) que tenga una clave foránea no nula, existe una fila en TP (o en una de sus subtablas) con una clave padre coincidente. Esta norma se aplica para cualquier acción que afecte a una fila de TP o TD, con independencia de cómo se inicie la acción.
  2. Efectos sobre las sentencias DROP TABLE:
     – para las restricciones de referencia en que la tabla 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 la restricción de referencia y se invoca su norma de supresión para cada una de las filas suprimidas.
     – para las restricciones de referencia en las que la tabla dependiente es una supertabla de la tabla 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, el definidor de la tabla recibe el privilegio CONTROL. Cuando se crea una subtabla, el definidor de la tabla otorga automáticamente para la subtabla el privilegio SELECT que cada usuario o grupo tiene sobre la supertabla inmediata.

- ***Tamaño de fila:*** el número máximo de bytes permitido en la fila de una tabla depende del tamaño de página del espacio de tablas donde se crea la tabla (*nombre1-espacio-tablas*). La lista siguiente muestra el límite del tamaño de fila y el límite del número de columnas asociados a cada tamaño de página del espacio de tablas.

Tabla 20. Límites para el número de columnas y el tamaño de fila de cada tamaño de página del espacio de tablas

Tamaño de página	Límite del tamaño de fila	Límite de la cuenta de columnas
4K	4.005	500
8K	8.101	1.012
16K	16.293	1.012
32K	32.677	1.012

El número de columnas real para una tabla puede limitarse más mediante la fórmula siguiente:

```
Columnas en total * 8 + Número de columnas LOB * 12 <=
  Límite del tamaño de fila para el tamaño de página
```

- ***Número total de bytes:*** la tabla siguiente contiene el número total de bytes de las columnas por tipo de datos. Esta tabla se utiliza para calcular el tamaño de fila. El número total de bytes depende de si se ha activado o no VALUE COMPRESSION. Cuando VALUE COMPRESSION no se ha activado, el número total de bytes también depende de si la columna es anulable o no.

  Si una tabla se basa en un tipo estructurado, se reservarán 4 bytes adicionales de actividad general para identificar las filas de subtablas, con independencia de si se han definido o no subtablas. Las columnas de subtabla adicionales deben considerarse como columnas con posibilidad de nulos para la obtención del número total de bytes, incluso si se han definido como columnas sin posibilidad de nulos.

Tabla 21. Número total de bytes de las columnas por tipo de datos

	VALUE COMPRESSION se	VALUE COMPRESSION no se ha activado	
Tipo de datos	ha activado[1]	La columna es anulable	La columna no es anulable
SMALLINT	4	3	2
INTEGER	6	5	4
BIGINT	10	9	8
REAL	6	5	4
DOUBLE	10	9	8
DECIMAL	La parte integral de (p/2)+3, donde p es la precisión	La parte integral de (p/2)+2, donde p es la precisión	La parte integral de (p/2)+1, donde p es la precisión
DECFLOAT(16)	10	9	8
DECFLOAT(34)	18	17	16
CHAR (n)	n+2	n+1	n
VARCHAR (n)	n+2	n+5 (dentro de una tabla)	n+4 (dentro de una tabla)
LONG VARCHAR[2]	22	25	24
GRAPHIC (n)	n*2+2	n*2+1	n*2
VARGRAPHIC (n)	n*2+2	n*2+5 (dentro de una tabla)	n*2+4 (dentro de una tabla)

Tabla 21. Número total de bytes de las columnas por tipo de datos (continuación)

Tipo de datos	VALUE COMPRESSION se ha activado[1]	VALUE COMPRESSION no se ha activado	
		La columna es anulable	La columna no es anulable
LONG VARGRAPHIC[2]	22	25	24
DATE	6	5	4
TIME	5	4	3
TIMESTAMP	12	11	10
XML (sin INLINE LENGTH especificado)	82	85	84
XML (con INLINE LENGTH especificado)	INLINE LENGTH +2	INLINE LENGTH +4	INLINE LENGTH +3
Longitud máxima de LOB[3] 1024	70	73	72
Longitud máxima de LOB 8192	94	97	96
Longitud máxima de LOB 65.536	118	121	120
Longitud máxima de LOB 524.000	142	145	144
Longitud máxima de LOB 4.190.000	166	169	168
Longitud máxima de LOB 134.000.000	198	201	200
Longitud máxima de LOB 536.000.000	222	225	224
Longitud máxima de LOB 1.070.000.000	254	257	256
Longitud máxima de LOB 1.470.000.000	278	281	280
Longitud máxima de LOB 2.147.483.647	314	317	316

[1] Existen 2 bytes adicionales de almacenamiento utilizados por cada fila cuando VALUE COMPRESSION se ha activado para esa fila.

[2]Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro.

[3] Cada valor de LOB tiene un *descriptor de LOB* en el registro base que apunta a la ubicación del valor real. El tamaño de dicho descriptor varía en función de la longitud máxima que se haya definido para la columna.

Para un *tipo diferenciado*, el número total de bytes equivale a la longitud del tipo de fuente del tipo diferenciado. Para un *tipo de referencia*, el número total de bytes equivale a la longitud del tipo de datos incorporado en el que se basa el tipo de referencia. Para un *tipo estructurado*, el número total de bytes equivale a INLINE LENGTH + 4. El valor INLINE LENGTH es el valor que se especifica (o que se calcula implícitamente) para la columna de la cláusula *opciones-columna*.

Los tamaños de fila para las tablas de ejemplo siguiente parten de que no se especifica VALUE COMPRESSION:

```
DEPARTMENT 63 (0 + 3 + 33 + 7 +  3 + 17)
ORG        57 (0 + 3 + 19 + 2 + 15 + 18)
```

Si VALUE COMPRESSION se especifica, los tamaños de fila pasarán a ser los siguientes:

```
DEPARTMENT 69 (2 + 5 + 31 + 8 +  5 + 18)
ORG        53 (2 + 4 + 16 + 4 + 12 + 15)
```

- ***Número total de bytes de almacenamiento:*** la tabla siguiente contiene el número total de bytes de almacenamiento de las columnas por tipo de datos para valores de datos. El número total de bytes depende de si se ha activado o no VALUE COMPRESSION. Cuando VALUE COMPRESSION no se ha activado, el número total de bytes también depende de si la columna es anulable o no. Los valores de la tabla representan la cantidad de almacenamiento (en bytes) que se utiliza para almacenar el valor.

Tabla 22. Número total de bytes de almacenamiento basado en el formato de fila, tipo de datos y valor de datos

Valor de datos ?	**NULL**	**NULL**	**longitud cero**	**valor por omisión del sistema[2]**	**los otros valores de datos**	**los otros valores de datos**	**los otros valores de datos**
VALUE COMPRES-SION ?	**no activo**	**activo[1]**	**activo[1]**	**activo[1]**	**no activo**	**no activo**	**activo[1]**
Posibilidad de nulo de columna ?	**anulable**	**anulable**	**n/d**	**n/d**	**anulable**	**no anulable**	**n/d**
Tipo de datos							
SMALLINT	3	3	-	3	3	2	4
INTEGER	5	3	-	3	5	4	6
BIGINT	9	3	-	3	9	8	10
REAL	5	3	-	3	5	4	6
DOUBLE	9	3	-	3	9	8	10
DECIMAL	La parte integral de (*p*/2)+2, donde *p* es la precisión	3	-	3	La parte integral de (*p*/2)+2, donde *p* es la precisión	La parte integral de (*p*/2)+1, donde *p* es la precisión	La parte integral de (*p*/2)+3, donde *p* es la precisión
DECFLOAT(16)	9	3	-	3	9	8	10
DECFLOAT(34)	17	3	-	3	17	16	18
CHAR (*n*)	*n*+1	3	-	3	*n*+1	*n*	*n*+2
VARCHAR (*n*)	5	3	2	2	*N*+5, donde *N* es el número de bytes en los datos	*N*+4, donde *N* es el número de bytes en los datos	*N*+2, donde *N* es el número de bytes en los datos
LONG VARCHAR[3]	5	3	2	2	25	24	22
GRAPHIC (*n*)	*n**2+1	3	-	3	*n**2+1	*n**2	*n**2+2
VARGRAPHIC (*n*)	5	3	2	2	*N**2+5, donde *N* es el número de bytes en los datos	*N**2+4, donde *N* es el número de bytes en los datos	*N**2+2, donde *N* es el número de bytes en los datos
LONG VARGRAPHIC[3]	5	3	2	2	25	24	22
DATE	5	3	-	-	5	4	6
TIME	4	3	-	-	4	3	5
TIMESTAMP	11	3	-	-	11	10	12

Tabla 22. Número total de bytes de almacenamiento basado en el formato de fila, tipo de datos y valor de datos (continuación)

Valor de datos ?	NULL	NULL	longitud cero	valor por omisión del sistema[2]	los otros valores de datos	los otros valores de datos	los otros valores de datos
VALUE COMPRES-SION ?	no activo	activo[1]	activo[1]	activo[1]	no activo	no activo	activo[1]
Posibilidad de nulo de columna ?	anulable	anulable	n/d	n/d	anulable	no anulable	n/d
Tipo de datos							
Longitud máxima de LOB[2] 1.024	5	3	2	2	(De 60 a 68)+5	(De 60 a 68)+4	(De 60 a 68)+2
Longitud máxima de LOB 8192	5	3	2	2	(De 60 a 92)+5	(De 60 a 92)+4	(De 60 a 92)+2
Longitud máxima de LOB 65.536	5	3	2	2	(De 60 a 116)+5	(De 60 a 116)+4	(De 60 a 116)+2
Longitud máxima de LOB 524.000	5	3	2	2	(De 60 a 140)+5	(De 60 a 140)+4	(De 60 a 140)+2
Longitud máxima de LOB 4.190.000	5	3	2	2	(De 60 a 164)+5	(De 60 a 164)+4	(De 60 a 164)+2
Longitud máxima de LOB 134.000.000	5	3	2	2	(De 60 a 196)+5	(De 60 a 196)+4	(De 60 a 196)+2
Longitud máxima de LOB 536.000.000	5	3	2	2	(De 60 a 220)+5	(De 60 a 220)+4	(De 60 a 220)+2
Longitud máxima de LOB 1.070.000.000	5	3	2	2	(De 60 a 252)+5	(De 60 a 252)+4	(De 60 a 252)+2
Longitud máxima de LOB 1.470.000.000	5	3	2	2	(De 60 a 276)+5	(De 60 a 276)+4	(De 60 a 276)+2
Longitud máxima de LOB 2.147.483.647	5	3	2	2	(De 60 a 312)+5	(De 60 a 312)+4	(De 60 a 312)+2
XML	5	3	-	-	85	84	82

[1] Existen 2 bytes adicionales de almacenamiento utilizados por cada fila cuando VALUE COMPRESSION se ha activado para esa fila.

[2] Cuando COMPRESS SYSTEM DEFAULT se ha especificado para la columna.

[3]Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro.

- ***Columnas de dimensión:*** Puesto que cada valor diferenciado de una columna de dimensión se asigna a un bloque distinto de la tabla, puede que se desee aplicar un clúster en una expresión como, por ejemplo, "INTEGER(ORDER_DATE)/

100". En este caso, puede definirse una columna generada para la tabla y, a continuación, esta columna generada puede utilizarse en la cláusula ORGANIZE BY DIMENSIONS. Si la expresión es 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 sea una elección adecuada.
- Una tabla puede tener como máximo una política de seguridad.

- DB2 aplica las restricciones de integridad de referencia definidas en las tablas protegidas. En este caso las violaciones de restricción pueden ser difíciles de depurar, ya que DB2 no le permitirá ver qué fila ha ocasionado una violación si no tiene la etiqueta de seguridad o las credenciales de exenciones adecuadas.
- Al definir el orden de las columnas de una tabla, las columnas actualizadas con frecuencia deben colocarse al final de la definición para reducir la cantidad de datos anotados cronológicamente. Esto incluye las columnas ROW CHANGE TIMESTAMP. Se garantiza que las columnas ROW CHANGE TIMESTAMP se actualizarán con cada actualización de fila.
- ***Seguridad y duplicación:*** La duplicación puede hacer que las filas de datos de una tabla protegida se dupliquen fuera de la base de datos. Tenga cuidado al configurar la duplicación para una tabla protegida, porque DB2 no puede proteger los datos situados fuera de la base de datos.
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - La palabra clave CONSTRAINT puede omitirse en una *definición-columna* que defina a una cláusula de referencias
    - El *nombre-restricción* puede especificarse a continuación de FOREIGN KEY (sin la palabra clave CONSTRAINT)
    - SUMMARY puede especificarse opcionalmente tras CREATE
    - DEFINITION ONLY puede especificarse en lugar de WITH NO DATA
    - La cláusula PARTITIONING KEY puede especificarse en lugar de la cláusula DISTRIBUTE BY
    - Se puede especificar REPLICATED en lugar de DISTRIBUTE BY REPLICATION
  - Para mantener la compatibilidad con las versiones anteriores de DB2 y garantizar la coherencia:
    - Puede utilizarse una coma para separar varias opciones en la cláusula *opciones-identidad*
  - Para mantener la compatibilidad con DB2 para 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
    - Se puede especificar PART en lugar de PARTITION
    - Se puede especificar PARTITION *número-partición* en lugar de PARTITION *nombre-partición*. Un *número-partición* no puede identificar una partición que no se haya especificado anteriormente en la sentencia CREATE TABLE. Si no se especifica un *número-partición*, el gestor de bases de datos genera un número de partición exclusivo.
    - Se puede especificar VALUES en lugar de ENDING AT
  - También recibe soporte la sintaxis siguiente:
    - NOMINVALUE, NOMAXVALUE, NOCYCLE, NOCACHE y NOORDER

Ejemplos

Ejemplo 1: Cree la tabla TDEPT en el espacio de tablas DEPARTX. DEPTNO, DEPTNAME, MGRNO y ADMRDEPT son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. NOT NULL significa que la

columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. La clave primaria es la columna DEPTNO.

```
CREATE TABLE TDEPT
  (DEPTNO   CHAR(3)     NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT CHAR(3)     NOT NULL,
   PRIMARY KEY(DEPTNO))
IN DEPARTX
```

Ejemplo 2: Cree la tabla PROJ en el espacio de tablas SCHED. PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTAFF, PRSTDATE, PRENDATE y MAJPROJ son nombres de columnas. CHAR significa que la columna contendrá datos de caracteres. DECIMAL significa que la columna contendrá datos decimales empaquetados. 5,2 significa lo siguiente: 5 indica el número de dígitos decimales y 2 indica el número de dígitos a la derecha de la coma decimal. NOT NULL significa que la columna no puede contener ningún valor nulo. VARCHAR significa que la columna contendrá datos de caracteres de longitud variable. DATE significa que la columna contendrá información de fecha en un formato de tres partes (año, mes y día).

```
CREATE TABLE PROJ
  (PROJNO   CHAR(6)      NOT NULL,
   PROJNAME VARCHAR(24)  NOT NULL,
   DEPTNO   CHAR(3)      NOT NULL,
   RESPEMP  CHAR(6)      NOT NULL,
   PRSTAFF  DECIMAL(5,2)         ,
   PRSTDATE DATE                 ,
   PRENDATE DATE                 ,
   MAJPROJ  CHAR(6)      NOT NULL)
IN SCHED
```

Ejemplo 3: Cree una tabla llamada EMPLOYEE_SALARY donde los salarios desconocidos se consideren 0. No se especifica ningún espacio de tablas, de modo que la tabla se creará en un espacio de tablas seleccionado por el sistema basándose en las 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 TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS

CREATE TYPE JOHNDOE.MILES  AS FLOAT WITH COMPARISONS

CREATE TABLE EMPLOYEE
  (ID          INTEGER NOT NULL,
   NAME   CHAR (30),
   SALARY      T_SALARY NOT NULL WITH DEFAULT,
   LIVING_DIST MILES    DEFAULT MILES(1) )
```

Ejemplo 5: Cree tipos diferenciados para la imagen y audio y utilícelos para las columnas de una tabla. No se ha especificado ningún espacio de tablas para que la tabla se cree en un espacio de tablas que el sistema selecciona basándose en las normas que se describen para la cláusula IN *nombre-espacio-tabla1*. Suponga que el registro especial CURRENT PATH tiene el valor por omisión.

```
CREATE TYPE IMAGE AS BLOB (10M)

CREATE TYPE AUDIO AS BLOB (1G)

CREATE TABLE PERSON
  (SSN    INTEGER NOT NULL,
   NAME   CHAR (30),
   VOICE  AUDIO,
   PHOTO  IMAGE)
```

Ejemplo 6: Cree la tabla EMPLOYEE en el espacio de tablas HUMRES. Las restricciones definidas en la tabla son las siguientes:

- Los valores del número de departamento deben estar comprendidos entre 10 y 100.
- El trabajo de un empleado sólo puede ser 'Sales', 'Mgr' o 'Clerk'.
- Aquellos empleados que lleven en la compañía desde 1986 deben ganar más de 40.500 dólares.

Nota: Si las columnas incluidas en las restricciones de comprobación pueden contener valores nulos, también podrían ser NULL.

```
CREATE TABLE EMPLOYEE
  (ID          SMALLINT NOT NULL,
   NAME        VARCHAR(9),
   DEPT        SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
   JOB         CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
   HIREDATE    DATE,
   SALARY      DECIMAL(7,2),
   COMM        DECIMAL(7,2),
   PRIMARY KEY (ID),
   CONSTRAINT  YEARSAL CHECK (YEAR(HIREDATE) > 1986
     OR SALARY > 40500)
  )
  IN HUMRES
```

Ejemplo 7: Cree una tabla que esté completamente contenida en el espacio de tablas PAYROLL.

```
CREATE TABLE EMPLOYEE .....
  IN PAYROLL
```

Ejemplo 8: Cree una tabla con la parte de los datos en ACCOUNTING y la parte del índice en ACCOUNT_IDX.

```
CREATE TABLE SALARY.....
  IN ACCOUNTING INDEX IN ACCOUNT_IDX
```

Ejemplo 9: Cree una tabla y anote cronológicamente los cambios SQL en el formato por omisión.

```
CREATE TABLE SALARY1 .....
```

o

```
CREATE TABLE SALARY1 .....
  DATA CAPTURE NONE
```

Ejemplo 10: Cree una tabla y anote cronológicamente los cambios SQL en un formato expandido.

```
CREATE TABLE SALARY2 .....
  DATA CAPTURE CHANGES
```

Ejemplo 11: Cree una tabla EMP_ACT en el espacio de tablas SCHED. EMPNO, PROJNO, ACTNO, EMPTIME, EMSTDATE y EMENDATE son nombres de columna. Las restricciones definidas en la tabla son:

- El valor para el conjunto de columnas EMPNO, PROJNO y ACTNO en cualquier fila debe ser exclusivo.
- El valor de PROJNO debe coincidir con un valor existente para la columna PROJNO de la tabla PROJECT y, si el proyecto se suprime, todas las filas que hacen referencia al proyecto en EMP_ACT también deben suprimirse.

```
CREATE TABLE EMP_ACT
  (EMPNO       CHAR(6) NOT NULL,
   PROJNO      CHAR(6) NOT NULL,
   ACTNO       SMALLINT NOT NULL,
   EMPTIME     DECIMAL(5,2),
   EMSTDATE    DATE,
   EMENDATE    DATE,
   CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),
   CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)
                          REFERENCES PROJECT (PROJNO) ON DELETE CASCADE
  )
  IN SCHED
```

Se crea automáticamente un índice exclusivo denominado EMP_ACT_UNIQ en el mismo esquema para aplicar la restricción de unicidad.

Ejemplo 12: Cree una tabla que vaya a contener información sobre goles famosos para el 'hall of fame' del hockey sobre hielo. La tabla listará información sobre el jugador que marcó el gol, el portero contra el que lo marcó, la fecha y el lugar, así como una descripción. La columna de descripción puede contener nulos.

```
CREATE TABLE HOCKEY_GOALS
  ( BY_PLAYER      VARCHAR(30)   NOT NULL,
    BY_TEAM        VARCHAR(30)   NOT NULL,
    AGAINST_PLAYER VARCHAR(30)   NOT NULL,
    AGAINST_TEAM   VARCHAR(30)   NOT NULL,
    DATE_OF_GOAL   DATE          NOT NULL,
    DESCRIPTION    CLOB(5000) )
```

Ejemplo 13: Supongamos que se necesita una tabla de excepción para la tabla EMPLOYEE. Se puede crear una utilizando la sentencia siguiente.

```
CREATE TABLE EXCEPTION_EMPLOYEE AS
  (SELECT EMPLOYEE.*,
    CURRENT TIMESTAMP AS TIMESTAMP,
    CAST ('' AS CLOB(32K)) AS MSG
  FROM EMPLOYEE
  ) WITH NO DATA
```

Ejemplo 14: Supongamos los espacios de tablas siguientes que tienen los atributos indicados:

```
TBSPACE            PAGESIZE    USER   USERAUTH
------------------ ----------- ------ --------
DEPT4K                    4096 BOBBY  S
PUBLIC4K                  4096 PUBLIC S
DEPT8K                    8192 BOBBY  S
DEPT8K                    8192 RICK   S
PUBLIC8K                  8192 PUBLIC S
```

- Si RICK crea la tabla siguiente, se colocará en el espacio de tablas PUBLIC4K, pues el número de bytes es menor que 4005; pero si BOBBY crea la misma tabla, se colocará en el espacio de tablas DEPT4K, pues BOBBY tiene un privilegio USE otorgado explícitamente:

```
CREATE TABLE DOCUMENTS
  (SUMMARY    VARCHAR(1000),
   REPORT     VARCHAR(2000))
```

- Si BOBBY crea la tabla siguiente, se colocará en el espacio de tablas DEPT8K, pues el número de bytes es mayor que 4005, y BOBBY tiene un privilegio USE otorgado explícitamente. En cambio, si DUNCAN crea la misma tabla, se colocará en el espacio de tablas PUBLIC8K, pues DUNCAN no tiene ningún privilegio específico:

```
CREATE TABLE CURRICULUM
  (SUMMARY    VARCHAR(1000),
   REPORT     VARCHAR(2000),
   EXERCISES  VARCHAR(1500))
```

Ejemplo 15: Creación de una tabla que tiene una columna LEAD definida con el tipo estructurado EMP. Especifique 300 bytes para el valor INLINE LENGTH de la columna LEAD, lo cual significa que cualquier instancia de LEAD que no pueda caber dentro de los 300 bytes se almacenará fuera de la tabla (separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB).

```
CREATE TABLE PROJECTS (PID INTEGER,
  LEAD EMP INLINE LENGTH 300,
  STARTDATE DATE,
            ...)
```

Ejemplo 16: Creación de una tabla DEPT, con cinco columnas llamadas DEPTNO, DEPTNAME, MGRNO, ADMRDEPT y LOCATION. La columna DEPT debe definirse como columna de identidad para que DB2 genere siempre un valor para ella. Los valores de la columna DEPT deben comenzar en 500 y aumentar según incrementos de 1.

```
CREATE TABLE DEPT
  (DEPTNO     SMALLINT      NOT NULL
                 GENERATED ALWAYS AS IDENTITY
                   (START WITH 500, INCREMENT BY 1),
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT   SMALLINT      NOT NULL,
   LOCATION   CHAR(30))
```

Ejemplo 17: Cree una tabla SALES que se distribuya en la columna YEAR y tenga dimensiones en las columnas REGION y YEAR. Los datos se distribuirán por las particiones de base de datos según los valores de generación aleatoria de la columna YEAR. En cada partición de base de datos, los datos se organizarán en extensiones basadas en combinaciones exclusivas de los valores de las columnas REGION y YEAR de esas particiones de base de datos.

```
CREATE TABLE SALES
  (CUSTOMER            VARCHAR(80),
   REGION              CHAR(5),
   YEAR       INTEGER)
DISTRIBUTE BY HASH (YEAR)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
```

Ejemplo 18: Cree una tabla SALES con la columna PURCHASEYEARMONTH que se genera a partir de la columna PURCHASEDATE. Utilice una expresión para crear una columna que sea monotónica con respecto a la columna PURCHASEDATE original y, por lo tanto, que sea adecuada para utilizarla como

dimensión. La tabla se distribuye en la columna REGION y cada partición de base de datos se organiza en extensiones según la columna PURCHASEYEARMONTH; es decir, las distintas regiones estarán en particiones de base de datos diferentes y los distintos meses de compras pertenecerán a diferentes celdas (o conjuntos de extensiones) de esas particiones de base de datos.

```
CREATE TABLE SALES
  (CUSTOMER            VARCHAR(80),
   REGION              CHAR(5),
   PURCHASEDATE        DATE,
   PURCHASEYEARMONTH   INTEGER
                          GENERATED ALWAYS AS (INTEGER(PURCHASEDATE)/100))
DISTRIBUTE BY HASH (REGION)
ORGANIZE BY DIMENSIONS (PURCHASEYEARMONTH)
```

Ejemplo 19: Cree la tabla CUSTOMER con la columna CUSTOMERNUMDIM que se genera a partir de la columna CUSTOMERNUM. Utilice una expresión para crear una columna que sea 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 (en mayúsculas) en un servidor Informix y cree un apodo denominado EMPLOYEE (en mayúsculas) en esa tabla:

```
CREATE TABLE employee
  (EMP_NO       CHAR(6)       NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* no está delimitado, el nombre de tabla base remota aparecerá en letras mayúsculas, incluso si la fuente de datos remota almacena los nombres en minúsculas.

Cree una tabla base remota denominada employee (en minúsculas) en un servidor Informix y cree un apodo denominado EMPLOYEE (en mayúsculas) en esa tabla:

```
CREATE TABLE employee
  (EMP_NO       CHAR(6)       NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER',
   REMOTE_TABNAME '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 (en minúsculas) en un servidor Informix y cree un apodo denominado employee (en minúsculas) en esa tabla:

```
CREATE TABLE "employee"
  (EMP_NO       CHAR(6)       NOT NULL,
   ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

Si no se ha especificado la opción REMOTE_TABNAME y el *nombre-tabla* está delimitado, el nombre de tabla base remota será idéntico a *nombre-tabla*.

Ejemplo 21: Cree una tabla 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 de 98 bytes: 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (para columnas anulables) + 1 (para alineación) + 10 (para la cabecera). Con un tamaño de página de 4 KB (o 4096 bytes), después de contabilizar la actividad general de página, hay 4038 bytes disponibles, lo que constituye espacio suficiente para 41 registros por página. Si se

permiten 1 millón de registros de estudiantes, se necesitan (1 millón dividido por 41 registros por página) 24.391 páginas. Con dos páginas adicionales por actividad general de tabla, el número final de páginas de 4-KB que se asignan cuando se crea la tabla es 24.393.

Ejemplo 22: Cree una tabla denominada DEPARTMENT con una dependencia funcional que no tenga especificada ningún nombre de restricción.

```
CREATE TABLE DEPARTMENT
  (DEPTNO     SMALLINT      NOT NULL,
   DEPTNAME VARCHAR(36) NOT NULL,
   MGRNO      CHAR(6),
   ADMRDEPT   SMALLINT      NOT NULL,
   LOCATION   CHAR(30),
  CHECK (DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED)
```

Ejemplo 23: Cree una tabla con filas protegidas.

```
CREATE TABLE TOASTMASTERS
  (PERFORMANCE DB2SECURITYLABEL,
   POINTS      INTEGER,
   NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

Ejemplo 24: Cree una tabla con columnas protegidas.

```
CREATE TABLE TOASTMASTERS
  (PERFORMANCE CHAR(8),
   POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
   NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

Ejemplo 25: Cree una tabla con filas y columnas protegidas.

```
CREATE TABLE TOASTMASTERS
  (PERFORMANCE DB2SECURITYLABEL,
   POINTS      INTEGER COLUMN SECURED WITH CLUBPOSITION,
   NAME        VARCHAR(50))
  SECURITY POLICY CONTRIBUTIONS
```

Ejemplo 26: Los objetos grandes de una tabla particionada residen, por omisión, en el mismo espacio de tablas que los datos. Este uso por omisión se puede alterar temporalmente utilizando la cláusula LONG IN para especificar uno o varios espacios de tablas para los objetos grandes. Cree una tabla denominada DOCUMENTS cuyos datos de objetos grandes se almacenarán (de modo rotativo para cada partición de datos) en los espacios de tablas TBSP1 y TBSP2.

```
CREATE TABLE DOCUMENTS
  (ID          INTEGER,
   CONTENTS CLOB)
  LONG IN TBSP1, TBSP2
  PARTITION BY RANGE (ID)
    (STARTING 1 ENDING 1000
     EVERY 100)
```

De modo alternativo, puede emplear el formato largo de la sintaxis para identificar explícitamente un espacio de tablas grande para cada partición de datos. En este ejemplo, los datos CLOB de la primera partición de datos se colocan en LARGE_TBSP3, y los datos CLOB de las demás particiones de datos se reparten entre LARGE_TBSP1 y LARGE_TBSP2 de modo rotativo.

```
CREATE TABLE DOCUMENTS
  (ID          INTEGER,
   CONTENTS CLOB)
  LONG IN LARGE_TBSP1, LARGE_TBSP2
  PARTITION BY RANGE (ID)
```

```
    (STARTING 1 ENDING 100
     IN TBSP1 LONG IN LARGE_TBSP3,
     STARTING 101 ENDING 1000
     EVERY 100)
```

Ejemplo 27: Cree una tabla particionada denominada ACCESSNUMBERS con dos particiones de datos. La fila (10, NULL) debe colocarse en la primera partición y la fila (NULL, 100) debe colocarse en la segunda partición de datos (la última).

```
CREATE TABLE ACCESSNUMBERS
  (AREA INTEGER,
   EXCHANGE INTEGER)
  PARTITION BY RANGE (AREA NULLS LAST, EXCHANGE NULLS FIRST)
    (STARTING (1,1) ENDING (10,100),
     STARTING (11,1) ENDING (MAXVALUE,MAXVALUE))
```

Como los valores nulos de la segunda columna se clasificarían en primer lugar, la fila (11, NULL) se clasificaría debajo del límite inferior de la última partición de datos (11, 1); el intento de insertar esta fila devuelve un error. La fila (12, NULL) quedaría en la última partición de datos.

Ejemplo 28: Cree una tabla denominada RATIO con una sola partición de datos y la columna de particionamiento PERCENT.

```
CREATE TABLE RATIO
  (PERCENT INTEGER)
  PARTITION BY RANGE (PERCENT)
    (STARTING (MINVALUE) ENDING (MAXVALUE))
```

Esta definición de tabla permite insertar cualquier valor entero para la columna PERCENT. La definición siguiente de la tabla RATIO permite insertar en la columna PERCENT cualquier valor entero entre 1 y 100, ambos incluidos.

```
CREATE TABLE RATIO
  (PERCENT INTEGER)
  PARTITION BY RANGE (PERCENT)
    (STARTING 0 EXCLUSIVE ENDING 100 INCLUSIVE)
```

Ejemplo 29: Cree una tabla denominada MYDOCS con dos columnas: una es un identificador, y la otra almacena documentos XML.

```
CREATE TABLE MYDOCS
  (ID          INTEGER,
   DOC XML)
IN HLTBSPACE
```

Ejemplo 30: Cree una tabla denominada NOTES con cuatro columnas, una de ellas para almacenar notas XML.

```
CREATE TABLE NOTES
  (ID          INTEGER,
   DESCRIPTION VARCHAR(255),
   CREATED     TIMESTAMP,
   NOTE        XML)
```

Ejemplo 31: cree una tabla, EMP_INFO, que contenga un número de teléfono y una dirección para cada empleado. Incluya una columna ROW CHANGE TIMESTAMP en la tabla para seguir la modificación de la información del empleado.

```
CREATE TABLE EMP_INFO
  (EMPNO       CHAR(6) NOT NULL,
   EMP_INFOCHANGE  NOT NULL GENERATED ALWAYS
    FOR EACH ROW ON UPDATE
```

```
 AS ROW CHANGE TIMESTAMP,
EMP_ADDRESS     VARCHAR(300),
EMP_PHONENO     CHAR(4),
PRIMARY KEY (EMPNO) )
```

Ejemplo 32: Cree una tabla denominada BOOKS con cuatro columnas, una de ellas llamada DATE_ADDED, que inserta la indicación de fecha y hora (TIMESTAMP) por omisión.

```
CREATE TABLE BOOKS
  (ISBN_NUM   INTEGER,
   TITLE      VARCHAR(255),
   AUTHOR     VARCHAR(255),
   DATE_ADDED TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP)
```

CREATE TABLESPACE

La sentencia CREATE TABLESPACE define un nuevo espacio de tablas en la base de datos, asigna contenedores al espacio de tablas y registra la definición y los atributos del espacio de tablas en el catálogo.

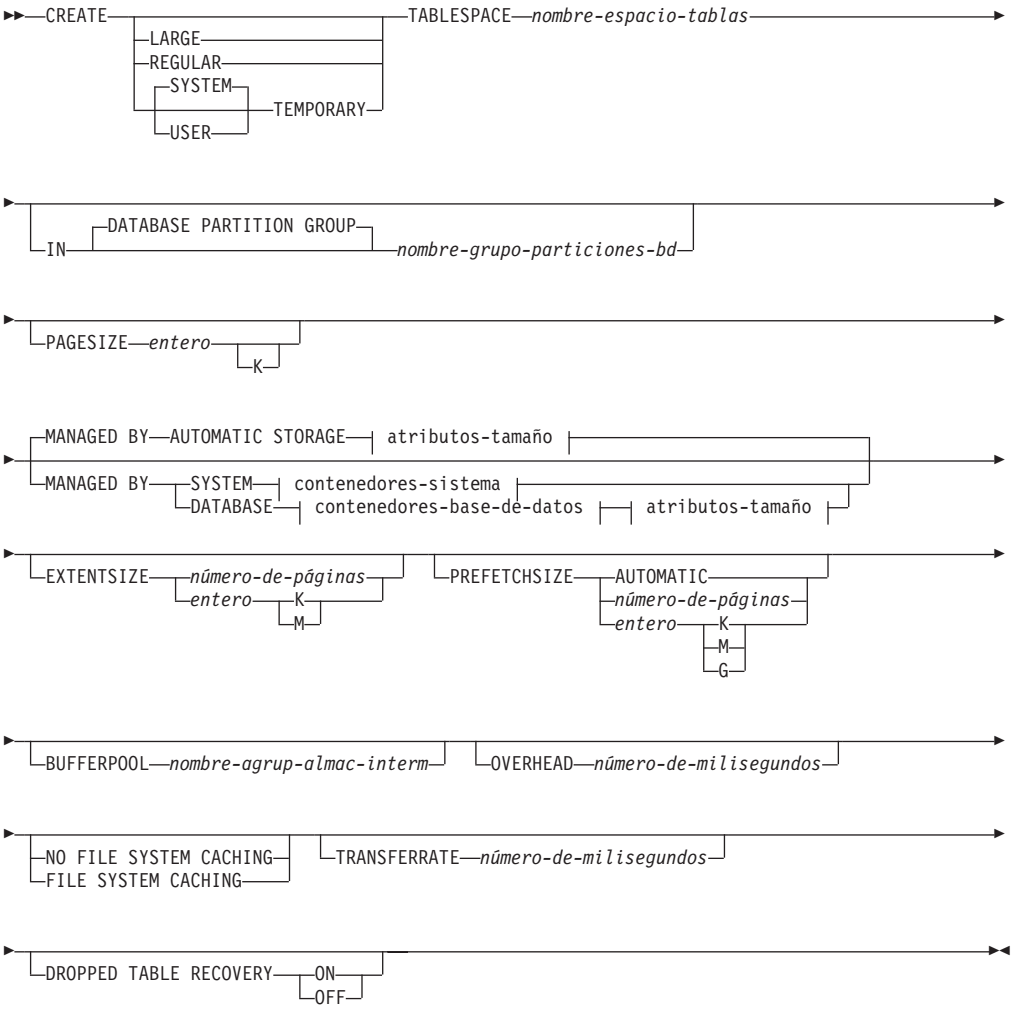
Invocación

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

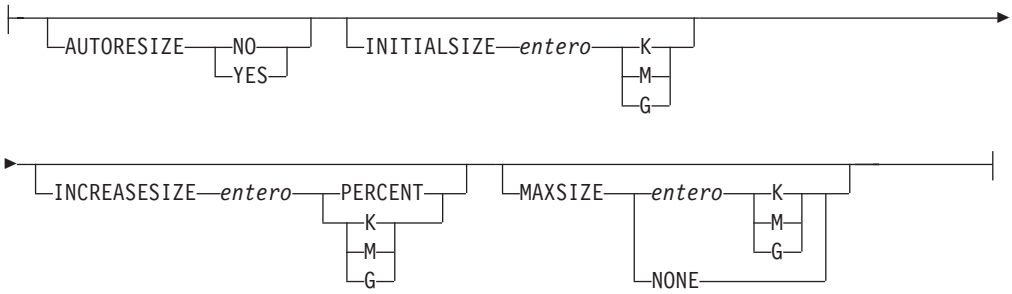
Autorización

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

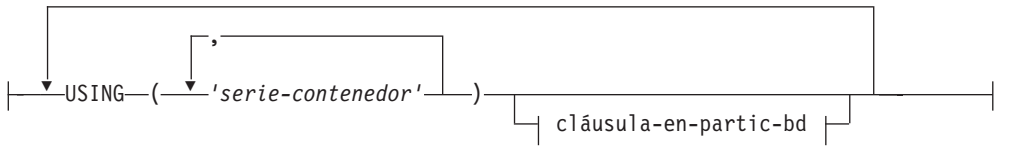
Sintaxis



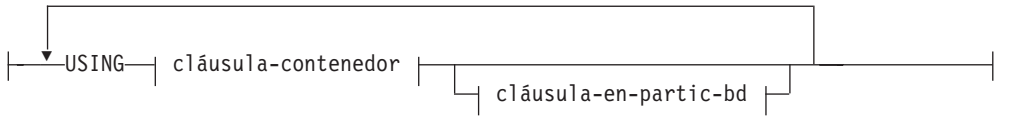
atributos-tamaño:



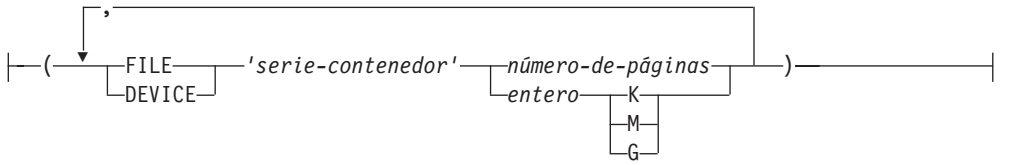
contenedores-sistema:



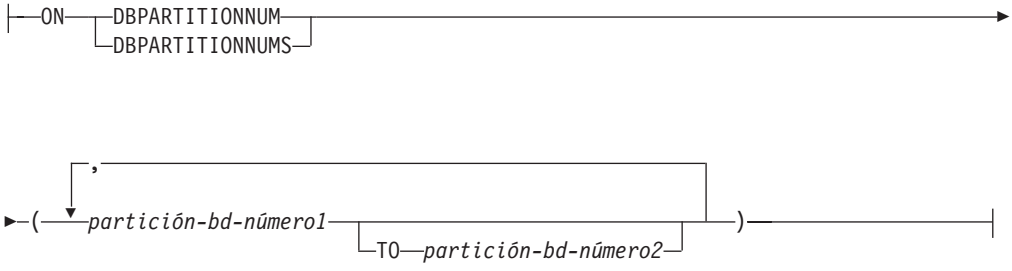
contenedores-basedatos:



cláusula-contenedor:



cláusula-en-partic-bd:



Descripción

LARGE, REGULAR, SYSTEM TEMPORARY o USER TEMPORARY
Especifica el tipo de espacio de tablas que se creará. Si no se especifica ningún tipo, la cláusula MANAGED BY determina el valor por omisión.

LARGE
Almacena todos los datos permanentes. Este tipo sólo se permite en espacios de tablas de espacio gestionado por la base de datos (DMS). También es el tipo por omisión de los espacios de tablas DMS si no se especifica ningún tipo. Si una tabla se coloca en un espacio de tablas grande:

- La tabla puede ser mayor que una tabla de un espacio de tablas normal. Para obtener detalles sobre los límites de espacio de tablas y de tabla, consulte el apartado "Límites de SQL y XML".
- La tabla puede dar soporte a más de 255 filas por página de datos, lo que puede mejorar la utilización del espacio de las páginas de datos.
- Los índices definidos en la tabla requerirán una entrada de dos bytes por fila adicional, a diferencia de los índices definidos en una tabla que reside en un espacio de tablas normal.

REGULAR
Almacena todos los datos permanentes. Este tipo se aplica a espacios de tablas DMS y SMS. Éste es el único tipo permitido en los espacios de tablas SMS y también es el tipo por omisión para espacios de tablas SMS si no se ha especificado ningún tipo.

SYSTEM TEMPORARY
Almacena temporalmente tablas, áreas de trabajo utilizadas por el gestor de bases de datos para llevar a cabo operaciones como clasificaciones o uniones. Una base de datos siempre debe tener al menos un espacio de tablas SYSTEM TEMPORARY, puesto que las tablas temporales sólo se pueden almacenar en un espacio de tablas de este tipo. Al crear una base de datos, se crea automáticamente un espacio de tablas temporal.

USER TEMPORARY
Almacena tablas temporales globales declaradas. Cuando se crea una base de datos, no existen espacios de tablas temporales de usuario. Para permitir la definición de tablas temporales declaradas, debe crearse al menos un espacio de tablas temporal de usuario con los privilegios USE adecuados.

nombre-espacio-tablas
Nombra el espacio de tablas. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-espacio-tablas* no debe identificar un espacio de tablas que ya exista en el catálogo (SQLSTATE 42710). El *nombre-espacio-tablas* no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

IN DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*
Especifica el grupo de particiones de base de datos para el espacio de tablas. El grupo de particiones de base de datos debe existir. El único grupo de particiones de base de datos que se puede especificar al crear un espacio de tablas SYSTEM TEMPORARY es IBMTEMPGROUP. Las palabras clave DATABASE PARTITION GROUP son opcionales.

Si no se especifica el grupo de particiones de base de datos, se utiliza el grupo de particiones de base de datos por omisión (IBMDEFAULTGROUP) para los espacios de tablas REGULAR, LARGE y USER TEMPORARY. Para los espacios de tablas SYSTEM TEMPORARY, se utiliza el grupo de particiones de base de datos por omisión IBMTEMPGROUP.

PAGESIZE *entero* **[K]**
Define el tamaño de las páginas utilizadas para el espacio de tablas. Los valores válidos para el *entero* sin el sufijo K son 4 096, 8 192, 16 384 o 32 768.

Los valores válidos para el *entero* con el sufijo K son 4, 8, 16 o 32. Se permite cualquier número de espacios entre *entero* y K, incluso ningún espacio. Se produce un error si el tamaño de página no es uno de estos valores (SQLSTATE 428DE) o si el tamaño de página no es el mismo que el de la agrupación de almacenamientos intermedios asociada al espacio de tablas (SQLSTATE 428CB).

El valor por omisión se proporciona mediante el parámetro de configuración de base de datos **tamaño-página**, que se establece cuando se crea la base de datos.

MANAGED BY AUTOMATIC STORAGE
Especifica que el espacio de tablas debe ser un espacio de tablas de almacenamiento automático. Si el almacenamiento automático no se define para la base de datos, se devuelve un error (SQLSTATE 55060).

Un espacio de tablas de almacenamiento automático se crea como un espacio de tablas gestionado por el sistema (SMS) o como un espacio de tablas gestionado por la base de datos (DMS). Cuando se elige DMS y no se especifica el tipo del espacio de tablas, el comportamiento por omisión consiste en crear un espacio de tablas grande. Con un espacio de tablas de almacenamiento automático, el gestor de bases de datos determina qué contenedores deben asignarse al espacio de tablas, basándose en las vías de acceso de almacenamiento asociadas a la base de datos.

atributos-tamaño
Especifique los atributos de tamaño para un espacio de tablas de almacenamiento automático o para un espacio de tablas DMS que no sea de almacenamiento automático. Los espacios de tablas SMS no permiten cambiar el tamaño automáticamente.

AUTORESIZE
Especifica si debe habilitarse o no la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático. Los espacios de tablas con dicha posibilidad aumentan de tamaño automáticamente cuando se llenan. El valor por omisión es NO para los espacios de tablas DMS y YES para los espacios de tablas de almacenamiento automático.

NO
Especifica que debe inhabilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

YES
Especifica que debe habilitarse la posibilidad de cambiar automáticamente el tamaño de un espacio de tablas DMS o de un espacio de tablas de almacenamiento automático.

INITIALSIZE *entero* **K | M | G**
Especifica el tamaño inicial, por partición de base de datos, de un espacio de tablas de almacenamiento automático. Esta opción sólo es válida para espacios de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un tamaño coherente en los contenedores del espacio de tablas. Además, si el espacio de tablas se puede cambiar automáticamente y el tamaño inicial no es lo suficientemente grande para contener metadatos que se deben añadir al nuevo espacio de tablas, DB2 ampliará el espacio de tablas mediante

INCREASESIZE hasta que éste sea suficiente. Si el espacio de tablas permite cambiar el tamaño automáticamente, pero no se ha especificado la cláusula INITIALSIZE, el gestor de bases de datos determinará un valor apropiado.

INCREASESIZE *entero* **PERCENT o INCREASESIZE** *entero* **K | M | G**
Especifica la cantidad, por partición de base de datos, en la que aumentará automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente, cuando se llene el espacio de tablas y se haya efectuado una petición de espacio. El valor entero debe ir seguido de:

- PERCENT para especificar la cantidad como porcentaje del tamaño de espacio de tablas en el momento en el que se efectúe una petición de espacio. Si se especifica PERCENT, el valor entero debe estar comprendido entre 0 y 100 (SQLSTATE 42615).
- K (de kilobytes), M (de megabytes) o G (de gigabytes) para especificar la cantidad en bytes.

Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior o superior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas. Si el espacio de tablas permite cambiar el tamaño automáticamente, pero no se ha especificado la cláusula INCREASESIZE, el gestor de bases de datos determinará un valor apropiado.

MAXSIZE *entero* **K | M | G o MAXSIZE NONE**
Especifica el tamaño máximo hasta el cual puede aumentar automáticamente un espacio de tablas habilitado para cambiar de tamaño automáticamente. Si el espacio de tablas permite cambiar el tamaño automáticamente, pero no se ha especificado la cláusula MAXSIZE, el valor por omisión es NONE.

entero
Especifica un límite fijo sobre el tamaño, por partición de base de datos, hasta el cual puede aumentar automáticamente un espacio de tablas DMS o un espacio de tablas de almacenamiento automático. El valor entero debe ir seguido de K (de kilobytes), M (de megabytes) o G (de gigabytes). Tenga en cuenta que el valor real utilizado puede ser ligeramente inferior al especificado, ya que el gestor de bases de datos procura mantener un incremento coherente en los contenedores del espacio de tablas.

NONE
Especifica que debe permitirse incrementar el espacio de tablas hasta la capacidad del sistema de archivos o hasta el tamaño máximo del espacio de tablas (descrito en "Límites de SQL y XML").

MANAGED BY SYSTEM
Especifica que el espacio de tablas debe ser un espacio de tablas SMS. Si no está especificado el tipo de espacio de tablas, el comportamiento por omisión es crear un espacio de tablas normal.

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, es relativo al directorio de base de datos y puede ser un alias de nombre de vía de acceso (un enlace simbólico en sistemas UNIX) a almacenamiento que no esté físicamente asociado con el directorio de base de datos. Por ejemplo, *dbdir*/work/c1 puede ser un enlace simbólico a un sistema de archivos separado.

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*, no debe contener ningún archivo ni ningún subdirectorio (SQLSTATE 428B2).

El formato de *serie-contenedor* es dependiente del sistema operativo. En sistemas operativos Windows, un nombre de vía de acceso de directorio absoluto empieza con una letra de unidad y dos puntos (:); en sistemas UNIX, un nombre de vía de acceso absoluto empieza con una barra inclinada (/). Un nombre de vía de acceso relativo en ninguna plataforma empieza por un carácter que dependa de un sistema operativo.

En la actualidad sólo se da soporte a recursos remotos (como, por ejemplo, unidades redirigidas a LAN o sistemas de archivos montados en NFS) cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o S4100, o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que 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). Un sistema de archivos montado en NFS en AIX se debe montar en modalidad ininterrumpida utilizando la opción **-o nointr**.

cláusula-en-partic-bd
Especifica las particiones de base de datos en las que se crean los contenedores en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones de base de datos en el grupo de particiones de base de datos que no están especificadas explícitamente en ninguna otra *cláusula-en-particiones-bd*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, si la *cláusula-en-particiones-base-de-datos* no está especificada, los contenedores también se crearán en todas las particiones de base de datos nuevas añadidas a la base de datos.

MANAGED BY DATABASE
Especifica que el espacio de tablas debe ser un espacio de tablas DMS. Si no está especificado el tipo de espacio de tablas, el comportamiento por omisión es crear un espacio de tablas grande.

contenedores-basedatos
Especifica los contenedores de un espacio de tablas DMS.

USING
Introduce una cláusula-contenedor.

cláusula-contenedor
Especifica los contenedores de un espacio de tablas DMS.

(FILE | DEVICE *'serie-contenedor' número-de-páginas,...***)**
Para un espacio de tablas DMS, identifica uno o varios contenedores que pertenecerán al espacio de tablas y donde se almacenarán los

datos del espacio de tablas. Se especifican el tipo del contenedor (FILE o DEVICE) y su tamaño (en páginas de PAGESIZE). El tamaño también puede especificarse como un valor entero seguido de K (para kilobytes), M (para megabytes) o G (para gigabytes). Si se especifica de esta manera, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el número de páginas para el contenedor. Se puede especificar una mezcla de contenedores FILE y DEVICE. La *serie-contenedor* no puede sobrepasar los 254 bytes de longitud.

Para un contenedor FILE, *serie-contenedor* debe ser un nombre de archivo absoluto o relativo. El nombre de archivo, si no es absoluto, será relativo al directorio de la base de datos. Si algún componente del nombre 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 sobregraba, y si es menor de lo especificado, se amplía. El archivo no se truncará si es más grande de lo que se ha especificado.

Para un contenedor DEVICE, *serie-contenedor* debe ser un nombre de dispositivo. El dispositivo ya debe existir.

Todos los contenedores deben ser únicos en todas las bases de datos. Un contenedor sólo puede pertenecer a un espacio de tablas. El tamaño de los contenedores puede variar; sin embargo, el rendimiento óptimo se consigue cuando todos los contenedores tienen el mismo tamaño. El formato exacto *serie-contenedor* depende del sistema operativo.

En la actualidad sólo se da soporte a recursos remotos (como, por ejemplo, unidades redirigidas a LAN o sistemas de archivos montados en NFS) cuando se utiliza Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200 o S4100, o NEC Storage NS Series con un servidor Windows DB2. Tenga en cuenta que 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-partic-bd
Especifica las particiones de base de datos en las que se crean los contenedores en una base de datos particionada. Si no se especifica esta cláusula, los contenedores se crean en las particiones de base de datos en el grupo de particiones de base de datos que no están especificadas explícitamente en ninguna otra *cláusula-en-particiones-base-de-datos*. Para un espacio de tablas SYSTEM TEMPORARY definido en el grupo de particiones de base de datos IBMTEMPGROUP, si la *cláusula-en-particiones-base-de-datos* no está especificada, los contenedores también se crearán en todas las particiones de base de datos nuevas añadidas a la base de datos.

cláusula-en-partic-bd
Especifica las particiones de base de datos en las que se crean contenedores en una base de datos particionada.

ON DBPARTITIONNUMS
Palabras clave que indican que las particiones individuales de la base de datos están especificadas. DBPARTITIONNUM es sinónimo de DBPARTITIONNUMS.

part-bd-núm1
Especifique un número de partición de base de datos.

TO *partición-bd-número2*
Especifique un rango de números de partición de base de datos. El valor de *partición-bd-número2* debe ser mayor que o igual al valor de *partición-bd-número1* (SQLSTATE 428A9). Los contenedores se crean en cada una de las particiones de base de datos entre los valores especificados e incluidos éstos. Una partición de base de datos especificada debe encontrarse en el grupo de particiones de base de datos del espacio de tablas.

La partición de base de datos especificada por el número y cada una de las particiones de base de datos que hay dentro de un rango especificado deben existir en el grupo de particiones de base de datos del espacio de tablas (SQLSTATE 42729). Un número de partición de base de datos sólo puede aparecer explícitamente o dentro de un rango en exactamente una *cláusula-en-particiones-base-de-datos* de la sentencia (SQLSTATE 42613).

EXTENTSIZE *número-de-páginas*
Especifica el número de páginas de PAGESIZE que se grabarán en un contenedor antes de pasar al siguiente contenedor. El valor del tamaño de extensión también puede especificarse como un valor entero seguido de K (para kilobytes) o de M (para megabytes). Si se especifica de 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 la base de datos **dft_extent_sz**, que tiene un rango válido de 2-256 páginas.

PREFETCHSIZE
Especifica que deben leerse los datos necesarios para una consulta antes de que la consulta haga referencia a los mismos, de modo que la consulta no deba esperar a que se lleve a cabo la E/S.

El valor por omisión lo proporciona el parámetro de configuración de la base de datos **dft_prefetch_sz**.

AUTOMATIC
Especifica que el tamaño de captación previa de un espacio de tablas debe actualizarse automáticamente; es decir, el tamaño de captación previa se gestionará a través del gestor de bases de datos DB2.

DB2 actualizará el tamaño de captación previa automáticamente siempre que el número de contenedores de un espacio de tablas cambie (seguido por una ejecución satisfactoria de una sentencia ALTER TABLESPACE que añada o desactive uno o más contenedores). El tamaño de captación previa se actualiza al iniciarse la base de datos.

número-de-páginas
Especifica el número de páginas PAGESIZE del espacio de tablas que se leerán cuando se realice la captación previa de datos. El valor 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 este modo, el nivel mínimo del número de bytes dividido por el tamaño de página se utiliza para determinar el valor de número de páginas para el tamaño de la captación previa.

BUFFERPOOL *nombre-agrup-almac-interm*
El nombre de la agrupación de almacenamientos intermedios utilizado para las tablas de este espacio de tablas. La agrupación de almacenamientos intermedios debe existir (SQLSTATE 42704). Si no se especifica, se utiliza la agrupación de almacenamientos intermedios por omisión (IBMDEFAULTBP). El tamaño de página de la agrupación de almacenamientos intermedios debe coincidir con el tamaño de página especificado (o tomado por omisión) para el espacio de tablas (SQLSTATE 428CB). Debe haberse definido el grupo de particiones de base de datos del espacio de tablas para la agrupación de almacenamientos intermedios (SQLSTATE 42735).

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

Para una base de datos creada en la versión 9 o posterior, la actividad general del controlador de E/S, la búsqueda de disco y el tiempo de latencia por omisión es de 7,5 milisegundos. Para una base de datos migrada desde una versión anterior de DB2 a la versión 9 o posterior, el valor por omisión es de 12,67 milisegundos.

FILE SYSTEM CACHING o NO FILE SYSTEM CACHING
Especifica si las operaciones de E/S se deben almacenar en antememoria o no a nivel del sistema de archivos. Si no se especifica ninguna opción, el valor por omisión es:

- FILE SYSTEM CACHING para JFS en AIX, Linux System z, todos los sistemas de archivos no VxFS en Solaris, HP-UX, archivos de espacio de tablas temporal SMS en todas las plataformas y todos los datos grandes y LOB
- NO FILE SYSTEM CACHING en todas las otras plataformas y tipos de sistema de archivos

FILE SYSTEM CACHING
Especifica que todas las operaciones de E/S en el espacio de tablas de destino se deben almacenar en antememoria en el nivel del sistema de archivos.

NO FILE SYSTEM CACHING
Especifica que todas las operaciones de E/S no se deben almacenar en antememoria en el nivel del sistema de archivos.

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

Para una base de datos creada en la versión 9 o posterior, el tiempo por omisión para leer una página de la memoria es de 0,06 milisegundos. Para una base de datos migrada desde una versión anterior de DB2 a la versión 9 o posterior, el valor por omisión es de 0,18 milisegundos.

DROPPED TABLE RECOVERY
Indica si las tablas descartadas en el espacio de tablas especificado se pueden recuperar mediante la opción **RECOVER DROPPED TABLE** del mandato ROLLFORWARD DATABASE. Esta cláusula sólo se puede especificar para un espacio de tablas normal o más grande (SQLSTATE 42613).

ON
Especifica que las tablas descartadas pueden recuperarse. Éste ha sido el valor por omisión desde la versión 8.

OFF
Especifica que las tablas descartadas no pueden recuperarse. Éste es el valor por omisión en la versión 7.

Normas

- Si el almacenamiento automático no se define para la base de datos, se devuelve un error (SQLSTATE 55060).
- La cláusula INITIALSIZE no puede especificarse con la cláusula MANAGED BY SYSTEM o MANAGED BY DATABASE (SQLSTATE 42601).
- La cláusula AUTORESIZE, INCREASESIZE o MAXSIZE no puede especificarse con la cláusula MANAGED BY SYSTEM (SQLSTATE 42601).
- La cláusula AUTORESIZE, INITIALSIZE, INCREASESIZE o MAXSIZE no puede especificarse para la creación de un espacio de tablas de almacenamiento automático temporal (SQLSTATE 42601).
- La cláusula INCREASESIZE o MAXSIZE no puede especificarse si el espacio de tablas no permite cambiar el tamaño automáticamente (SQLSTATE 42601).
- AUTORESIZE no puede habilitarse para los espacios de tablas DMS que están definidos para utilizar contenedores de dispositivo sin procesar (SQLSTATE 42601).
- Un espacio de tablas debe ser, inicialmente, lo bastante grande como para contener cinco extensiones (SQLSTATE 57011).
- El tamaño máximo de un espacio de tablas debe ser superior a su tamaño inicial (SQLSTATE 560B0).
- Las operaciones con contenedores (ADD, EXTEND, RESIZE, DROP o BEGIN NEW STRIPE SET) no pueden realizarse en espacios de tablas de almacenamiento automático porque el gestor de bases de datos controla la gestión del espacio de dichos espacios de tabla (SQLSTATE 42858).
- Cada definición de contenedor requiere 53 bytes más el número de bytes necesario para almacenar el nombre de contenedor. La longitud combinada de todos los nombres de contenedores para el espacio de tablas no puede superar los 20.480 bytes (SQLSTATE 54034).
- Para una base de datos particionada, si más de una partición de base de datos reside en el mismo nodo físico, no se puede especificar el mismo dispositivo ni la misma vía de acceso para más de una partición de base de datos (SQLSTATE 42730). En este caso, especifique una *serie-contenedor* para cada partición de base de datos o utilice un nombre de vía de acceso relativo.

- ***Límites de tamaño de contenedor:*** En espacios de tablas DMS, un contenedor debe tener al menos el doble del tamaño de extensión en páginas de longitud (SQLSTATE 54039). El tamaño máximo de un contenedor depende del sistema operativo.

Notas

- La elección entre un espacio de tablas gestionado por la base de datos y un espacio de tablas gestionado por el sistema es una elección fundamental que implica concesiones.
- Si existe más de un espacio de tablas TEMPORARY en la base de datos, éstos se utilizan de modo rotativo para equilibrar el uso.
- Al crear contenedor SMS o DMS, puede especificar una expresión de la partición de base de datos para la sintaxis de la serie. Normalmente, se especifica la expresión de la partición de base de datos cuando se utiliza varias particiones de base de datos lógicas en el sistema de la base de datos particionada. De esta forma, se garantiza que los nombres del contenedor sean exclusivos en los servidores de la partición de base de datos. Si se especifica la expresión, el número de partición de base de datos forma parte del nombre del contenedor o, si se especifican argumentos adicionales, el resultado del argumento forma parte del nombre del contenedor.

  Se utiliza el argumento " $N" ([blanco]$N) 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 se puede utilizar de una de las formas siguientes.

Tabla 23. 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] % representa el operador del 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
  (archivo '/DB2/contenedores/TS2/contenedor $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/ $N2000 /dmscont'

se crean como:
  '/dbdir/node5/cont1'
  '/1005/file1'
  '5/container'
  '/dir/2000/dmscont'
```

- Un espacio de tablas de almacenamiento automático se crea como un espacio de tablas SMS o como un espacio de tablas DMS. DMS se selecciona para espacios de tablas grandes y normales, y SMS se selecciona para espacios de tablas temporales. Tenga en cuenta que no puede contar con este comportamiento, dado que es posible que cambie en un release posterior. Cuando se elige DMS y no se especifica el tipo del espacio de tablas, el comportamiento por omisión consiste en crear un espacio de tablas grande.
- La creación de un espacio de tablas de almacenamiento automático no incluye definiciones de contenedor. El gestor de bases de datos determina automáticamente la ubicación y el tamaño, si corresponde, de los contenedores basándose en las vías de acceso de almacenamiento asociadas a la base de datos. El gestor de bases de datos intentará ampliar los espacios de tablas grandes y normales cuanto sea necesario, siempre y cuando no se alcance el tamaño máximo. Esto puede implicar la ampliación de los contenedores existentes o la adición de contenedores a un nuevo conjunto de bandas. Cada vez que se activa la base de datos, el gestor de bases de datos reconfigura automáticamente el número de contenedores y su ubicación para los espacios de tablas temporales que no se encuentren en estado anómalo.
- Un espacio de tablas de almacenamiento automático grande o normal no utilizará nuevas vías de acceso a almacenamiento (consulte la descripción de la sentencia ALTER DATABASE) hasta que no se agote el espacio en una de las vías de acceso a almacenamiento existentes que el espacio de tablas utiliza. Los espacios de tablas de almacenamiento automático temporales sólo pueden utilizar las nuevas vías de acceso de almacenamiento una vez que la base de datos se haya desactivado y, luego, reactivado.
- ***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

Ejemplos

Ejemplo 1: Cree un espacio de tabla DMS grande en un sistema UNIX utilizando tres dispositivos de 10.000 páginas de 4 K cada uno. Especifique sus características de E/S.

```
CREATE TABLESPACE PAYROLL
  MANAGED BY DATABASE
  USING (DEVICE'/dev/rhdisk6' 10000,
    DEVICE '/dev/rhdisk7' 10000,
    DEVICE '/dev/rhdisk8' 10000)
  OVERHEAD 12.67
  TRANSFERRATE 0.18
```

Ejemplo 2: Cree un espacio de tablas SMS normal en Windows mediante tres directorios en tres unidades distintas, con un tamaño de extensión de 64 páginas y un tamaño de captación previa de 32 páginas.

```
CREATE TABLESPACE ACCOUNTING
  MANAGED BY SYSTEM
  USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
  EXTENTSIZE 64
  PREFETCHSIZE 32
```

Ejemplo 3: Cree un espacio de tabla DMS temporal de sistema en un sistema UNIX utilizando dos archivos de 50.000 páginas cada uno y un tamaño de extensión de 256 páginas.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY DATABASE
  USING (FILE 'dbtmp/tempspace2.f1' 50000,
    FILE 'dbtmp/tempspace2.f2' 50000)
  EXTENTSIZE 256
```

Ejemplo 4: Cree un espacio de tabla DMS grande en un grupo de particiones de base de datos ODDNODEGROUP (particiones 1, 3 y 5 de base de datos) en un sistema UNIX. Utilice el dispositivo `/dev/rhdisk0` para 10.000 páginas de 4 K en cada partición de base de datos. Especifique un dispositivo específico de particiones de base de datos con 40.000 páginas de 4K para cada partición de base de datos.

```
CREATE TABLESPACE PLANS
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
  ON DBPARTITIONNUM (1)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
  ON DBPARTITIONNUM (3)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
  ON DBPARTITIONNUM (5)
```

Ejemplo 5: Cree un espacio de tablas de almacenamiento automático grande denominado DATATS que permita que el sistema realice todas las decisiones con respecto al tamaño y al crecimiento del espacio de tablas.

```
CREATE TABLESPACE DATATS
```

o

```
CREATE TABLESPACE DATATS
  MANAGED BY AUTOMATIC STORAGE
```

Ejemplo 6: cree un espacio de tablas de almacenamiento automático temporal llamado TEMPDATA.

```
CREATE TEMPORARY TABLESPACE TEMPDATA
```

o

```
CREATE TEMPORARY TABLESPACE TEMPDATA
  MANAGED BY AUTOMATIC STORAGE
```

Ejemplo 7: cree un espacio de tala de almacenamiento automático grande llamado USERSPACE3 con un tamaño inicial de 100 megabytes y un tamaño máximo de 1 gigabyte.

```
CREATE TABLESPACE USERSPACE3
  INITIALSIZE 100 M
  MAXSIZE 1 G
```

Ejemplo 8: Cree un espacio de tablas de almacenamiento automático grande denominado LARGEDATA con un índice de incremento del 10 por ciento (es decir, su tamaño total aumenta en un 10 por ciento cada vez que cambia automáticamente) y un tamaño máximo de 512 megabytes. En lugar de especificar la cláusula INITIALSIZE, permita que el gestor de bases de datos determine un tamaño inicial apropiado para el espacio de tablas.

```
CREATE LARGE TABLESPACE LARGEDATA
  INCREASESIZE 10 PERCENT
  MAXSIZE 512 M
```

Ejemplo 9: cree un espacio de tablas DMS grande llamado USERSPACE4 con dos contenedores de archivo (cada contenedor tiene 1 megabyte de tamaño), un índice de incremento de 2 megabytes y un tamaño máximo de 100 megabytes.

```
CREATE TABLESPACE USERSPACE4
  MANAGED BY DATABASE USING (FILE '/db2/file1' 1 M, FILE '/db2/file2' 1 M)
  AUTORESIZE YES
  INCREASESIZE 2 M
  MAXSIZE 100 M
```

Ejemplo 10: cree espacios de tablas DMS grandes utilizando dispositivos RAW en un sistema operativo Windows.

- Para especificar unidades físicas completas, utilice el formato \\.*unidad-física*:

  ```
  CREATE TABLESPACE TS1
    MANAGED BY DATABASE USING (DEVICE '\\.\PhysicalDrive5' 10000,
      DEVICE '\\.\PhysicalDrive6' 10000)
  ```

- Para especificar particiones lógicas mediante letras de unidad:

  ```
  CREATE TABLESPACE TS2
    MANAGED BY DATABASE USING (DEVICE '\\.\G:' 10000,
      DEVICE '\\.\H:' 10000)
  ```

- Para especificar particiones lógicas mediante identificadores exclusivos globales (GUID) del volumen, emplee el programa de utilidad db2listvolumes para recuperar el GUID del volumen de todas las particiones locales, a continuación, copie el GUID de la partición lógica que desee incluir en la cláusula del contenedor de espacios de tablas:

  ```
  CREATE TABLESPACE TS3
    MANAGED BY DATABASE USING (
      DEVICE '\\?\Volume{2ca6a0c1-8542-11d8-9734-00096b5322d2}\' 20000M)
  ```

Es preferible que utilice los GUID del volumen en lugar del formato de letra de unidad si tiene más particiones que letras de unidad disponibles en la máquina.

- Para especificar particiones lógicas mediante puntos de unión (o puntos de montaje de volúmenes), monte la partición RAW en otro volumen con formato NTFS como un punto de unión y, a continuación, especifique la vía de acceso al punto de unión en el volumen NTFS como la vía de acceso al contenedor. Por ejemplo:

```
CREATE TABLESPACE TS4
  MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,
    DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

  DB2 primero consulta la partición para ver si ésta incluye un sistema de archivos; en caso afirmativo, la partición no se trata como un dispositivo RAW y DB2 lleva a cabo operaciones de E/S del sistema de archivos en la partición.

CREATE THRESHOLD

La sentencia CREATE THRESHOLD define un umbral.

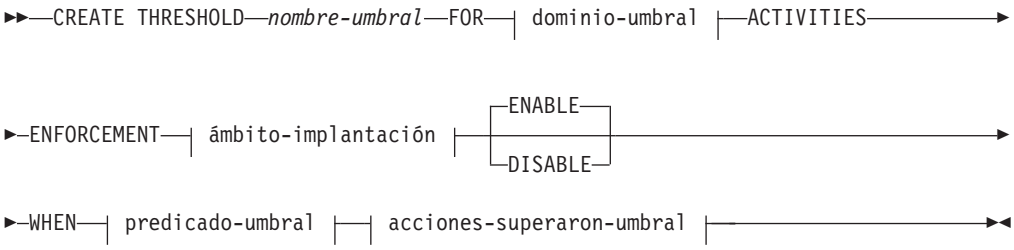
Invocación

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

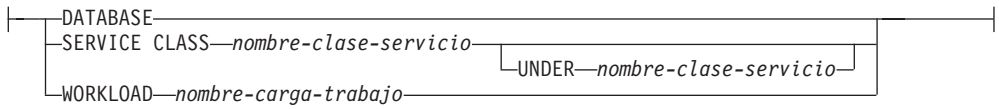
Autorización

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

Sintaxis

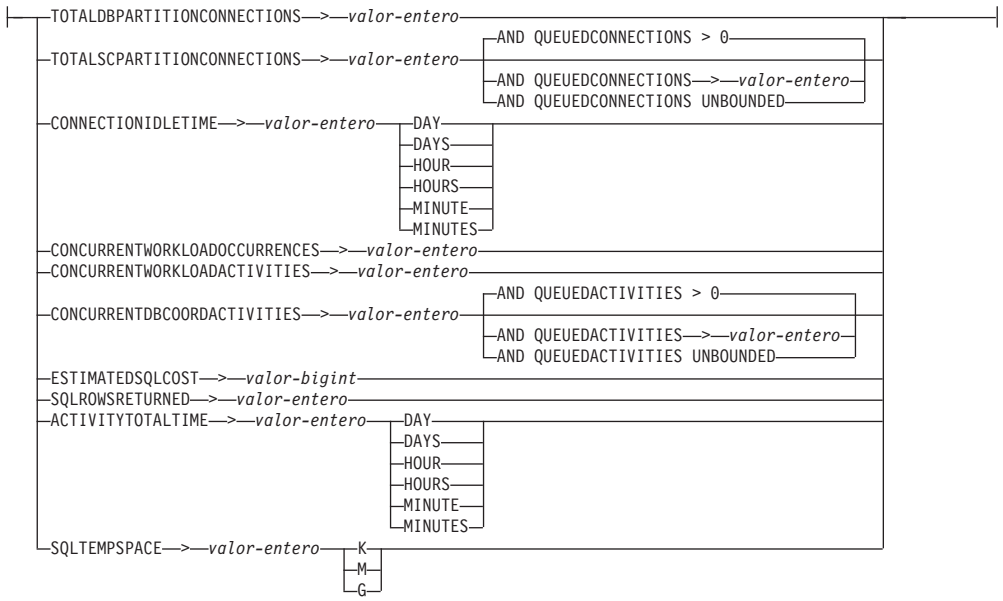


dominio-umbral:

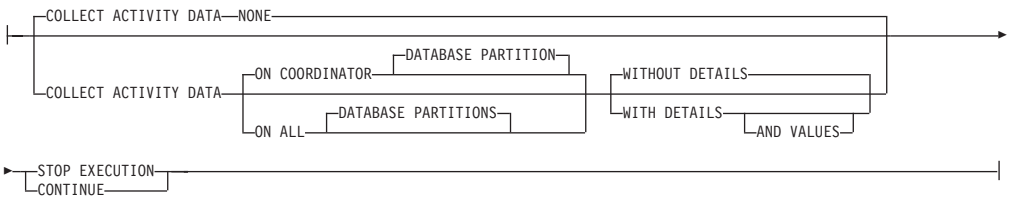


ámbito-implantación:

predicado-umbral:



acciones-superaron-umbral:



Descripción

nombre-umbral
Da nombre al umbral. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-umbral* no debe identificar una carga de trabajo que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

FOR *dominio-umbral* **ACTIVITIES**
Especifica el dominio de definición del umbral.

DATABASE
Este umbral se aplica a cualquier actividad de la base de datos.

SERVICE CLASS *nombre-clase-servicio*
Este umbral se aplica a las actividades que se ejecutan en la clase de servicio *nombre-clase-servicio*. Si no se especifica UNDER, *nombre-clase-servicio* debe identificar una superclase de servicio existente (SQLSTATE 42704). Si se especifica UNDER, *nombre-clase-servicio* debe identificar una subclase de servicio específica de la superclase de servicio especificada como consecuencia de la palabra clave UNDER (SQLSTATE 42704).

UNDER *nombre-clase-servicio*
Especifica una superclase de servicio. El *nombre-clase-servicio* debe identificar una superclase de servicio existente (SQLSTATE 42704).

WORKLOAD *nombre-carga-trabajo*
Este umbral se aplica a la carga de trabajo especificada. El *nombre-carga-trabajo* debe identificar una carga de trabajo existente (SQLSTATE 42704).

ENFORCEMENT *ámbito-implantación*
El ámbito de implantación del umbral.

DATABASE
El umbral se implanta en todas las particiones del dominio de definición; es decir, todas las particiones de base de datos de la base de datos y todas las particiones de base de datos de la clase de servicio.

DATABASE PARTITION
El umbral se implanta con base a las particiones de la base de datos. No hay ninguna coordinación entre las particiones de base de datos para implantar el umbral.

WORKLOAD OCCURRENCE
El umbral sólo se implanta en una aparición de carga de trabajo. Dos apariciones de carga de trabajo que se ejecuten simultáneamente en la misma partición de la base de datos dispondrán cada una de ellas de su propio número de ejecución para este umbral.

ENABLE o DISABLE
Especifica si el umbral está habilitado o no para que lo utilice el gestor de la base de datos.

ENABLE
El gestor de base de datos utiliza el umbral para restringir la ejecución de actividades de base de datos.

DISABLE
El gestor de base de datos no utiliza el umbral para restringir la ejecución de actividades de base de datos.

WHEN *predicado-umbral*
Especifica la condición del umbral.

TOTALDBPARTITIONCONNECTIONS > *valor-entero*
Esta condición define una vinculación superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en una partición de base de datos. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá la conexión de cualquier conexión de coordinador nueva. Proseguirán todas las conexiones en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser DATABASE y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037).

TOTALSCPARTITIONCONNECTIONS > *valor-entero*
Esta condición define una vinculación superior sobre el número de conexiones de coordinador que pueden ejecutarse simultáneamente en una partición de base de datos de una superclase de servicio específica. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá que cualquier conexión nueva se una a la clase de servicio. Proseguirán todas las conexiones en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser SERVICE SUPERCLASS y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037).

AND QUEUEDCONNECTIONS > *valor-entero* **o AND QUEUEDCONNECTIONS UNBOUNDED**
Especifica un tamaño de cola para el momento en que se supere el número máximo de conexiones de coordinador. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna conexión de coordinador. Especificar UNBOUNDED pondrá en cola todas las conexiones que superen el número máximo especificado de conexiones de coordinador y las *acciones-umbral-superado* nunca se ejecutarán. El valor por omisión es cero.

CONNECTIONIDLETIME > *valor-entero* **(DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)**
Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá a una conexión permanecer desocupada. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. El dominio de definición para esta condición debe ser DATABASE o SERVICE SUPERCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037). Esta condición se implanta lógicamente en la partición de base de datos del coordinador.

Si se especifica la acción STOP EXECUTION con umbrales CONNECTIONIDLETIME, la conexión para la aplicación se descarta cuando se supera el umbral. Cualquier intento posterior por parte de la aplicación para acceder al servidor de datos no recibirá SQLSTATE 5U026.

Puesto que la granularidad mínima para este umbral es de cinco minutos, los valores especificados se redondean al múltiplo de cinco minutos diferente a cero que más se aproxime a cero. En los casos en los que el redondeo genere un desbordamiento o un subdesbordamiento, se seleccionará el valor más próximo del rango de entero.

El valor máximo par este umbral es de 2 147 400 segundos. Cualquier valor especificado que tenga un número de segundos equivalente superior a 2 147 400 segundos se establecerá en este número de segundos.

Es posible que los cambios en la variable de registro DB2CHECKCLIENTINTERVAL que ocasionan comprobaciones menos frecuentes afecten a la granularidad de este umbral.

CONCURRENTWORKLOADOCCURRENCES > *valor-entero*
Esta condición define una vinculación superior sobre el número de apariciones simultáneas para la carga de trabajo en cada partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser WORKLOAD y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037).

CONCURRENTWORKLOADACTIVITIES > *valor-entero*
Esta condición define una vinculación superior sobre el número de actividades de coordinador simultáneas y actividades anidadas para la carga de trabajo en cada partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser WORKLOAD y el ámbito de implantación para esta condición debe ser WORKLOAD OCCURRENCE (SQLSTATE 5U037).

Cada actividad anidada debe satisfacer las condiciones siguientes:

- Debe ser una actividad de coordinador reconocida. No se contará ninguna actividad de coordinador anidada que no caiga dentro de los tipos de actividades reconocidos. De modo análogo, no se contará ninguna actividad de subagente anidado, como por ejemplo las peticiones de nodo remoto.
- Debe invocarse directamente desde la lógica de usuario, como por ejemplo un procedimiento escrito por el usuario que emita sentencias de SQL.

En consecuencia, las actividades de coordinador anidado que se iniciaron automáticamente bajo la invocación de un programa de utilidad de DB2 o rutinas de los esquemas SYSIBM, SYSFUN o SYSPROC no se cuentan respecto a la vinculación superior especificada por medio de este umbral.

Este umbral no cuenta las actividades SQL internas, como por ejemplo las iniciadas estableciendo una restricción o renovando una tabla de consulta materializada, ya que son iniciadas por medio del gestor de base de datos y no invocadas directamente mediante la lógica de usuario.

CONCURRENTDBCOORDACTIVITIES > *valor-entero*
Esta condición define una vinculación superior sobre el número de actividades de coordinador de base de datos reconocidas que pueden ejecutarse simultáneamente en todas las particiones de base de datos del dominio especificado. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que se impedirá la ejecución de las actividades de cualquier coordinador de base de datos nueva. Proseguirán todas las actividades de coordinador de base de datos que estén en cola o en ejecución en la actualidad. El dominio de definición para esta condición debe ser DATABASE, acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea utilizando una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET),SERVICE SUPERCLASS o SERVICE SUBCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037).

Importante: Antes de utilizar los umbrales CONCURRENTDBCOORDACTIVITIES, familiarícese con los efectos que pueden tener sobre el sistema de base de datos. Para obtener más información, consulte el tema "Umbral CONCURRENTDBCOORDACTIVITIES".

AND QUEUEDACTIVITIES > *valor-entero* **o AND QUEUEDACTIVITIES UNBOUNDED**
Especifica un tamaño de cola para el momento en que se supere el número máximo de actividades de coordinador de base de datos. Este valor puede ser cualquier entero positivo, incluyendo el cero (SQLSTATE 42820). Un valor de cero significa que no se pondrá en cola ninguna actividad de coordinador de base de datos. Especificar UNBOUNDED pondrá en cola todas las actividades de coordinador de base de datos que superen el número máximo especificado de actividades de coordinador de base de datos y las *acciones-umbral-superado* nunca se ejecutarán. El valor por omisión es cero.

ESTIMATEDSQLCOST > *valor-bigint*
Esta condición define una vinculación superior para el coste asignado de optimizador (en activaciones de temporizador) de una actividad. Este valor puede ser cualquier entero grande positivo diferente a cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser DATABASE,

acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea utilizando una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET), SERVICE SUPERCLASS o SERVICE SUBCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037). Esta condición se implanta en la partición de base de datos del coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador del lenguaje de manipulación (DML) de datos de tipo
- Las actividades de DML anidado que se invocan desde la lógica del usuario. En consecuencia, esta condición no hace el seguimiento de las actividades de DML (a menos que su coste se incluya en la estimación del padre, en cuyo caso se hace un seguimiento indirecto de las mismas) que puede iniciar el gestor de base de datos (como por ejemplo, programas de utilidad, procedimientos o SQL interno).

SQLROWSRETURNED > *valor-entero*
Esta condición define una vinculación superior para el número de filas devueltas a una aplicación cliente desde el servidor de la aplicación. Este valor puede ser cualquier entero diferente a cero (SQLSTATE 42820). El dominio de definición para esta condición debe ser DATABASE, acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea utilizando una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET), SERVICE SUPERCLASS o SERVICE SUBCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037). Esta condición se implanta en la partición de base de datos del coordinador. Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML
- Actividades de DML anidado que derivan de la lógica del usuario. Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

Los conjuntos de resultados devueltos desde un procedimiento se tratan como actividades individuales por separado. Las filas que devuelve el propio procedimiento no se agregan.

ACTIVITYTOTALTIME > *valor-entero* **(DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)**
Esta condición define una vinculación superior para la cantidad de tiempo que el gestor de base de datos permitirá que se ejecute una actividad, incluyendo el tiempo que la actividad estuvo en cola. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820). Utilice una palabra clave de duración válida para especificar una unidad de tiempo apropiada para *valor-entero*. El dominio de definición para esta condición debe ser DATABASE, acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea utilizando una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET),SERVICE SUPERCLASS o SERVICE SUBCLASS y el ámbito de implantación debe ser DATABASE (SQLSTATE 5U037). Esta condición se implanta lógicamente en la partición de base de datos del coordinador.

Puesto que la granularidad mínima para este umbral es de cinco minutos, los valores especificados se redondean al múltiplo de cinco minutos diferente a cero que más se aproxime a cero. En los casos en los que el redondeo genere un desbordamiento o un subdesbordamiento, se seleccionará el valor más próximo del rango de entero.

El valor máximo de este umbral es de 2.147.483.400 segundos. Cualquier valor especificado que tenga un número de segundos equivalente superior a 2.147.483.400 segundos se establecerá en este número de segundos.

Es posible que los cambios en la variable de registro DB2CHECKCLIENTINTERVAL que ocasionan comprobaciones menos frecuentes afecten a la granularidad de este umbral.

SQLTEMPSPACE > *valor-entero* **(K | M | G)**
Esta condición define una vinculación superior para el tamaño de espacio de tablas temporal del sistema en cualquier partición de base de datos. Este valor puede ser cualquier entero positivo diferente a cero (SQLSTATE 42820).

Si se especifica *valor-entero K* (ya sea en mayúsculas o en minúsculas), el tamaño máximo es *valor-entero* multiplicado por 1024. Si se especifica *valor-entero M*, el tamaño máximo es *valor entero* multiplicado por 1 048 576. Si se especifica *valor-entero G*, el tamaño máximo es *valor entero* multiplicado por 1 073 741 824.

El dominio de definición para esta condición debe ser DATABASE, acción de trabajo (un umbral para un dominio de definición de acción de trabajo se crea utilizando una sentencia CREATE WORK ACTION SET o ALTER WORK ACTION SET),SERVICE SUPERCLASS o SERVICE SUBCLASS y el ámbito de implantación debe ser DATABASE PARTITION (SQLSTATE 5U037). Esta condición hace el seguimiento de las siguientes actividades:

- Actividades de coordinador de tipo DML y de trabajo de subagente correspondiente (ejecución de subsección)
- Actividades de DML anidado que derivan de la lógica del usuario y su trabajo de subagente correspondiente (ejecución de subsección). Las actividades iniciadas por medio del gestor de base de datos mediante un programa de utilidad, procedimiento o SQL interno no resultan afectadas por esta condición.

acciones-superaron-umbral
Especifica la acción que va a adoptarse cuando se supera una condición. Cada vez que se supera una condición, se registra un suceso en el supervisor de sucesos de violaciones, si hay alguno activo.

STOP EXECUTION
La ejecución de la actividad se detiene y se devuelve un error (SQLSTATE 5U026).

CONTINUE
La ejecución de la actividad no se detiene.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades que ha superado este umbral se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. El valor por omisión es COLLECT ACTIVITY DATA NONE. Si se especifica COLLECT ACTIVITY DATA, el valor por omisión es WITHOUT DETAILS.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que supera el umbral.

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todas las particiones si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL DATABASE PARTITIONS no tiene efecto y la información sobre actividades siempre se recopila únicamente en la partición coordinadora. Para umbrales de predicción y de reacción, los detalles o valores de actividades únicamente se recopilarán en la partición coordinadora.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades que ha superado el umbral se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. La información del entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

Notas

- ***Acción de umbral superado de CONTINUE y datos de supervisor de sucesos:*** Los datos de supervisor de sucesos sólo se recopilan una vez por partición cuando se ha superado una condición de umbral. Si la acción de umbral superado es CONTINUE, la actividad sigue ejecutándose y no se recopilan más datos de supervisor de sucesos para este umbral en la partición afectada. Por ejemplo, tome en consideración un umbral de tiempo de 10 minutos con una acción de CONTINUE. Una vez que una actividad haya superado la vinculación superior de 10 minutos, se recopilarán datos de supervisor de sucesos para el umbral en la partición afectada.
- ***Acción de umbral de CONTINUE para un umbral de puesta en cola:*** si se especifica una acción de umbral de CONTINUE para un umbral de puesta en cola, el tamaño de la cola será ilimitado, independientemente de si se ha especificado un valor máximo para el tamaño de la cola.
- ***Inmovilización de una clase de servicio:*** La condición de umbral de TOTALSCPARTITIONCONNECTIONS puede utilizarse para simular clases de servicio de inmovilizadas que normalmente no pueden inmovilizarse (por ejemplo, la clase de usuario por omisión o la clase de sistema por omisión). Esto resulta útil, ya que los umbrales no se aplican a los usuarios con autorización DBADM que se ejecuten en SYSDEFAULTADMWORKLOAD, mientras que una clase de servicio inmovilizado no está disponible para todo el mundo. En consecuencia, las clases de servicio por omisión no pueden inmovilizarse directamente sino a través de un umbral que permita a los usuarios con autorización DBADM unirlas cuando se conecten a la base de datos utilizando SYSDEFAULTADMWORKLOAD.

Ejemplos

Ejemplo 1: Cree un umbral que implante una utilización de espacio de tablas temporal máximo de 50M (por partición de la base de datos) en cualquier actividad de la base de datos. Ha de detenerse cualquier actividad que viole este umbral.

```
CREATE THRESHOLD DBMAX50MEGTEMPSPACE
  FOR DATABASE ACTIVITIES
  ENFORCEMENT DATABASE PARTITION
  WHEN SQLTEMPSPACE > 50 M
  STOP EXECUTION
```

Ejemplo 2: Cree un segundo umbral para limitar el tiempo de ejecución por omisión de cualquier actividad de la base de datos a un máximo de 1 hora. Ha de detenerse cualquier actividad que viole este umbral.

```
CREATE THRESHOLD DBMAX1HOURRUNTIME
  FOR DATABASE ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ACTIVITYTOTALTIME > 1 HOUR
  STOP EXECUTION
```

Ejemplo 3: Suponga que se ha creado una superclase de servicio denominada BIGQUERIES para dar acomodo a las consultas utilizando más espacio temporal que el medio y en ejecución por más de 1 hora. Los umbrales definidos en esta clase de servicio anularán temporalmente los valores establecidos por encima del nivel de la base de datos. Tenga en cuenta el modo en que se permite que siga la ejecución de las actividades que violen los umbrales de esta superclase, pero se recopila información detallada para un posterior análisis.

```
CREATE THRESHOLD BIGQUERIESMAX500MEGTEMPSPACE
  FOR SERVICE CLASS BIGQUERIES  ACTIVITIES
  ENFORCEMENT DATABASE PARTITION
  WHEN SQLTEMPSPACE  > 500 M
  COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
  CONTINUE

CREATE THRESHOLD BIGQUERIESLONGRUNNINGTIME
  FOR SERVICE CLASS BIGQUERIES  ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ACTIVITYTOTALTIME   > 10 HOURS
  COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
  CONTINUE
```

Ejemplo 4: Suponiendo que exista una carga de trabajo llamada PAYROLL, cree un umbral que implante el número máximo de actividades en la carga de trabajo en un número igual o inferior a 10.

```
CREATE THRESHOLD MAXACTIVITIESINPAYROLL
  FOR WORKLOAD PAYROLL ACTIVITIES
  ENFORCEMENT WORKLOAD OCCURRENCE
  WHEN CONCURRENTWORKLOADACTIVITIES > 10
  STOP EXECUTION
```

Ejemplo 5: Cree un umbral que implante un número máximo de 2 actividades simultáneas en la clase de servicio BIGQUERIES.

```
  CREATE THRESHOLD
MAXBIGQUERIESCONCURRENCY
    FOR SERVICE CLASS BIGQUERIES  ACTIVITIES
    ENFORCEMENT DATABASE
    WHEN CONCURRENTDBCOORDACTIVITIES > 2
    STOP EXECUTION
```

CREATE TRANSFORM

La sentencia CREATE TRANSFORM define funciones de transformación, identificados por un nombre de grupo, que se utilizan para intercambiar valores de tipo estructurado con programas en lenguaje del sistema principal y con funciones externas.

Invocación

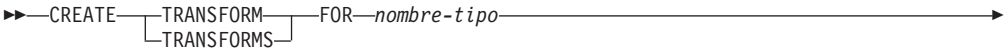
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

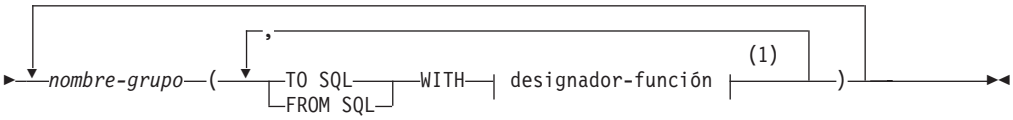
Autorización

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

- Propietario del tipo identificado por *nombre-tipo* y privilegio EXECUTE en cada función especificada
- Autorización SYSADM o DBADM

Sintaxis





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
Da nombre al grupo de transformación. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-grupo* no debe identificar un grupo de transformaciones que ya exista en el catálogo para el *nombre-tipo* especificado (SQLSTATE 42739). El *nombre-grupo* no debe empezar por los caracteres 'SYS' (SQLSTATE 42939). Como máximo, se puede especificar una de las clases siguientes de funciones FROM SQL y TO SQL para un grupo dado cualquiera (SQLSTATE 42628).

TO SQL
Define la función específica que se utiliza para transformar un valor al formato del tipo estructurado SQL definido por el usuario. La función debe tener todos sus parámetros como tipos de datos incorporados y el tipo devuelto es *nombre-tipo*.

FROM SQL
Define la función específica que se utiliza para transformar un valor a un tipo de datos incorporado representativo del tipo estructurado SQL definido por el usuario. La función debe tener un parámetro del tipo de datos *nombre-tipo* y devolver un tipo de datos incorporado (o conjunto de tipos de datos incorporados).

WITH *designador-función*
Identifica la función de transformación de forma exclusiva.

Si se especifica FROM SQL, *designador-función* debe identificar una función que satisface los requisitos siguientes:

- Existe un parámetro del tipo *nombre-tipo*.
- El tipo de retorno es un tipo incorporado o una fila donde todas sus columnas tienen tipos incorporados.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación FROM SQL que tiene LANGUAGE SQL.

Si se especifica TO SQL, *designador-función* debe identificar una función que satisface los requisitos siguientes:

- Todos los parámetros tienen tipos incorporados.
- El tipo de retorno es *nombre-tipo*.
- La signatura especifica LANGUAGE o la utilización de otra función de transformación TO SQL que tiene LANGUAGE SQL.

Si *designador-función* identifica una función que no cumple estos requisitos (de acuerdo con su utilización como función de transformación FROM SQL o TO SQL), se genera un error (SQLSTATE 428DC).

No se pueden especificar métodos (incluso si se especifica con FUNCTION ACCESS) como transformaciones mediante *designador-función*. En lugar de ello, sólo las funciones definidas por la sentencia CREATE FUNCTION pueden actuar como transformaciones (SQLSTATE 42704 o 42883).

Para obtener más información, consulte el apartado “Designadores de función, método y procedimiento” en la página 18“Designadores de función, método y procedimiento” en la página 18.

Normas

- El tipo o los tipos incorporados que se devuelven de la función FROM SQL deben corresponder directamente al tipo o tipos incorporados que son parámetros de la función TO SQL. Esto es una consecuencia lógica de la relación

inversa que existe entre estas dos funciones. 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 la sentencia SET CURRENT DEFAULT TRANSFORM GROUP para el SQL dinámico), las funciones de transformación del grupo de transformación 'DB2_PROGRAM' se utilizan (si se han definido) cuando el programa de aplicación está recuperando o enviando variables del lenguaje principal que están basadas en el tipo estructurado definido por el usuario identificado por *nombre-tipo*. Al recuperar un valor del tipo de datos *nombre-tipo*, se invoca la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado devuelto por la función de transformación. De forma similar, al enviar una variable de lenguaje principal que se asignará a un valor del tipo de datos *nombre-tipo*, se invocará la transformación TO SQL para transformar el valor del tipo de datos incorporado en el valor de tipo de datos estructurado. Si no se especifica un grupo de transformación o un grupo DB2_PROGRAM no está definido para el tipo estructurado indicado), se emite un error (SQLSTATE 42741).
- La representación del tipo de datos incorporado para una variable del lenguaje principal de tipo estructurado debe asignarse:
  - desde el resultado de la función de transformación FROM SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando normas de asignación para la recuperación) y
  - al parámetro de la función de transformación TO SQL para el tipo estructurado tal como está definida por la opción especificada TRANSFORM GROUP del mandato de precompilación (utilizando normas de asignación de memoria).

  Si la asignación de una variable de lenguaje principal no es compatible con el tipo necesario para la función de transformación aplicable, se genera un error (para el enlace de entrada: SQLSTATE 42821; para el enlace de salida: SQLSTATE 42806). Para obtener información acerca de los errores que resultan de las asignaciones de series de caracteres, consulte “Asignaciones de series de caracteres”.
- Las funciones de transformación identificadas en el grupo de transformación por omisión denominado 'DB2_FUNCTION' se utilizan siempre que una función definida por el usuario no escrita en SQL se invoca utilizando el tipo de datos *nombre-tipo* como parámetro o tipo de retorno. Esto se aplica cuando la función no especifica la cláusula TRANSFORM GROUP. Cuando se invoca la función con un argumento del tipo de datos *nombre-tipo*, se ejecuta la transformación FROM SQL para transformar el tipo estructurado en el tipo de datos incorporado devuelto por la función de transformación. De forma similar, cuando el tipo de datos de retorno de la función es un tipo de datos *nombre-tipo*, se invoca la transformación TO SQL para transformar el valor de tipo de datos incorporado devuelto del programa de función externa en el valor de tipo estructurado.
- Si un tipo estructurado contiene un atributo que también es un tipo estructurado, las funciones de transformación asociados deben expandir (o ensamblar) de forma recursiva todos los tipos estructurados anidados. Esto significa que los resultados o los parámetros de las funciones de transformación sólo constan del conjunto de tipos incorporados que representan todos los

atributos base del tipo estructurado del sujeto (incluidos todos sus tipos estructurados anidados). No existe ninguna "aplicación en cascada" de las funciones de transformación para manejar los tipos estructurados anidados.

- Las funciones identificadas en esta sentencia se resuelven de acuerdo con las normas descritas anteriormente durante la ejecución de esta sentencia. Cuando estas funciones se utilizan (implícitamente) en sentencias de SQL subsiguientes, no son objeto de otro proceso de resolución. Las funciones de transformación definidos en esta sentencia se registran exactamente como se han resuelto en esta sentencia.
- Cuando se crean o 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 de estructura definida por el usuario polígono con funciones de transformación personalizadas para C y Java, respectivamente.

```
CREATE TRANSFORM FOR POLYGON
  mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
             TO SQL WITH FUNCTION myxform_structsql)
  myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
             TO SQL WITH FUNCTION myxform_javasql)
```

CREATE TRIGGER

La sentencia CREATE TRIGGER define un activador en una base de datos.

Invocación

Esta sentencia 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

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

- Privilegio ALTER para la tabla en la que se define el activador BEFORE o AFTER
- Privilegio CONTROL para la vista en la que se define el activador INSTEAD OF
- Propietario de la vista en la que se define el activador INSTEAD OF
- Privilegio ALTERIN para el esquema de la tabla o de la vista en la que se define el activador
- Autorización SYSADM o DBADM

y uno de estos:

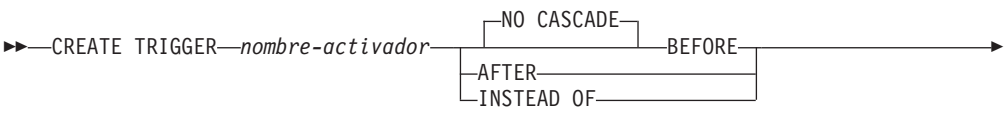
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema implícito o explícito del activador no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema del activador hace referencia a un esquema existente

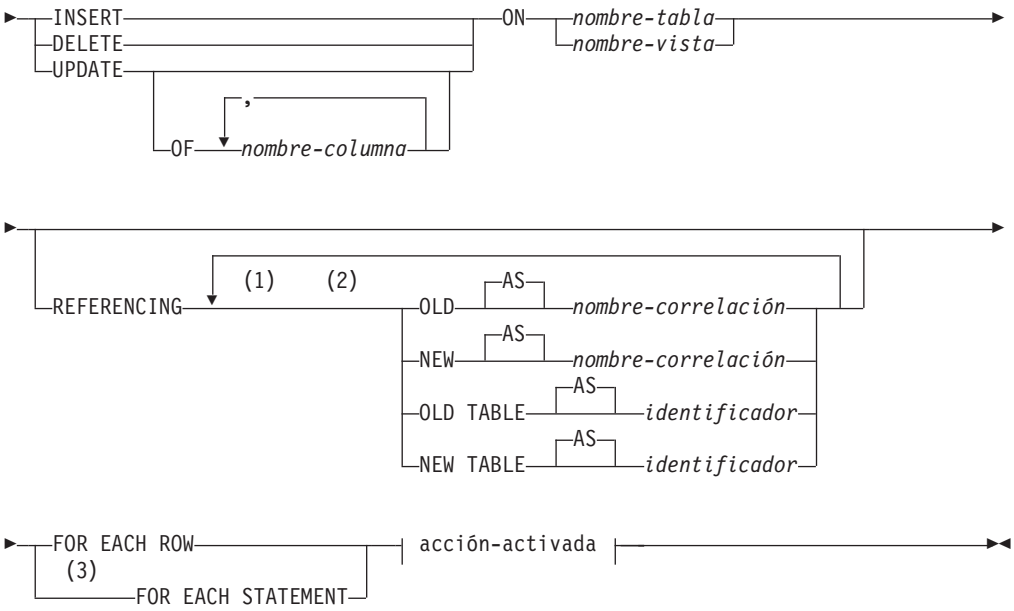
Si el ID de autorización de la sentencia no tiene autoridad SYSADM o DBADM, los privilegios (excluidos los privilegios de grupo) del ID de autorización de la sentencia deben incluir todos los siguientes, siempre que exista el activador:

- 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 que se han especificado.

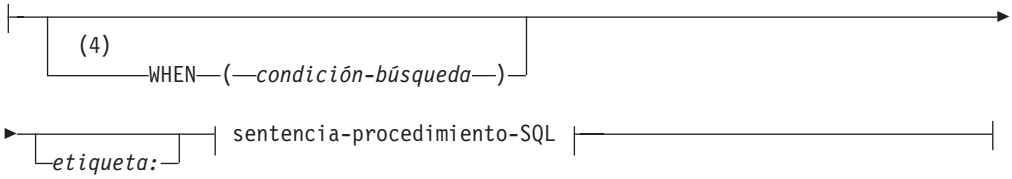
Si el definidor de un activador sólo puede crear el activador porque el definidor tiene la autorización SYSADM, se le otorgará la autorización DBADM explícita con el propósito de crear el activador.

Sintaxis

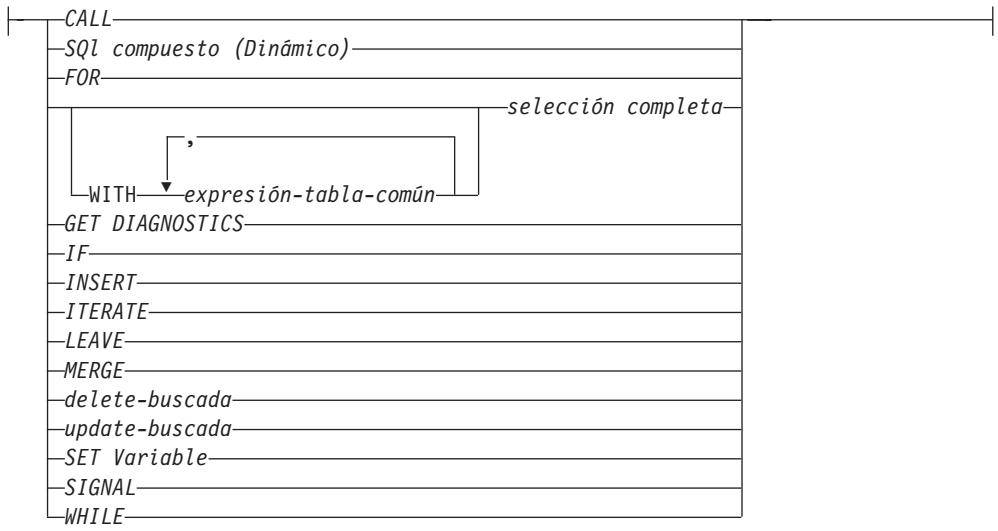




acción-activada:



sentencia-procedimiento-SQL:



Notas:

1 OLD y NEW sólo se pueden especificar una vez cada uno.

2 OLD TABLE y NEW TABLE sólo se pueden especificar una vez cada uno, y sólo para activadores AFTER o INSTEAD OF.

3 FOR EACH STATEMENT no puede especificarse para activadores BEFORE o para activadores INSTEAD OF.

4 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 se trata de un activador BEFORE, el *nombre-columna* especificado no puede ser una columna generada distinta de la columna de identidad (SQLSTATE 42989). Ningún *nombre-columna* puede aparecer más de una vez en la lista *nombre-columna* (SQLSTATE 42711). El activador sólo se activará mediante la actualización de una columna identificada en la lista*nombre-columna*. Esta cláusula no puede especificarse para un activador INSTEAD OF (SQLSTATE 42613).

ON

nombre-tabla
Designa la tabla sujeto de la definición del activador BEFORE o del activador AFTER. El nombre debe especificar una tabla base o un alias que se resuelva dando como resultado una tabla base (SQLSTATE 42704 ó 42809). El nombre no debe especificar una tabla de 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 sin columnas de tipo XML (SQLSTATE 42704 o 42809). El nombre no debe especificar una vista de catálogo (SQLSTATE 42832). El nombre no debe especificar una vista que se haya definido utilizando WITH CHECK OPTION (una vista simétrica) o una vista en la que se haya definido una vista simétrica, directa o indirectamente (SQLSTATE 428FQ).

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.

Un nombre de correlación NEW AS no puede ser de tipo XML.

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:

- Los nombres de la correlación OLD y NEW ni los nombres de OLD TABLE y NEW TABLE no pueden ser idénticos (SQLSTATE 42712).
- Para un activador solamente se puede especificar un *nombre-correlación* OLD y uno NEW (SQLSTATE 42613).
- Sólo se puede especificar un *identificador* OLD TABLE y otro NEW TABLE para un activador (SQLSTATE 42613).

- El *nombre-correlación* OLD y el *identificador* OLD TABLE sólo se pueden utilizar si el suceso del activador es una operación DELETE o una operación UPDATE (SQLSTATE 42898). Si la operación es DELETE, el *nombre-correlación* OLD captura el valor de la fila suprimida. Si la operación es UPDATE, captura el valor de la fila antes de la operación UPDATE. Lo mismo se aplica al *identificador* OLD TABLE y al conjunto de filas afectadas.
- El *nombre-correlación* NEW y el *identificador* NEW TABLE sólo se pueden utilizar si el suceso del activador es una operación INSERT o una operación UPDATE (SQLSTATE 42898). En ambas operaciones, el valor de NEW captura el nuevo estado de la fila como lo proporciona la operación original y modificado por cualquier activador BEFORE que se haya ejecutado hasta ese momento. Lo mismo se aplica al *identificador* NEW TABLE y al conjunto de filas afectadas.
- Los identificadores OLD TABLE o NEW TABLE no se pueden definir en un activador BEFORE (SQLSTATE 42898).
- Una variable de transición NEW no se puede definir en un activador AFTER (SQLSTATE 42987).
- Los nombres de correlación OLD o NEW no se pueden definir en un activador FOR EACH STATEMENT (SQLSTATE 42899).
- Las tablas de transición no pueden modificarse (SQLSTATE 42807).
- El total de las referencias a las columnas de la tabla de transición y a las variables de transición de la acción activada no puede sobrepasar el límite del número de columnas de una tabla o la suma de sus longitudes no puede exceder la longitud máxima de una fila de una tabla (SQLSTATE 54040).
- El ámbito de cada *nombre-correlación* y de cada *identificador* es la totalidad de la definición del activador.

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 realiza siempre.

La cláusula WHEN no puede especificarse para los activadores INSTEAD OF (SQLSTATE 42613).

Una referencia a una variable de transición con un tipo de datos XML sólo se puede utilizar en un predicado VALIDATED.

etiqueta:
Especifica la etiqueta de una sentencia de procedimiento de SQL. La etiqueta debe ser exclusiva dentro de una lista de sentencias de procedimiento de SQL, incluidas las sentencias compuestas anidadas dentro de la lista. Tenga en cuenta que las sentencias compuestas que no están anidadas pueden utilizar la misma etiqueta. Varias sentencias de control de SQL admiten la especificación de una lista de sentencias de procedimiento de SQL.

Sólo la sentencia FOR, la sentencia WHILE y la sentencia 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 acción activada de un activador BEFORE en una columna de tipo XML puede invocar la función XMLVALIDATE a través de una sentencia SET, puede dejar los valores de tipo XML sin cambiar o asignarlos a NUL mediante una sentencia SET.

La *sentencia-procedimiento-SQL* no debe contener una sentencia no soportada (SQLSTATE 42987).

La *sentencia-procedimiento-SQL* no puede hacer referencia a una variable de transición no definida (SQLSTATE 42703), a un objeto federado (SQLSTATE 42997) o a una tabla temporal declarada (SQLSTATE 42995).

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 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).
- 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 sentencias de SQL activadas se devuelven mediante SQLSTATE 09000 a menos que el error se considere grave. Si el error es grave, se devuelve el SQLSTATE de error grave. El campo SQLERRMC de la SQLCA de un error no grave incluirá el nombre de activador, el SQLCODE, el SQLSTATE y tantos símbolos de la anomalía como quepan.

  La *sentencia-procedimiento-SQL* puede incluir una sentencia SIGNAL SQLSTATE o una función RAISE_ERROR. En ambos casos, la SQLSTATE devuelta es la especificada en la sentencia SIGNAL SQLSTATE o la condición RAISE_ERROR.
- Crear un activador con un nombre de esquema que todavía no existe dará como resultado la creación implícita del esquema siempre y cuando el ID de autorización de la sentencia tenga la autoridad IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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.
- ***Columna de indicación de fecha y hora de cambio de fila***: Después de la ejecución de todos los activadores BEFORE, el gestor de bases de datos genera un valor para una columna de indicación de fecha y hora de cambio de fila. No se puede hacer referencia a una columna de indicación de fecha y hora de cambio de fila en el cuerpo de un activador BEFORE (SQLSTATE 42989).
- ***Columna generada por expresión***: Después de la ejecución de todos los activadores BEFORE, el gestor de bases de datos genera un valor para una columna generada por expresión. No se puede hacer referencia a una columna generada por expresión en el cuerpo de un activador BEFORE (SQLSTATE 42989).

- *Columna DB2SECURITYLABEL*: Se puede hacer referencia a una columna DB2SECURITYLABEL en el cuerpo de activador de BEFORE TRIGGER, pero no se puede cambiar en el cuerpo de un activador BEFORE (SQLSTATE 42989).
- *Vistas de sólo lectura*: añadir el activador INSTEAD OF para una vista influye en las características de sólo lectura de la vista. Si una vista de sólo lectura tiene una relación de dependencia con un activador INSTEAD OF, el tipo de operación que se define para el activador INSTEAD OF define si se puede suprimir, insertar o actualizar la vista.
- *Valores de las variables de transición y activadores INSTEAD OF*: los valores iniciales de las nuevas variables de transición o de las nuevas columnas de la tabla de transición que se pueden ver en un activador INSTEAD OF INSERT se establecen como se indica a continuación:
  - Si se especifica explícitamente un valor para una columna en la operación de inserción, la variable de transición nueva correspondiente es dicho valor especificado explícitamente.
  - Si no se especifica explícitamente un valor para una columna en la operación de inserción o se especifica la cláusula DEFAULTS, la variable de transición nueva correspondiente es:
    - El valor por omisión de la columna de tabla subyacente si la columna de vista se puede actualizar (sin el activador INSTEAD OF)
    - De lo contrario, será el valor nulo

  Los valores iniciales de las nuevas variables de transición que se pueden ver en un activador INSTEAD OF UPDATE se establecen como se indica a continuación:
  - Si se especifica un valor explícitamente para una columna de la operación de actualización, la nueva variable de transición correspondiente es dicho valor especificado explícitamente.
  - Si se especifica explícitamente la cláusula DEFAULT para una columna de la operación de actualización, la nueva variable de transición correspondiente es:
    - El valor por omisión de la columna de tabla subyacente si la columna de vista se puede actualizar (sin el activador INSTEAD OF)
    - De lo contrario, será el valor nulo
  - De lo contrario, la nueva variable de transición correspondiente es el valor existente de la columna en la fila.
- *Activadores y tablas con tipo*: Un activador BEFORE o AFTER puede asociarse a una tabla con tipo en cualquier nivel de una jerarquía de tablas. Si una sentencia de SQL activa varios activadores, 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.
- ***Compatibilidades***
  – Para mantener la compatibilidad con las versiones anteriores de DB2:
    - OLD_TABLE se puede especificar en lugar de OLD TABLE, y NEW_TABLE se puede especificar en lugar de NEW TABLE
    - Se puede especificar MODE DB2SQL a continuación de FOR EACH ROW o FOR EACH STATEMENT

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 lleve a cabo la siguiente acción de comprobación y de acción (si es necesaria):

- Si la cantidad disponible es inferior al 10% de la cantidad máxima de existencias, se emitirá una petición de entrega solicitando el número de artículos de la pieza en cuestión sea la cantidad máxima en existencias menos la cantidad disponible.

El activador interactuará con la tabla PARTS con estas columnas: PARTNO, DESCRIPTION, ON_HAND, MAX_STOCKED y PRICE.

ISSUE_SHIP_REQUEST es una función definida por el usuario que envía un formulario de pedido de piezas adicionales a la empresa adecuada.

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N
  FOR EACH ROW
  WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
  BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
  END
```

Ejemplo 3: Repita el supuesto del Ejemplo 2, pero utilice una sentencia completa en lugar de una sentencia VALUES para invocar la función definida por el usuario. Este ejemplo también muestra cómo definir el activador como un activador de sentencia en lugar de un activador de fila. Para cada fila de la tabla de transición que se evalúa como verdadera para la cláusula WHERE, se emite una petición de entrega para la pieza.

```
CREATE TRIGGER REORDER
   AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
   REFERENCING NEW TABLE AS NTABLE
   FOR EACH STATEMENT
```

```
    BEGIN ATOMIC
       SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
          FROM NTABLE
       WHERE (ON_HAND < 0.10 * MAX_STOCKED);
  END
```

Ejemplo 4: cree un activador que provoque un error cuando se produzca una actualización que daría como resultado un aumento de salario mayor que el diez por ciento del salario actual.

```
CREATE TRIGGER RAISE_LIMIT
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING NEW AS N OLD AS O
  FOR EACH ROW
  WHEN (N.SALARY > 1.1 * O.SALARY)
         SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT='Incremento de salario>10%'
```

Ejemplo 5: suponga una aplicación que registre y haga seguimientos de los cambios en los precios de las existencias. Esta base de datos contiene dos tablas CURRENTQUOTE y QUOTEHISTORY.

```
Tablas: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
        QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)
```

Cuando la columna QUOTE de CURRENTQUOTE se actualiza, la nueva oferta (quote en inglés) debe copiarse con una indicación de fecha y hora, en la tabla QUOTEHISTORY. Asimismo, la columna STATUS de CURRENTQUOTE debería actualizarse para reflejar si las existencias:

1. son un valor en alza;
2. están en valor máximo del año;
3. son un valor a la baja;
4. están en el valor mínimo del año;
5. son un valor estable.

Las sentencias CREATE TRIGGER que realizan este cometido son las siguientes:

- Definición del activador para determinar el estado:

```
  CREATE TRIGGER STOCK_STATUS
    NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
    REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
    FOR EACH ROW
    BEGIN ATOMIC
       SET NEWQUOTE.STATUS =
         CASE
            WHEN NEWQUOTE.QUOTE >
                  (SELECT MAX(QUOTE) FROM QUOTEHISTORY
                  WHERE SYMBOL = NEWQUOTE.SYMBOL
                  AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
               THEN 'Alto'
            WHEN NEWQUOTE.QUOTE <
(SELECT MIN(QUOTE) FROM QUOTEHISTORY
                  WHERE SYMBOL = NEWQUOTE.SYMBOL
                  AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
               THEN 'Bajo'
            WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
               THEN 'En alza'
            WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
               THEN 'A la baja'
            WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
               THEN 'Estable'
         END;
    END
```

- Definición del activador para registrar un cambio en la tabla QUOTEHISTORY:

```
CREATE TRIGGER RECORD_HISTORY
  AFTER UPDATE OF QUOTE ON CURRENTQUOTE
  REFERENCING NEW AS NEWQUOTE
  FOR EACH ROW
  BEGIN ATOMIC
    INSERT INTO QUOTEHISTORY
      VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
  END
```

Ejemplo 6: cree un activador que altere temporalmente los cambios realizados en el campo de ubicación en el registro de empleado de la tabla org. Este activador puede ser útil si se procesan nuevos registros de empleados obtenidos cuando se adquirió una empresa más pequeña y la ubicación de destino asignada al empleado es 'Toronto' y la nueva ubicación de destino es 'Los Angeles'. El activador before asegurará que independientemente del valor que la aplicación asigne para este campo, el valor resultante final será 'Los Angeles'.

```
CREATE TRIGGER LOCATION_TRIGGER
  NO CASCADE
  BEFORE UPDATE ON ORG
  REFERENCING
    OLD AS PRE
    NEW AS POST
  FOR EACH ROW
  WHEN (POST.LOCATION = 'Toronto')
        SET POST.LOCATION = 'Los Angeles';
 END
```

Ejemplo 7: cree un activador BEFORE que valide automáticamente los documentos XML que contengan descripciones de productos nuevos antes de que se inserten en la tabla PRODUCT de la base de datos SAMPLE:

```
CREATE TRIGGER NEWPROD NO CASCADE BEFORE INSERT ON PRODUCT
   REFERENCING NEW AS N
   FOR EACH ROW
   BEGIN ATOMIC
      SET (N.DESCRIPTION) = XMLVALIDATE(N.DESCRIPTION
         ACCORDING TO XMLSCHEMA ID producto);
   END
```

CREATE TRUSTED CONTEXT

La sentencia CREATE TRUSTED CONTEXT define un contexto fiable en el servidor actual.

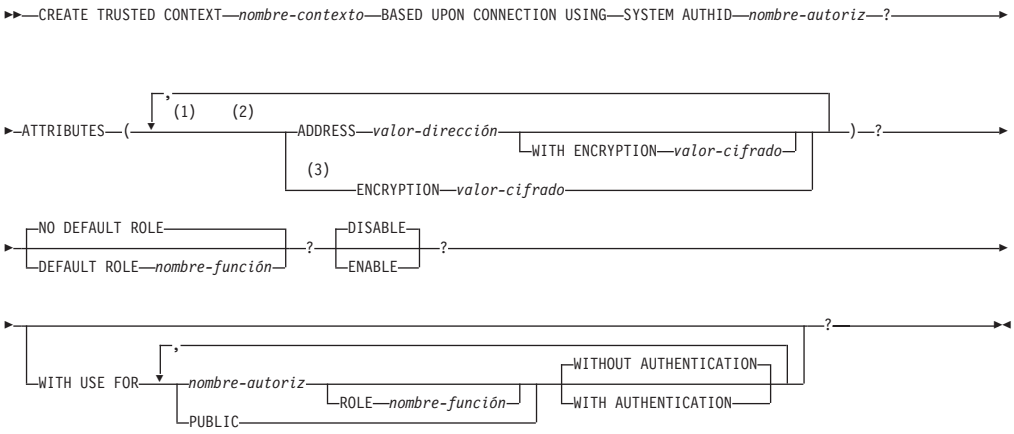
Invocación

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

Autorización

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

Sintaxis



Notas:

1 Cada una de las cláusulas ATTRIBUTES, DEFAULT ROLE, ENABLE y WITH USE puede especificarse una vez como máximo (SQLSTATE 42614).

2 Cada nombre de atributo y su valor correspondiente deben ser exclusivos (SQLSTATE 4274D).

3 ENCRYPTION no se puede especificar más de una vez (SQLSTATE 42614); sin embargo, WITH ENCRYPTION puede especificarse para cada una de las veces que se especifique ADDRESS.

Descripción

nombre-contexto
Da nombre al contexto fiable. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El nombre no debe identificar un contexto fiable que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

BASED UPON CONNECTION USING SYSTEM AUTHID *nombre-autoriz*
Especifica que el contexto es una conexión establecida por medio del ID de

autorización del sistema *nombre-autoriz*, que no debe asociarse con un contexto fiable existente (SQLSTATE 428GL). No puede ser el ID de autorización de la sentencia (SQLSTATE 42502).

ATTRIBUTES (...)
Especifica una lista de uno o más atributos de confianza de conexión sobre la que se define el contexto fiable.

ADDRESS *valor-dirección*
Especifica la dirección de comunicación real que utiliza el cliente para comunicar con el servidor de base de datos. El único protocolo soportado es TCP/IP. El atributo ADDRESS puede especificarse varias veces, pero cada par *valor-dirección* debe ser exclusivo para el conjunto de atributos (SQLSTATE 4274D).

Al establecer una conexión fiable, si se definen varios valores para el atributo ADDRESS de un contexto fiable, se toma en consideración una conexión candidata para que coincida con este atributo si la dirección que utiliza la conexión coincide con cualquiera de los valores definidos para el atributo ADDRESS del contexto fiable.

valor-dirección
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ADDRESS. El *valor-dirección* debe ser una dirección IPv4, una dirección IPv6 o un nombre de dominio seguro.

- Una dirección IPv4 no debe contener espacios iniciales y está representada como dirección decimal con puntos. Un ejemplo de una dirección IPv4 es 9.112.46.111. El valor 'localhost' o su representación equivalente '127.0.0.1' no dará como resultado una coincidencia; en su lugar debe especificarse la dirección IPv4 real del sistema principal.
- Una dirección IPv6 no debe contener espacios iniciales y está representada como dirección hexadecimal con dos puntos. Un ejemplo de una dirección IPv6 es 2001:0DB8:0000:0000:0008:0800:200C:417A. Las direcciones IPv6 correlacionadas con IPv4 (por ejemplo, ::ffff:192.0.2.128) no dará como resultado una coincidencia. De modo análogo, 'localhost' o su representación abreviada de IPv6 '::1' no dará como resultado una coincidencia.
- Un nombre de dominio se convierte en una dirección IP por medio del servidor del nombre de dominio en el que se determina una dirección IPv4 o IPv6 resultante. Un ejemplo de un nombre de dominio es corona.torolab.ibm.com. Cuando un nombre de dominio se convierte en una dirección, el resultado de esta conversión podría ser un conjunto de una o más direcciones IP. En este caso, se dice que la conexión de entrada coincide con el atributo ADDRESS de un objeto de contexto fiable si la dirección IP en la que se origina la conexión coincide con alguna de las direcciones IP para las que se convirtió el nombre de dominio. Al crear un objeto de contexto fiable, resulta ventajoso proporcionar valores de nombre de dominio para el atributo ADDRESS en vez de las direcciones IP estáticas, particularmente en entornos de Protocolo de configuración de sistema principal dinámico (Dynamic Host Configuration Protocol - DHCP). Con DHCP, un dispositivo puede disponer de una dirección de IP diferente cada vez que se conecte a la red. Por tanto, si se proporciona una dirección IP estática para el atributo ADDRESS de un objeto de contexto fiable, es posible que algún dispositivo

adquiera alguna conexión fiable de modo no intencional. Proporcionar nombres de dominio para el atributo ADDRESS de un objeto de contexto fiable evita este problema en entornos DHCP.

WITH ENCRYPTION *valor-cifrado*
Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red para este *valor-dirección* específico. Este *valor-cifrado* altera temporalmente el valor de atributo ENCRYPTION global para este *valor-dirección* específico.

valor-cifrado
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser una de las siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico
- LOW, se necesita un cifrado ligero mínimo; el tipo de autentificación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el valor de cifrado para esta dirección específica
- HIGH, debe utilizarse el cifrado de Secure Sockets Layer (SSL) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 en el caso de que la conexión de entrada vaya a coincidir con el valor de cifrado para esta dirección específica

ENCRYPTION *valor-cifrado*
Especifica el nivel mínimo de cifrado de la secuencia de datos o cifrado de red. El valor por omisión es NONE.

valor-cifrado
Especifica una constante de serie que contiene el valor que ha de asociarse al atributo fiable ENCRYPTION para este *valor-dirección* específico. El *valor-cifrado* debe ser una de las siguientes (SQLSTATE 42615):

- NONE, no se necesita ningún nivel de cifrado específico para una conexión de entrada que coincida con el atributo ENCRYPTION de este objeto de contexto fiable
- LOW, se necesita un cifrado ligero mínimo; el tipo de autentificación del gestor de base de datos debe ser DATA_ENCRYPT si una conexión de entrada va a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable
- HIGH, debe utilizarse el cifrado de Secure Sockets Layer (SSL) para la comunicación de datos entre el cliente de DB2 y el servidor de DB2 en el caso de que una conexión de entrada vaya a coincidir con el atributo ENCRYPTION de este objeto de contexto fiable.

La tabla siguiente resume el momento en que puede utilizarse un contexto fiable, en función del cifrado utilizado por la conexión existente. Si el contexto fiable no puede utilizarse para la conexión, se devuelve un aviso (SQLSTATE 01679) y el campo SQLWARN8 de SQLCA se establece en 'Y', el cual indica que la conexión es una conexión regular (no fiable).

Tabla 24. Cifrado y contextos fiables

Cifrado utilizado por la conexión existente	Valor de ENCRYPTION para el contexto fiable	¿Puede utilizarse el contexto fiable para la conexión?
No hay cifrado	'NONE'	Sí
No hay cifrado	'LOW'	No
No hay cifrado	'HIGH'	No
Cifrado bajo (DATA_ENCRYPT)	'NONE'	Sí
Cifrado bajo (DATA_ENCRYPT)	'LOW'	Sí
Cifrado bajo (DATA_ENCRYPT)	'HIGH'	No
Cifrado alto (SSL)	'NONE'	Sí
Cifrado alto (SSL)	'LOW'	Sí
Cifrado alto (SSL)	'HIGH'	Sí

NO DEFAULT ROLE o DEFAULT ROLE *nombre-función*
Especifica si se asocia o no una función por omisión con una conexión fiable basada en este contexto fiable. El valor por omisión es NO DEFAULT ROLE.

NO DEFAULT ROLE
Especifica que el contexto fiable no tiene una función por omisión.

DEFAULT ROLE *nombre-función*
Especifica que *nombre-función* es la función por omisión para el contexto fiable. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). Esta función se utiliza con el usuario de una conexión fiable, basada en este contexto fiable, cuando el usuario no tenga definida una función específica de usuario como parte de la definición del contexto fiable.

DISABLE o ENABLE
Especifica si se crea o no el contexto fiable en el estado habilitado o inhabilitado. El valor por omisión es DISABLE.

DISABLE
Especifica que se cree el contexto fiable en el estado inhabilitado. Un contexto fiable que esté inhabilitado no se tomará en consideración cuando se establezca una conexión fiable.

ENABLE
Especifica que se cree el contexto fiable en el estado habilitado.

WITH USE FOR
Especifica los usuarios que pueden utilizar una conexión fiable basada en este contexto fiable.

nombre-autoriz
Especifica que la conexión fiable puede utilizarse por medio del *nombre-autoriz* especificado. El *nombre-autoriz* no debe especificarse más de una vez con la cláusula WITH USE FOR (SQLSTATE 428GM). Tampoco debe ser el ID de autorización de la sentencia (SQLSTATE 42502). Si la definición de un contexto fiable permite el acceso tanto por parte de PUBLIC como por parte de una lista de usuarios, las especificaciones para un usuario alterarán temporalmente las especificaciones para PUBLIC. Por

ejemplo, suponga que se define un contexto fiable que permite el acceso tanto de PUBLIC WITH AUTHENTICATION como de JOE WITHOUT AUTHENTICATION. Si JOE utiliza el contexto fiable, no se necesitará la autentificación. Sin embargo, si GEORGE utiliza el contexto fiable, se necesitará la autentificación.

ROLE *nombre-función*
Especifica que *nombre-función* es la función que ha de utilizarse para el usuario cuando una conexión fiable está utilizando el contexto fiable. El *nombre-función* debe identificar una función que exista en el servidor actual (SQLSTATE 42704). La función explícitamente especificada para el usuario altera temporalmente las funciones por omisión asociadas con el contexto fiable.

PUBLIC
Especifica que cualquier usuario puede utilizar una conexión fiable basada en este contexto fiable. PUBLIC no debe especificarse más de una vez (SQLSTATE 428GM). Todos los usuarios que utilicen dicha conexión fiable hacen uso de los privilegios asociados con la función por omisión para el contexto fiable asociado. Si no se define una función por omisión para el contexto fiable, no habrá ninguna función asociada con los usuarios que usen una conexión fiable basada en este contexto fiable.

WITHOUT AUTHENTICATION o WITH AUTHENTICATION
Especifica si conmutar el usuario en una conexión fiable requiere autentificación del usuario o no. El valor por omisión es WITHOUT AUTHENTICATION.

WITHOUT AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable para este usuario no requiere autentificación.

WITH AUTHENTICATION
Especifica que conmutar el usuario actual en una conexión fiable para este usuario requiere autentificación.

Normas

- Una sentencia de SQL exclusiva del contexto fiable debe ir después de una sentencia COMMIT o ROLLBACK (SQLSTATE 5U021). Las sentencias de SQL exclusivas del contexto fiable son:
  - CREATE TRUSTED CONTEXT, ALTER TRUSTED CONTEXT o DROP (TRUSTED CONTEXT)
- Una sentencia de SQL exclusiva del contexto fiable no puede emitirse en una transacción global; por ejemplo, una transacción XA o una transacción global que se inicie como parte de una confirmación en dos fases para las transacciones federadas (SQLSTATE 51041).

Notas

- Al proporcionar una dirección IP como parte de una definición de contexto fiable, la dirección debe estar en el formato que esté vigente para la red. Por ejemplo, proporcionar una dirección en un formato IPv6 cuando la red sea IPv4 no dará como resultado una coincidencia. En un entorno mixto, resulta ventajoso especificar las representaciones IPv4 e IPv6 de la dirección, o mejor aún, especificar un nombre de dominio seguro (por ejemplo, corona.torolab.ibm.com), que oculta los detalles de formato de dirección.
- ***Especificar una función en la definición de un contexto fiable:*** La definición de un contexto fiable puede designar una función para un ID de autorización

específico y una función por omisión a utilizar para los ID de autorización para los que no se haya especificado una función específica en la definición del contexto fiable. Esta función puede utilizarse con una conexión fiable basada en el contexto fiable, pero no hará que la función esté disponible fuera de una conexión fiable basada en el contexto fiable.

- Al emitir una sentencia de SQL de lenguaje de manipulación de datos (DML) que utilice una conexión fiable, se tomarán en consideración los privilegios que mantenga una función asignada por el contexto vigentes para el ID de autorización en la definición del contexto fiable asociado, además de otros privilegios que mantenga directamente el ID de autorización de la sentencia, u otras funciones que mantenga indirectamente el ID de autorización de la sentencia.
- No se tomarán en consideración, para las sentencias de SQL de lenguaje de definición de datos (DDL), los privilegios que mantenga una función asignada por el contexto vigentes para el ID de autorización en la definición del contexto fiable asociado. Por ejemplo, para crear un objeto, el ID de autorización de la sentencia debe poder efectuar dicha acción sin incluir los privilegios que mantenga la función asignada por el contexto.
- Al instalar una aplicación nueva que autentifique en DB2 utilizando las mismas credenciales que una aplicación existente en la misma máquina y que saque partido de un contexto fiable, es posible que la aplicación nueva también saque partido del mismo objeto de contexto fiable (por ejemplo, heredando la función de contexto fiable). Es posible que no sea esta la intención del administrador de seguridad. Es posible que el administrador de seguridad desee activar el recurso de comprobación de DB2 para averiguar que aplicaciones están sacando partido de los objetos de contexto fiable.
- Sólo se permite una sentencia de SQL exclusiva del contexto fiable sin confirmar a la vez entre todas las particiones de la base de datos. Si se ejecuta una sentencia de SQL exclusiva del contexto fiable sin confirmar, las siguientes sentencias de SQL exclusivas del contexto fiable esperarán hasta que se confirme o retrotraiga la sentencia de SQL exclusiva del contexto fiable actual.
- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a dos ID de usuario diferentes. Cuando el usuario actual de la conexión se conmuta al ID de usuario JOE, no se requiere la autentificación. Sin embargo, la autentificación se requiere cuando el usuario actual de la conexión se conmuta al ID de usuario BOB. Tenga en cuenta que el contexto fiable tiene una función por omisión llamada *función-contexto*. Esto implica que los usuarios que trabajen dentro de los confines de este contexto fiable heredarán los privilegios asociados con la función *función-contexto*.

```
CREATE TRUSTED CONTEXT APPSERVER
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  DEFAULT ROLE CONTEXT_ROLE
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR JOE WITHOUT AUTHENTICATION
    BOB WITH AUTHENTICATION
```

Ejemplo 2: Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a cualquier ID de usuario sin autentificación.

```
CREATE TRUSTED CONTEXT SECUREROLE
  BASED UPON CONNECTION USING SYSTEM AUTHID PBIRD
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

Ejemplo 3: Cree una conexión fiable tal que el usuario actual de una conexión fiable basada en este contexto fiable pueda conmutarse a cualquier ID de usuario sin autentificación. La diferencia entre este contexto fiable y el contexto fiable creado en el ejemplo 2, es que este contexto fiable tiene un atributo adicional llamado ENCRYPTION. El valor del atributo ENCRYPTION para el contexto fiable SECUREROLEENCRYPT indica que el valor de cifrado que utilice una conexión debe tener como mínimo un "cifrado bajo" (consulte la Tabla 24 en la página 602) para coincidir con este atributo de contexto fiable.

```
CREATE TRUSTED CONTEXT SECUREROLEENCRYPT
  BASED UPON CONNECTION USING SYSTEM AUTHID SHARPER
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.113.204'
    ENCRYPTION 'LOW')
  WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

Ejemplo 4: Cree un contexto fiable tal que las conexiones efectuadas por el usuario WRJAIBI desde las direcciones 9.26.146.201 y 9.26.146.203 resulten fiables cuando no se utilice ningún cifrado, pero en el que una conexión efectuada por el usuario WRJAIBI desde la dirección 9.26.146.202 requiera un nivel de cifrado LOW para resultar fiable.

```
CREATE TRUSTED CONTEXT WALIDLOCSENSITIVE
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  ENABLE
  ATTRIBUTES (ADDRESS '9.26.146.201',
    ADDRESS '9.26.146.202' WITH ENCRYPTION 'LOW',
    ADDRESS '9.26.146.203'
    ENCRYPTION 'NONE')
```

CREATE TYPE (matriz)

La sentencia CREATE TYPE (matriz) define un tipo de matriz. Una matriz está basada en uno de los tipos de datos incorporados.

Invocación

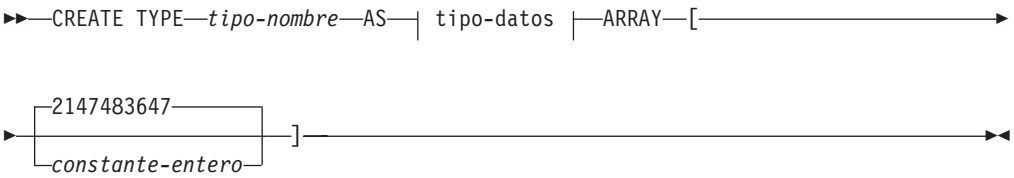
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo de matriz no hace referencia a un esquema existente
- Privilegio CREATE_IN para el esquema, si el nombre de esquema del tipo de matriz hace referencia a un esquema existente
- Autorización SYSADM o DBADM

Sintaxis



Descripción

nombre-tipo
Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe designar ningún otro tipo (incorporado o definido por el usuario) que ya exista en el servidor actual. El nombre sin calificar no debe ser el mismo que un tipo de datos incorporado o BOOLEAN, BINARY o VARBINARY (SQLSTATE 42918).

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

Si se especifica un nombre compuesto de dos partes *nombre-tipo*, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos
Especifica el tipo de datos de los elementos de matriz. Éste es uno de los tipos de datos que se describen en "CREATE TABLE", a parte de LONG VARCHAR, LONG VARGRAPHIC, REFERENCE, SYSPROC.DB2SECURITYLABEL, XML o los tipos definidos por el usuario (SQLSTATE 429C2).

ARRAY [*constante-entero***]**
Especifica que el tipo es una matriz con una cardinalidad máxima de

constante-entero. El valor debe ser un número positivo mayor que cero e inferior al mayor valor entero positivo (SQLSTATE 42820). El valor por omisión es el mayor valor entero positivo (2 147 483 647).

La cardinalidad de una matriz de un determinado sistema está limitada por la cantidad total de memoria disponible para las aplicaciones DB2. como tal, aunque puedan crearse matrices con grandes cardinalidades, es posible que no puedan utilizarse todos los elementos.

Ejemplos

Ejemplo 1: Cree un tipo de matriz denominado PHONENUMBERS con un máximo de 5 elementos que tengan el tipo de datos DECIMAL(10, 0).

```
   CREATE TYPE
PHONENUMBERS AS DECIMAL(10,0)
     ARRAY[5]
```

Ejemplo 2: Cree un tipo de matriz denominado NUMBERS con valor por omisión número de elementos en el esquema GENERIC.

```
   CREATE TYPE GENERIC.NUMBERS AS BIGINT
     ARRAY[]
```

CREATE TYPE (diferenciado)

La sentencia CREATE TYPE (diferenciado) define un tipo diferenciado. El tipo diferenciado siempre tiene su fuente en los tipos de datos incorporados. La ejecución satisfactoria de la sentencia también genera funciones para 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

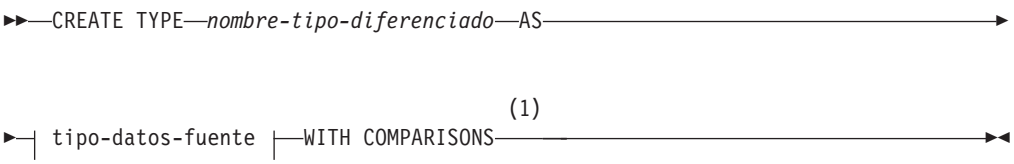
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

El ID de autorización de la sentencia debe tener como mínimo uno de los privilegios siguientes:

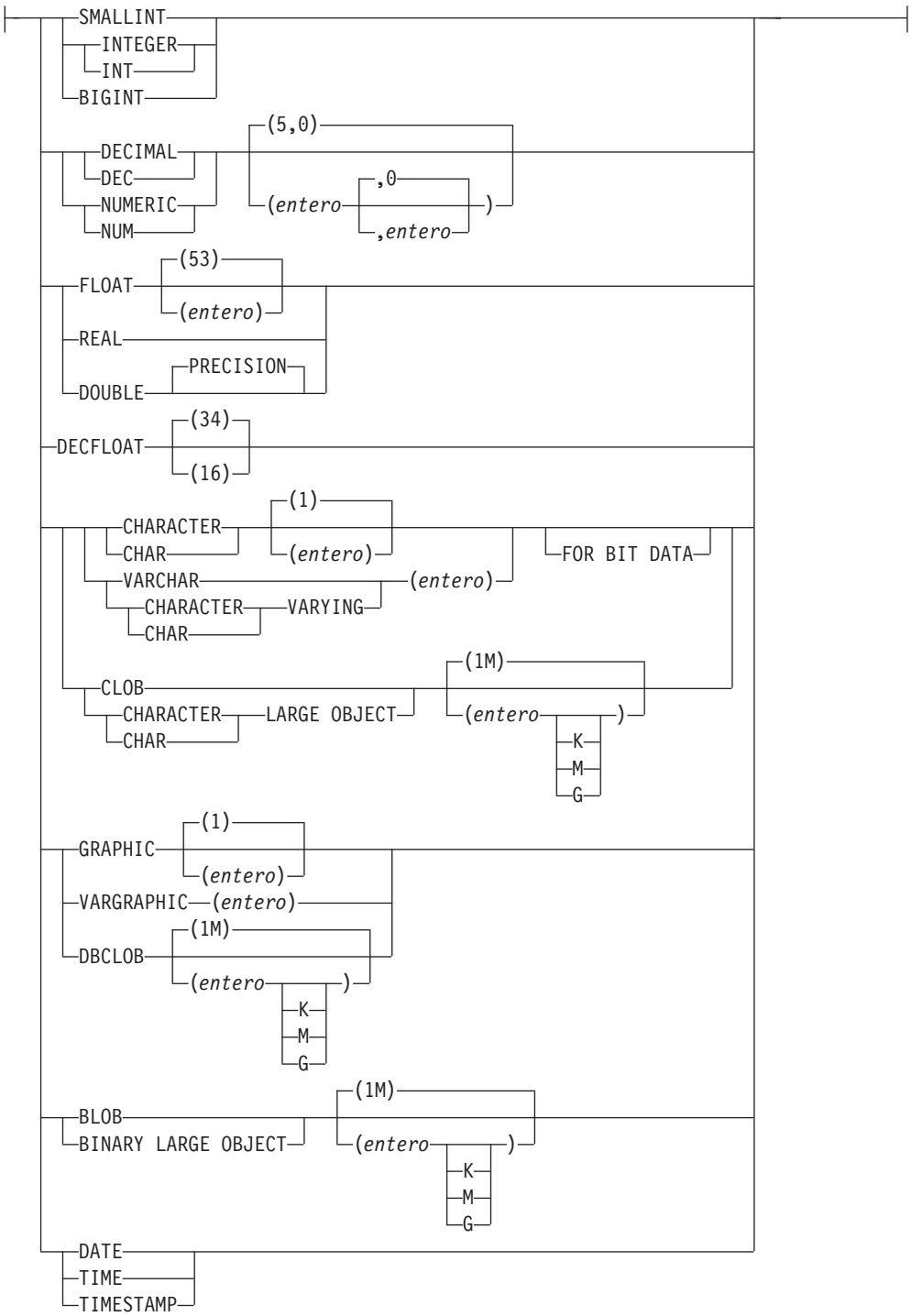
- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo diferenciado no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo diferenciado hace referencia a un esquema existente
- Autorización SYSADM o DBADM

Sintaxis



tipo-datos-fuente:

tipo-incorporado:



Notas:

1 Necesario para todos los tipos-datos-fuente excepto LOB, para el cual las comparaciones no están soportadas.

Descripción

nombre-tipo-diferenciado
Indica el nombre del tipo diferenciado. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado o

definido por el usuario) que ya exista en el servidor actual. El nombre no calificado no debe ser el mismo que el nombre de un tipo de datos incorporado ni BOOLEAN, BINARY o VARBINARY (SQLSTATE 42918). El nombre no calificado tampoco debe ser ARRAY, INTERVAL ni ROWID.

En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. La forma calificada es un *nombre-esquema* seguido de un punto y un identificador SQL.

Algunos nombres que se utilizan como palabras clave en los predicados están reservados para que los utilice el sistema y no se pueden utilizar como *nombre-tipo-diferenciado* (SQLSTATE 42939). Los nombres son SOME, ANY, ALL, NOT, AND, OR, BETWEEN, NULL, LIKE, EXISTS, IN, UNIQUE, OVERLAPS, SIMILAR, MATCH y los operadores de comparación.

Si se especifica un *nombre-tipo-diferenciado* de dos partes, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

tipo-datos-fuente
Especifica el tipo de datos utilizado como base para la representación interna del tipo diferenciado. El tipo de datos debe ser un tipo de datos incorporado. Para obtener más información sobre los tipos de datos incorporados, consulte "CREATE TABLE". El tipo de datos fuente no puede ser de tipo XML o un tipo ARRAY (SQLSTATE 42601). Para la portabilidad de las aplicaciones entre plataformas, utilice los siguientes nombres de tipos de datos recomendados:

- DOUBLE o REAL en lugar de FLOAT
- DECIMAL en lugar de NUMERIC
- VARCHAR, BLOB o CLOB en lugar de LONG VARCHAR
- VARGRAPHIC o DBCLOB en lugar de LONG VARGRAPHIC

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 o DBCLOB, 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.

  El privilegio EXECUTE en todas las funciones generadas automáticamente durante la sentencia CREATE TYPE (diferenciado) se otorga a PUBLIC.
- La creación de un tipo diferenciado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN para el esquema se otorga a PUBLIC.
- Se generan las funciones siguientes para convertir el tipo de datos utilizando el tipo de fuente como fuente 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 25). El tipo del valor de retorno de esta función incluirá los parámetros dados en la sentencia CREATE TYPE (diferenciado). La función para convertir del tipo de fuente al tipo diferenciado tendrá un parámetro de entrada cuyo tipo es el tipo de fuente incluyendo sus parámetros. Por ejemplo,

```
CREATE TYPE T_SHOESIZE AS CHAR(2)
  WITH COMPARISONS

CREATE TYPE T_MILES AS DOUBLE
  WITH COMPARISONS
```

generará las funciones siguientes:

```
FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)

FUNCTION T_SHOESIZE (CHAR (2))
RETURNS T_SHOESIZE

FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE

FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

El esquema de las funciones de conversión generadas es el mismo que el esquema del tipo diferenciado. No debe existir ninguna otra función con este nombre y con la misma signatura en la base de datos (SQLSTATE 42710).

La tabla siguiente ofrece los nombres de las funciones para la conversión del tipo diferenciado al tipo de fuente y del tipo de fuente al tipo diferenciado para todos los tipos de datos predefinidos.

Tabla 25. 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*)
CLOB	*nombre-tipo-diferenciado*	CLOB (*n*)	*nombre-tipo-diferenciado*
	CLOB	*nombre-tipo-diferenciado*	CLOB (*n*)

Tabla 25. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
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*)
DBCLOB	*nombre-tipo-diferenciado*	DBCLOB (*n*)	*nombre-tipo-diferenciado*
	DBCLOB	*nombre-tipo-diferenciado*	DBCLOB (*n*)
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
DECFLOAT	*nombre-tipo-diferenciado*	DECFLOAT(*n*)	*nombre-tipo-diferenciado*
	DECFLOAT	*nombre-tipo-diferenciado*	DECFLOAT(*n*)
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

Tabla 25. Funciones CAST en tipos diferenciados (continuación)

Nombre de tipo de fuente	Nombre de función	Parámetro	Tipo de retorno
TIMESTAMP	*nombre-tipo-diferenciado*	TIMESTAMP	*nombre-tipo-diferenciado*
	TIMESTAMP	*nombre-tipo-diferenciado*	TIMESTAMP

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 fuente en las funciones incorporadas adecuadas. En particular, observe que es posible registrar funciones definidas por el usuario que tienen su fuente en funciones de columna incorporadas.

Cuando se crea un tipo diferenciado utilizando la cláusula WITH COMPARISONS, se crean operadores de comparación generados por el sistema. La creación de esos operadores de comparación generará entradas en la vista de catálogo SYSCAT.ROUTINES de las nuevas funciones.

El nombre de esquema del tipo diferenciado debe estar incluido en la vía de acceso de SQL o en la opción FUNCPATH BIND para que la utilización de estos operadores y de las funciones de conversión en las sentencias de SQL sea satisfactoria.

- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - CREATE DISTINCT TYPE puede especificarse en vez de CREATE TYPE
    - Los tipos de datos LONG VARCHAR y LONG VARGRAPHIC y las funciones de conversión están soportados pero han quedado obsoletos y se pueden eliminar en un release futuro. La cláusula WITH COMPARISONS sigue sin dar soporte a los tipos de datos LONG VARCHAR y LONG VARGRAPHIC.

Ejemplos

Ejemplo 1: Cree un tipo diferenciado llamado SHOESIZE que se base en un tipo de datos INTEGER.

```
CREATE 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 TYPE MILES AS DOUBLE WITH COMPARISONS
```

Esto también dará como resultado la creación de operadores de comparación (=, <>, <, =, >, >=) y que la función de conversión DOUBLE(MILES) devuelva DOUBLE y la función de conversión MILES(DOUBLE) devuelva MILES.

CREATE 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 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).

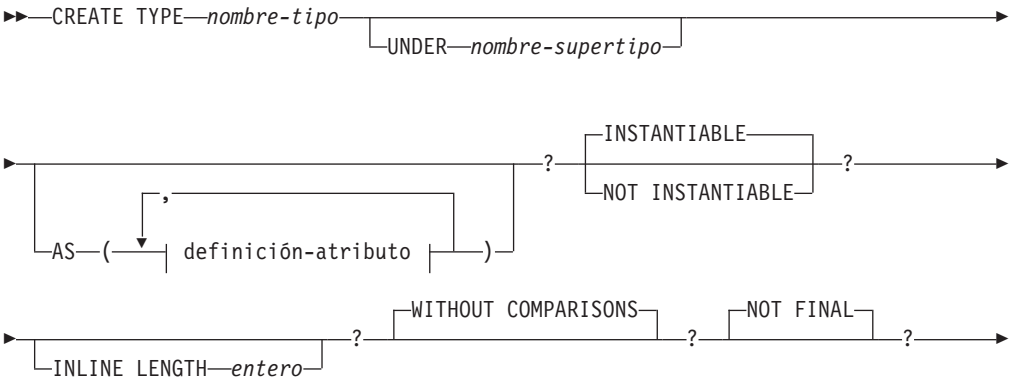
Autorización

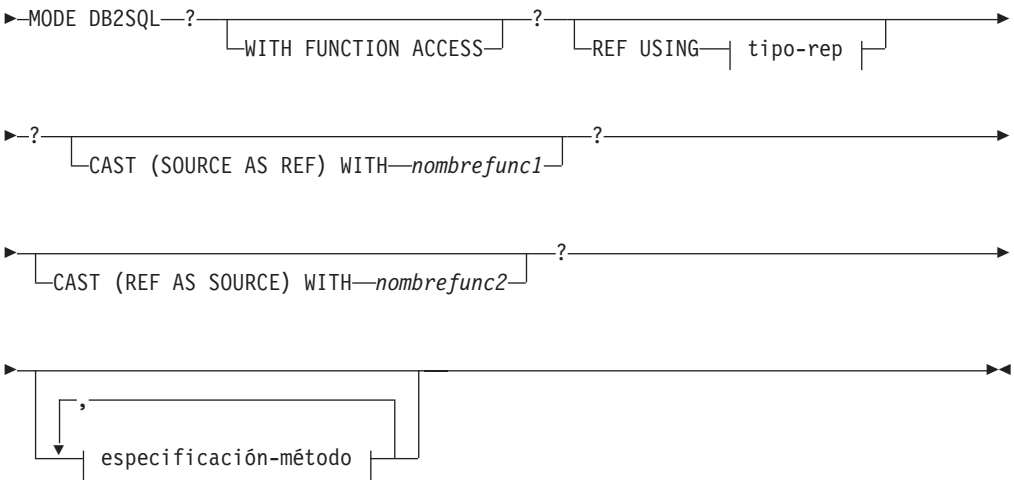
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización IMPLICIT_SCHEMA para la base de datos, si el nombre de esquema del tipo no hace referencia a un esquema existente
- Privilegio CREATEIN para el esquema, si el nombre de esquema del tipo hace referencia a un esquema existente
- Autorización SYSADM o DBADM

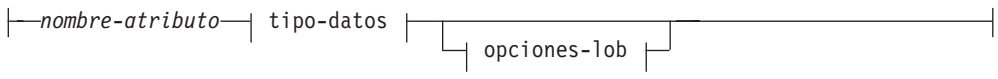
Si se especifica UNDER y el ID de autorización de la sentencia no es el mismo que el del propietario del tipo raíz de la jerarquía de tipos, se necesita autorización SYSADM o DBADM.

Sintaxis

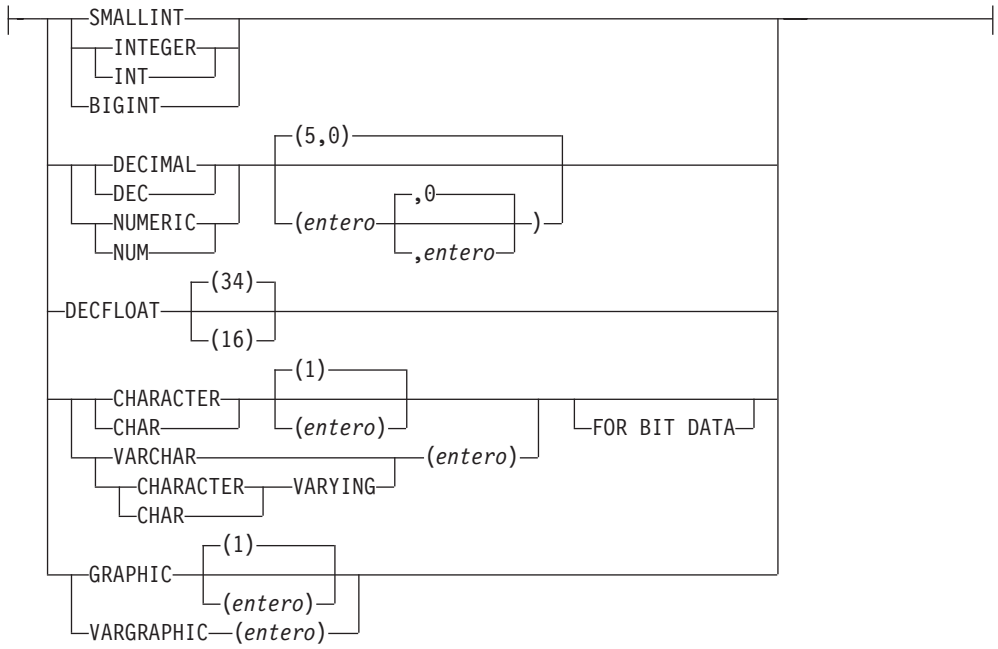




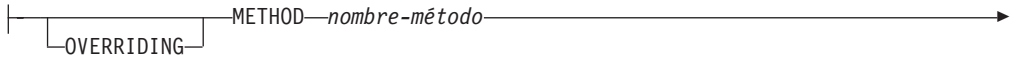
definición-atributo:

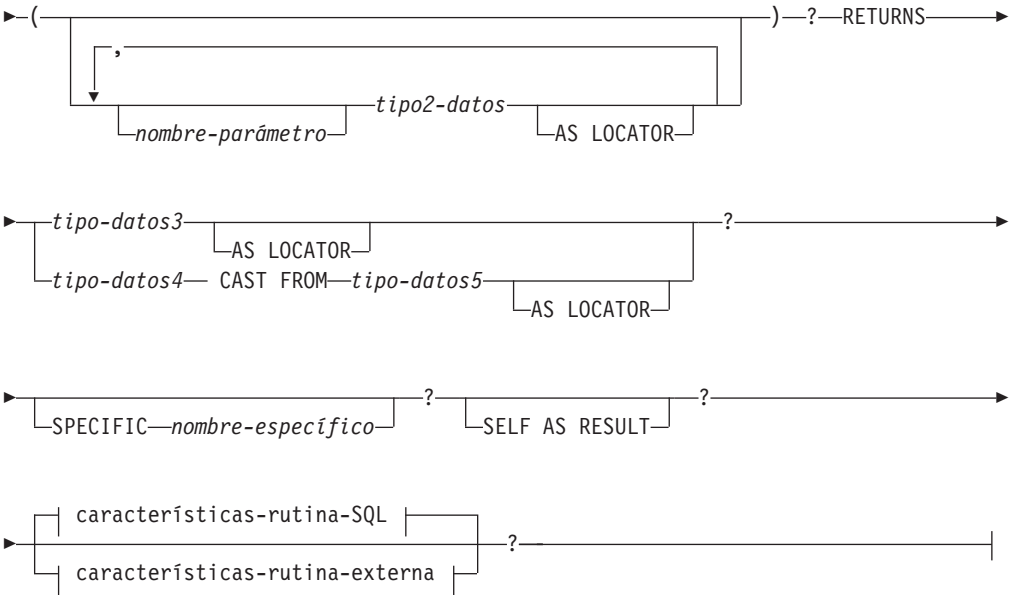


tipo-rep:

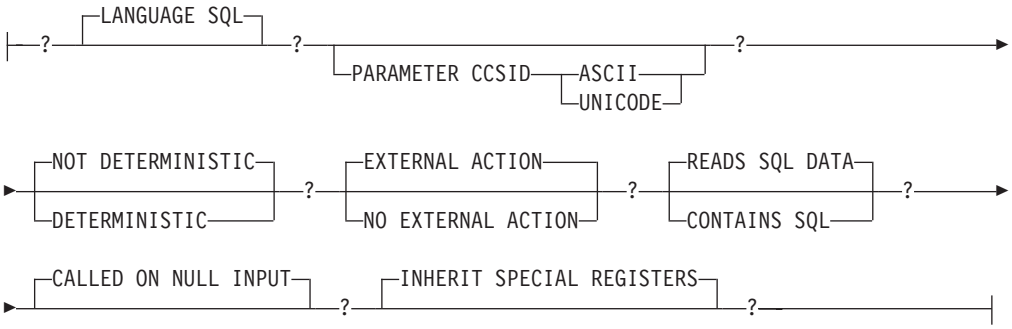


especificación-método:

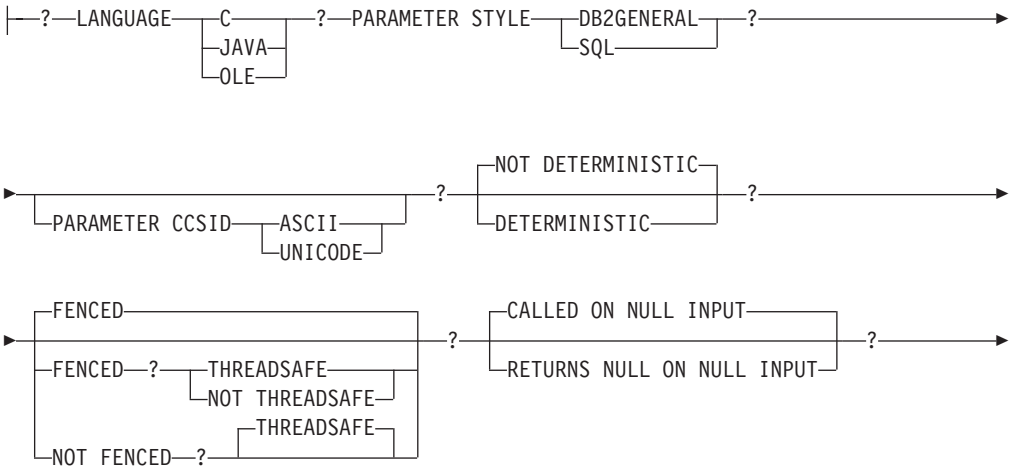


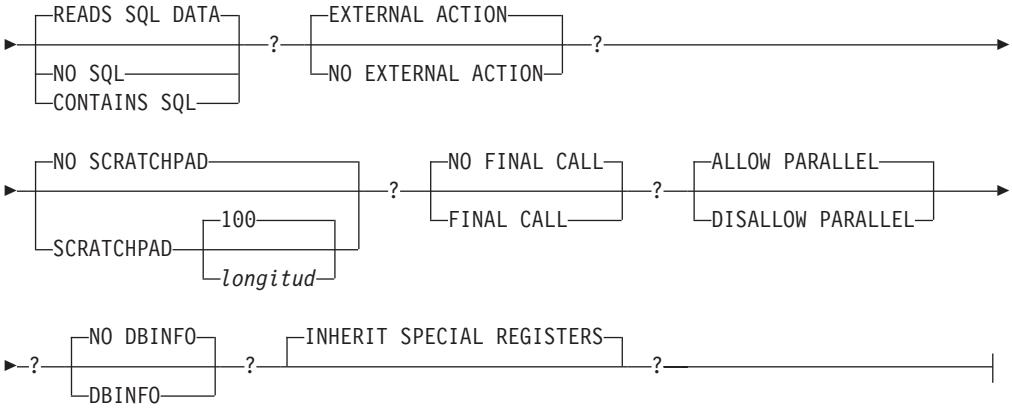


características-rutina-SQL:



características-rutina-externa:





Descripción

nombre-tipo

Indica el tipo. El nombre, incluido el calificador implícito o explícito, no debe identificar ningún otro tipo (incorporado, estructurado o diferenciado) que ya exista en el servidor actual. El nombre no calificado no debe ser el mismo que el nombre de un tipo de datos incorporado, BINARY, VARBINARY o BOOLEAN (SQLSTATE 42918). El nombre no calificado tampoco debe ser ARRAY, INTERVAL ni ROWID. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados.

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

Si se especifica un nombre compuesto de dos partes *nombre-tipo*, el nombre de esquema no puede empezar por los caracteres 'SYS' (SQLSTATE 42939).

UNDER *nombre-supertipo*

Especifica que este tipo estructurado es un subtipo por debajo del *nombre-supertipo* especificado. El *nombre-supertipo* debe identificar un tipo estructurado existente (SQLSTATE 42704). Si *nombre-supertipo* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. El tipo estructurado incluye todos los atributos del supertipo seguidos de los atributos adicionales que se proporcionan en la *definición-atributo*.

definición-atributo

Define los atributos del tipo estructurado.

nombre-atributo

El nombre de un atributo. El *nombre-atributo* no puede ser el mismo que el de otro atributo de este tipo estructurado o un supertipo de este tipo estructurado (SQLSTATE 42711).

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

tipo-datos
El tipo de datos del atributo. Es uno de los tipos de datos que s listan en "CREATE TABLE", que no sean XML (SQLSTATE 42601). El tipo de datos debe identificar un tipo de datos existente (SQLSTATE 42704). Si *tipo-datos* está especificado sin un nombre de esquema, el tipo se resuelve realizando una búsqueda en los esquemas de la vía de acceso de SQL. Encontrará la descripción de diversos tipos de datos en "CREATE TABLE". Si el tipo de datos del atributo es un tipo de referencia, el tipo destino de la referencia debe ser un tipo estructurado existente o se crea mediante esta sentencia (SQLSTATE 42704).

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

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

INSTANTIABLE o NOT INSTANTIABLE
Determina si se puede crear una instancia del tipo estructurado. El no poder crear una instancia de un tipo estructurado tiene estas consecuencias:

- no se genera ninguna función constructora para un tipo para el que no se pueden crear instancias
- el tipo que no admite creación de instancias no puede utilizarse como tipo de una tabla o vista (SQLSTATE 428DP)
- el tipo que no admite creación de instancias se puede utilizar como tipo de una columna (sólo pueden insertarse en la columna valores nulos o instancias de subtipos que permiten la creación de instancias.

Para crear instancias de un tipo que no permite crear instancias, se deben crear subtipos que permiten crear instancias. Si se especifica NOT INSTANTIABLE, no se puede crear ninguna instancia del nuevo tipo.

INLINE LENGTH *entero*
Esta opción indica el tamaño máximo, en bytes, de una instancia de columna de tipo estructurado para su almacenamiento "inline" con el resto de valores de la fila de una tabla. Las instancias de un tipo estructurado o de sus subtipos que son mayores que la longitud "inline" especificada se almacenan separadamente de la fila de la tabla base, de forma similar a como se manejan los valores LOB.

Si el valor especificado de INLINE LENGTH es menor que el tamaño del resultado de la función constructora para el tipo recién creado (32 bytes más 10 bytes por atributo) y menor que 292 bytes, se produce un error (SQLSTATE 429B2). Observe que el número de atributos incluye todos los atributos heredados a partir del supertipo del tipo.

El valor INLINE LENGTH del tipo, ya sea especificado explícitamente o tomado por omisión, es la longitud "inline" por omisión de las columnas que hacen uso del tipo estructurado. Este valor por omisión se puede alterar temporalmente por otro valor al crear la tabla.

El valor INLINE LENGTH no es aplicable cuando el tipo estructurado se utiliza como tipo de una tabla con tipo.

El valor por omisión de INLINE LENGTH para un tipo estructurado es calculado por el sistema. En 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 26.

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 26. Número de bytes para los tipos de datos de atributos

Tipo de datos de atributo	Número de bytes
DECIMAL	La parte integral de (*p*/2)+1, donde *p* es la precisión
DECFLOAT(*n*)	Si *n* es 16, el número de bytes es 8; si *n* es 34, el número de bytes es 16
CHAR (*n*)	*n*
VARCHAR (*n*)	*n*
GRAPHIC (*n*)	*n* * 2
VARGRAPHIC (*n*)	*n* * 2
TIMESTAMP	10
Tipo LOB	Cada atributo LOB tiene un descriptor de LOB, en la instancia del tipo estructurado, que apunta a la ubicación del valor real. El tamaño del descriptor varía en función de la longitud máxima definida para el atributo LOB (consulte la Tabla 27 en la página 620).
Tipo diferenciado	Longitud del tipo de fuente del tipo diferenciado
Tipo de referencia	Longitud del tipo de datos incorporados en la que se basa el tipo de referencia.
Tipo estructurado	inline_length(*tipo_atributo*)

Tabla 27. Tamaño del descriptor LOB como una función de la longitud máxima LOB

Longitud máxima de LOB	Tamaño del descriptor de LOB
1024	72
8192	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

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 REAL, FLOAT, DECFLOAT, BLOB, CLOB, DBCLOB, tipo matriz o tipo estructurado y debe tener una longitud igual o inferior a 32 672 bytes (SQLSTATE 42613).

Si esta cláusula no se especifica para el tipo raíz de una jerarquía de tipos estructurados, entonces se supone REF USING VARCHAR(16) FOR BIT DATA.

CAST (SOURCE AS REF) WITH *nombrefunc1*
Define el nombre de la función, generada por el sistema, que convierte un valor cuyo tipo de datos es *tipo-rep* al tipo de referencia de este tipo estructurado. No debe especificarse un nombre de esquema como parte de *nombrefunc1* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc1* es *nombre-tipo* (el nombre del tipo estructurado). El esquema no debe contener ya una signatura de función que coincida con *nombrefunc1(tipo-rep)* (SQLSTATE 42710).

CAST (REF AS SOURCE) WITH *nombrefunc2*
Define el nombre de la función, generada por el sistema, que convierte un valor de tipo de referencia para este tipo estructurado al tipo de datos *tipo-rep*. No se debe especificar un nombre de esquema como parte de *nombrefunc2* (SQLSTATE 42601). La función de conversión se crea en el mismo esquema que el tipo estructurado. Si no se especifica la cláusula, el valor por omisión para *nombrefunc2* es *tipo-rep* (el nombre del tipo de representación).

especificación-método
Define los métodos correspondientes a este tipo. Para utilizar un método es necesario proporcionarle un cuerpo mediante una sentencia CREATE METHOD (SQLSTATE 42884).

OVERRIDING
Especifica que el método que está definiéndose altera temporalmente a un método de un supertipo del tipo que está definiéndose. La alteración temporal permite volver a implementar métodos en los subtipos y, por lo tanto, ofrece funciones más específicas. La alteración temporal no recibe soporte para los siguientes tipos de métodos:

- Métodos de tabla y fila
- Métodos externos declarado con PARAMETER STYLE JAVA
- Métodos que puede utilizarse como predicados en una extensión de índice
- Métodos de mutación o de observador generados por el sistema

Si se intenta alterar temporalmente uno de estos métodos, se generará un error (SQLSTATE 42745).

Si un método es un método de alteración temporal válido, deberá existir un método original para uno de los supertipos adecuados del tipo que está definiéndose y deberán existir las relaciones siguientes entre el método de alteración temporal y el método original:

- El nombre de método del método que está definiéndose y el método original son equivalentes.
- El método que está definiéndose y el método original tienen el mismo número de parámetros.
- El tipo de datos de cada parámetro del método que está definiéndose y el tipo de datos de los parámetros correspondientes del método original son idénticos. Este requisito excluye al parámetro SELF implícito.

Si no existe un método original con estas características, se devolverá un error (SQLSTATE 428FV).

El método de alteración temporal hereda los siguientes atributos del método original:

- Lenguaje
- Indicación de determinismo
- Indicación de acción externa
- Una indicación que especifique si este método debe o no llamarse si alguno de sus argumentos es el valor nulo
- Conversión del resultado (si se ha especificado en el método original)
- Indicación SELF AS RESULT
- La indicación de acceso a los datos de SQL o CONSTAINS SQL
- Para los métodos externos:
  - Estilo del parámetro

- 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).

tipo2-datos
Especifica el tipo de datos de cada parámetro. Debe especificarse una entrada de la lista para cada parámetro que el método espera recibir. No se permiten más de 90 parámetros, incluido el parámetro implícito SELF. Si se excede este límite, se produce un error (SQLSTATE 54023).

Puede especificar abreviaturas y tipos de datos SQL que se pueden especificar como un tipo de columna en la sentencia CREATE TABLE y que tienen equivalentes en el lenguaje en el que se está utilizando para escribir el método. Para obtener detalles sobre la correlación entre tipos de datos SQL y tipos de datos de lenguaje principal, consulte el tema que pertenece a su lenguaje de la lista de temas relacionados que aparece más abajo.

Nota: Si el tipo de datos SQL en cuestión es un tipo estructurado, no existe una correlación por omisión con un tipo de datos de sistema principal. Se debe utilizar una función de transformación definida por el usuario para crear una correlación entre el tipo estructurado y el tipo de datos de sistema principal.

DECIMAL (o NUMERIC) y coma flotante decimal no son válidos con LANGUAGE C y OLE (SQLSTATE 42815).

No se pueden utilizar tipos de datos XML (SQLSTATE 42815).

Se puede especificar REF, pero no tiene un ámbito definido. Dentro del cuerpo del método, se puede utilizar un tipo de referencia en una expresión de vía de acceso sólo si primero se convierte el tipo para que tenga un ámbito. Similarmente, una referencia devuelta por un método

se puede utilizar en una expresión de vía de acceso sólo si primero se convierte para que tenga un ámbito.

AS LOCATOR
Para los tipos LOB o tipos diferenciados basados en un tipo LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB al método, en lugar de pasarle el valor real. Esto reduce considerablemente el número de bytes que se pasan al método y puede también mejorar el rendimiento, especialmente cuando el método sólo necesite unos pocos bytes.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Si el método que está declarándose altera temporalmente a otro método, la indicación AS LOCATOR del parámetro deberá coincidir exactamente con la indicación AS LOCATOR del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

Si el método que está declarándose altera temporalmente a otro método, la indicación FOR BIT DATA de cada parámetro deberá coincidir exactamente con la indicación FOR BIT DATA del correspondiente parámetro del método alterado temporalmente (SQLSTATE 428FV).

RETURNS
Esta cláusula obligatoria identifica el resultado del método.

tipo-datos3
Especifica el tipo de datos del resultado del método. En este caso, son válidas las mismas consideraciones que para los parámetros de métodos, 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 devuelve un error (SQLSTATE 42880).

Puesto que la longitud, la precisión o la escala de *tipo-datos4* puede inferirse de *tipo-datos5*, no es necesario (pero está permitido) especificar la longitud, la precisión o la escala de los tipos con parámetros especificados para *tipo-datos4*. En cambio, se pueden utilizar paréntesis vacíos como, por ejemplo, VARCHAR(). FLOAT() no puede utilizarse (SQLSTATE 42601), ya que el valor del parámetro indica tipos de datos distintos (REAL o DOUBLE).

Un tipo distinto no es válido como el tipo especificado en *tipo-datos5* (SQLSTATE 42815). XML no es válido como el tipo especificado en *tipo-datos4* o *tipo-datos5* (SQLSTATE 42815).

La operación de conversión también está sujeta a comprobaciones durante la ejecución, que pueden dar como resultado errores de conversión.

AS LOCATOR
Para tipos LOB o tipos diferenciados que están basados en tipos LOB, se puede añadir la cláusula AS LOCATOR. Esto indica que se debe pasar un localizador de LOB desde el método, en lugar del valor real.

Se produce un error (SQLSTATE 42601) si se especifica AS LOCATOR para un tipo que no sea LOB o para un tipo diferenciado basado en un LOB.

Si el método es FENCED o si LANGUAGE es SQL, no se puede especificar la cláusula AS LOCATOR (SQLSTATE 42613).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse la cláusula FOR BIT DATA (SQLSTATE 428FV).

SPECIFIC *nombre-específico*
Proporciona un nombre exclusivo para la instancia del método que se está definiendo. Este nombre específico puede utilizarse al crear el cuerpo del método o al 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 llamará al método definido por el usuario en una clase Java.

OLE
Esto significa que el gestor de bases de datos invocará el método definido por el usuario como si fuera un método expuesto por un objeto de automatización OLE. El método debe ajustarse a los tipos de datos de automatización OLE y al mecanismo de invocación, tal como se describe en la publicación *OLE Automation Programmer's Reference*.

Sólo se da soporte a LANGUAGE OLE para métodos definidos por el usuario almacenados en 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 las convenciones para pasar parámetros desde métodos externos y devolver el valor desde éstos, que están definidos como un método en una claseJava. 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 un tipo que no es determinante es aquel que hace referencia a registros especiales, variables globales o funciones no determinantes de un modo que afecta al tipo de resultado.

FENCED o NOT FENCED
Esta cláusula especifica si el método se considera "seguro" (NOT FENCED) o no (FENCED) para ejecutarse en el proceso o espacio de direcciones del entorno operativo del gestor de bases de datos.

Si un método se ha registrado como FENCED, el gestor de bases de datos protege sus recursos internos (por ejemplo, los almacenamientos intermedios) para que el método no pueda acceder a ellos. La mayoría de los métodos tienen la posibilidad de ejecutarse como FENCED o NOT FENCED. En general, un método que se ejecute como FENCED no funcionará tan bien como otro método similar que se ejecute como NOT FENCED.

PRECAUCIÓN:
La utilización de NOT FENCED para métodos no probados debidamente puede comprometer la integridad de 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 siguientes características:

- *longitud*, si se especifica, define el tamaño en bytes de la memoria de trabajo; este valor debe estar comprendido entre 1 y 32.767 (SQLSTATE 42820). El valor por omisión es 100.
- El valor de inicialización consta sólo de ceros hexadecimales.
- Su ámbito es la sentencia de SQL. Existe una memoria de trabajo para cada referencia al método externo contenida en la sentencia de SQL.

Por tanto, si el método X de la sentencia siguiente se define con la palabra clave SCRATCHPAD, se asignarán tres memorias de trabajo.

```
SELECT A, X..(A) FROM TABLEB
   WHERE X..(A) > 103 OR X..(A) < 19
```

Si se especifica ALLOW PARALLEL o se toma por omisión, el ámbito es diferente del anterior. Si el método se ejecuta en varias particiones de base de datos, se asignará un área reutilizable en cada partición de base de datos en la que se procese el método, para cada referencia al método en la sentencia de SQL. De 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 llamada son:

- Llamada normal: Se pasan los argumentos SQL y se espera la devolución de un resultado.
- Primera llamada: es la primera llamada al método externo para esta referencia específica al método contenida en esta sentencia de SQL específica. La primera llamada es una llamada normal.
- Llamada final: es una llamada final al método externo para permitir que el método libere recursos. La llamada final no es una llamada normal. Esta llamada final tiene lugar en las siguientes circunstancias:
  - Fin de sentencia: Esta situación se produce cuando se cierra el cursor, en el caso de sentencias orientadas a cursor, o cuando la sentencia termina su ejecución de otra manera.
  - Fin de transacción: Este caso se produce cuando no se da un fin de sentencia normal. Por ejemplo, cuando por alguna razón la lógica de una aplicación ignora el cierre del cursor.

  Si se produce una operación de confirmación mientras está abierto un cursor definido como WITH HOLD, se realiza una llamada final en el cierre subsiguiente del cursor o al final de la aplicación.

Si se especifica NO FINAL CALL, no se pasa al método externo ningún argumento que especifique el tipo de llamada, y no se realiza ninguna llamada final.

ALLOW PARALLEL o DISALLOW PARALLEL

Esta cláusula opcional especifica si, para una referencia individual al método, puede paralelizarse la invocación del método. En general, las invocaciones de la mayoría de los métodos escalares pueden paralelizarse, pero puede haber métodos (tales como los que dependen de una sola copia de una memoria de trabajo) que no puedan. Si se especifica ALLOW PARALLEL o DISALLOW PARALLEL para un método, DB2 aceptará esta especificación.

Deben tenerse en cuenta las cuestiones siguientes al determinar qué palabra clave es la adecuada para el método:

- ¿Son todas las invocaciones del método completamente independientes las unas de las otras? En caso afirmativo, especifique ALLOW PARALLEL.
- ¿Se actualiza la memoria de trabajo con cada invocación del método, proporcionando valores que son de interés para la siguiente invocación (el incremento de un contador, por ejemplo)? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Existe alguna acción externa que realice el método que sólo se deba llevar a cabo en una partición de base de datos? En caso afirmativo, especifique DISALLOW PARALLEL o acepte el valor por omisión.
- ¿Se utiliza la memoria de trabajo, pero requiere realizar algún proceso de inicialización costoso un número mínimo de veces? En caso afirmativo, especifique ALLOW PARALLEL.

En cualquier caso, el cuerpo de todos los métodos externos debe encontrarse en un directorio disponible en todas las particiones de base de datos.

El diagrama de sintaxis indica que el valor por omisión es ALLOW PARALLEL. Sin embargo, el valor por omisión es DISALLOW PARALLEL si en la sentencia se especifican una o más de las opciones siguientes:

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO o DBINFO

Esta cláusula opcional especifica si se pasará cierta información específica conocida por DB2 al método como un argumento en tiempo de una invocación adicional (DBINFO) o no (NO DBINFO). NO DBINFO es el valor por omisión. DBINFO no puede utilizarse para LANGUAGE OLE (SQLSTATE 42613). Si el método que está definiéndose altera temporalmente a otro método, no podrá especificarse esta cláusula (SQLSTATE 428FV).

Si se especifica DBINFO, al método se le envía una estructura que contiene la siguiente información:

- Nombre de base de datos - el nombre de la base de datos conectada actualmente.
- ID de aplicación - ID de aplicación exclusivo que se establece para cada conexión con la base de datos.
- ID de autorización de aplicación - el ID de autorización de ejecución de la aplicación, sin tener en cuenta los métodos anidados existentes entre este método y la aplicación.
- Página de códigos - identifica la página de códigos de la base de datos.
- Nombre de esquema - si se dan las mismas condiciones que para el Nombre de tabla, contiene el nombre del esquema; de lo contrario, está en blanco.
- Nombre de tabla - si 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 - exactamente bajo las mismas condiciones que para el Nombre de tabla, contiene el nombre de la columna que se está actualizando o insertando; de lo contrario está en blanco.
- Versión/release de base de datos - identifica la versión, release y nivel de modificación del servidor de bases de datos invocador del método.
- Plataforma - contiene el tipo de plataforma del servidor.
- Números de columna del resultado del método de tabla - no es aplicable a métodos.

INHERIT SPECIAL REGISTERS

Esta cláusula opcional especifica que los registros especiales del método heredarán sus valores iniciales de la sentencia que realiza la llamada. Para los cursores, los valores iniciales se heredan del valor de tiempo que corresponde a la apertura del cursor.

Al proceso que realiza la llamada del método no se le devolverá ninguno de los cambios que se han realizado en los registros especiales.

Algunos registros especiales, como los registros especiales de hora y fecha,reflejan una propiedad de la sentencia que se está ejecutando y, por consiguiente, nunca se heredan de quien llama.

Notas

- La creación de un tipo estructurado con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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 toma en consideración en la comparación.

- Funciones de conversión

  Se generan dos funciones de conversión para convertir entre el tipo de referencia generado REF(*nombre-tipo*) y el tipo de datos subyacente de este tipo de referencia.

  - El nombre de la función para convertir el tipo subyacente al tipo de referencia es el *nombrefunc1* implícito o explícito.

    El formato de esta función es:

    ```
    CREATE FUNCTION nombrefunc1 (tipo-rep)
      RETURNS REF(nombre-tipo) ...
    ```

  - El nombre de la función para convertir el tipo de referencia al tipo subyacente del tipo de referencia es el *nombrefunc2* implícito o explícito.

    El formato de esta función es:

    ```
    CREATE FUNCTION nombrefunc2 ( REF(nombre-tipo) )
      RETURNS tipo-rep ...
    ```

  Para algunos tipos-rep, existen funciones de conversión adicionales generadas con *nombrefunc1* para manejar las conversiones desde constantes.

  - Si *tipo-rep* es SMALLINT, la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 (INTEGER)
      RETURNS REF(nombre-tipo)
    ```

  - Si *tipo-rep* es CHAR(n), la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 ( VARCHAR(n))
      RETURNS REF(nombre-tipo)
    ```

  - Si *tipo-rep* es GRAPHIC(n), la función de conversión adicional generada tiene el formato:

    ```
    CREATE FUNCTION nombrefunc1 (VARGRAPHIC(n))
      RETURNS REF(nombre-tipo)
    ```

  El nombre de esquema del tipo estructurado debe estar incluido en la vía de acceso de SQL para que la utilización de estos operadores y de las funciones de conversión en las sentencias de SQL sea satisfactoria.

- Función constructora

  La función constructora se genera para permitir la creación de una nueva instancia del tipo. Esta nueva instancia tendrá un valor nulo para todos los atributos del tipo, incluidos los atributos que se heredan de un supertipo.

  El formato de la función constructora generada es:

  ```
  CREATE FUNCTION nombre-tipo ( )
    RETURNS nombre-tipo
   ...
  ```

  Si se especifica NOT INSTANTIABLE, no se genera ninguna función constructora.

– Métodos

- Métodos observadores

  Se define un método observador para cada atributo del tipo estructurado. Para cada atributo, el método observador devuelve el tipo del atributo. Si el asunto es nulo, el método observador devuelve un valor nulo del tipo de atributo.

Por ejemplo, los atributos de una instancia del tipo estructurado ADDRESS se pueden observar utilizando `C1..STREET`, `C1..CITY`, `C1..COUNTRY` y `C1..CODE` .

La signatura de método del método observador generado es como si la sentencia siguiente se hubiera ejecutado:

```
CREATE TYPE nombre-tipo
    ...
  METHOD nombre-atributo()
    RETURNS tipo-atributo
```

donde *nombre-tipo* es el nombre del tipo estructurado.

- Métodos mutadores

  Se define un método mutador preservador del tipo para cada atributo del tipo estructurado. Utilice métodos mutadores para cambiar atributos dentro de una instancia de un tipo estructurado. Para cada atributo, el método mutador devuelve una copia del asunto modificada mediante la asignación del argumento al atributo mencionado de la copia.

  Por ejemplo, una instancia del tipo estructurado ADDRESS se puede mutar utilizando `C1..CODE('M3C1H7')`. Si el sujeto es nulo, el método mutador emite un error (SQLSTATE 2202D).

  La signatura del método mutador generado es como si se hubiera ejecutado la sentencia siguiente:

```
CREATE TYPE nombre-tipo
      ...
   METHOD nombre-atributo (tipo-atributo)
     RETURNS nombre-tipo
```

  Si el tipo de datos del atributo es SMALLINT, REAL, CHAR o GRAPHIC, se genera un método mutador adicional para dar soporte a la mutación utilizando constantes:

  - Si *tipo-atributo* es SMALLINT, el mutador adicional da soporte a un argumento de tipo INTEGER.
  - Si *tipo-atributo* es REAL, el mutador adicional da soporte a un argumento de tipo DOUBLE.
  - Si *tipo-atributo* es CHAR, el mutador adicional da soporte a un argumento de tipo VARCHAR.
  - Si *tipo-atributo* es GRAPHIC, el mutador adicional da soporte a un argumento de tipo VARGRAPHIC.

- Si el tipo estructurado se utiliza como tipo de columna, la longitud máxima de una instancia del tipo es 1 GB durante la ejecución (SQLSTATE 54049).

- Cuando se crea un nuevo subtipo para un tipo estructurado existente (para utilizarlo como tipo de columna), se deben reexaminar y actualizar según sea necesario las funciones de transformación existentes que dan soporte a los tipos estructurados asociados. Tanto si el nuevo tipo está en la misma jerarquía que un tipo determinado o en la jerarquía de un tipo anidado, es probable que la función de transformación asociada a este tipo deba modificarse para incluir algunos o todos los nuevos atributos originados por el nuevo subtipo. En términos generales, puesto que es el conjunto de funciones de transformación que se asocia a un tipo determinado (o jerarquía de tipo) que permite a UDF y a la aplicación cliente acceder al tipo estructurado, las funciones de transformación deben escribirse de modo que se dé soporte a *todos* los atributos de una jerarquía compuesta determinada (es decir, incluyendo el cierre transitivo de todos los subtipos y sus tipo estructurados anidados).

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. El privilegio EXECUTE en todas las funciones generadas automáticamente durante la sentencia CREATE TYPE (diferenciado) se otorga a PUBLIC.
  - Cuando en una sentencia de SQL se utiliza un método externo, el usuario que define el método debe disponer de privilegio EXECUTE para cualquiera de los paquetes que el método utiliza.
- En un entorno de 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.
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para 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

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
```

CREATE TYPE MAPPING

La sentencia CREATE TYPE MAPPING define una correlación entre los tipos de datos siguientes:

- El tipo de datos de una columna de una tabla o una vista de fuente de datos que se va a definir para una base de datos federada
- Un tipo de datos correspondiente que ya está definido para la base de datos federada

La correlación puede asociar el tipo de datos de base de datos federada con un tipo de datos de:

- Una fuente de datos especificada
- Un rango de fuentes de datos; por ejemplo, todas las fuentes de datos de un tipo y versión en particular

Una correlación de tipo de datos sólo debe crearse si una correlación existente no es adecuada.

Si se pueden aplicar varias correlaciones de tipos al crear un apodo o una tabla (DDL transparente), se aplicará la más reciente.

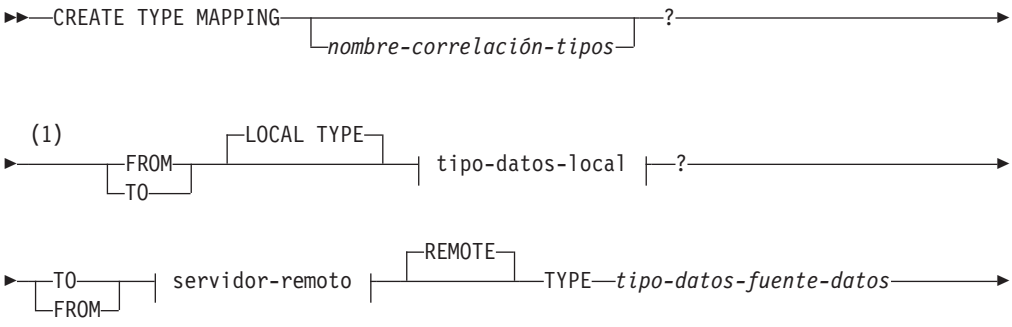
Invocación

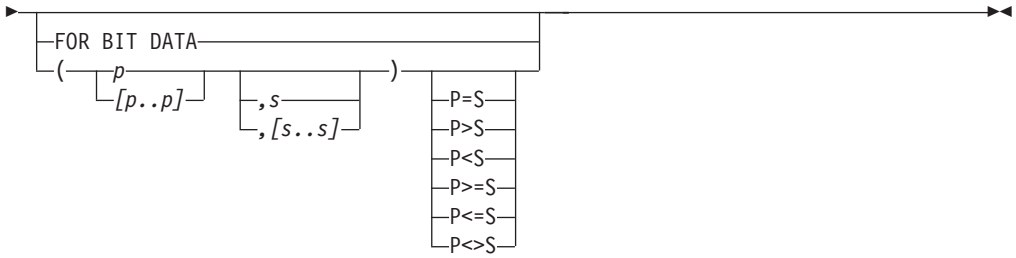
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

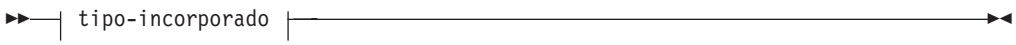
Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM.

Sintaxis

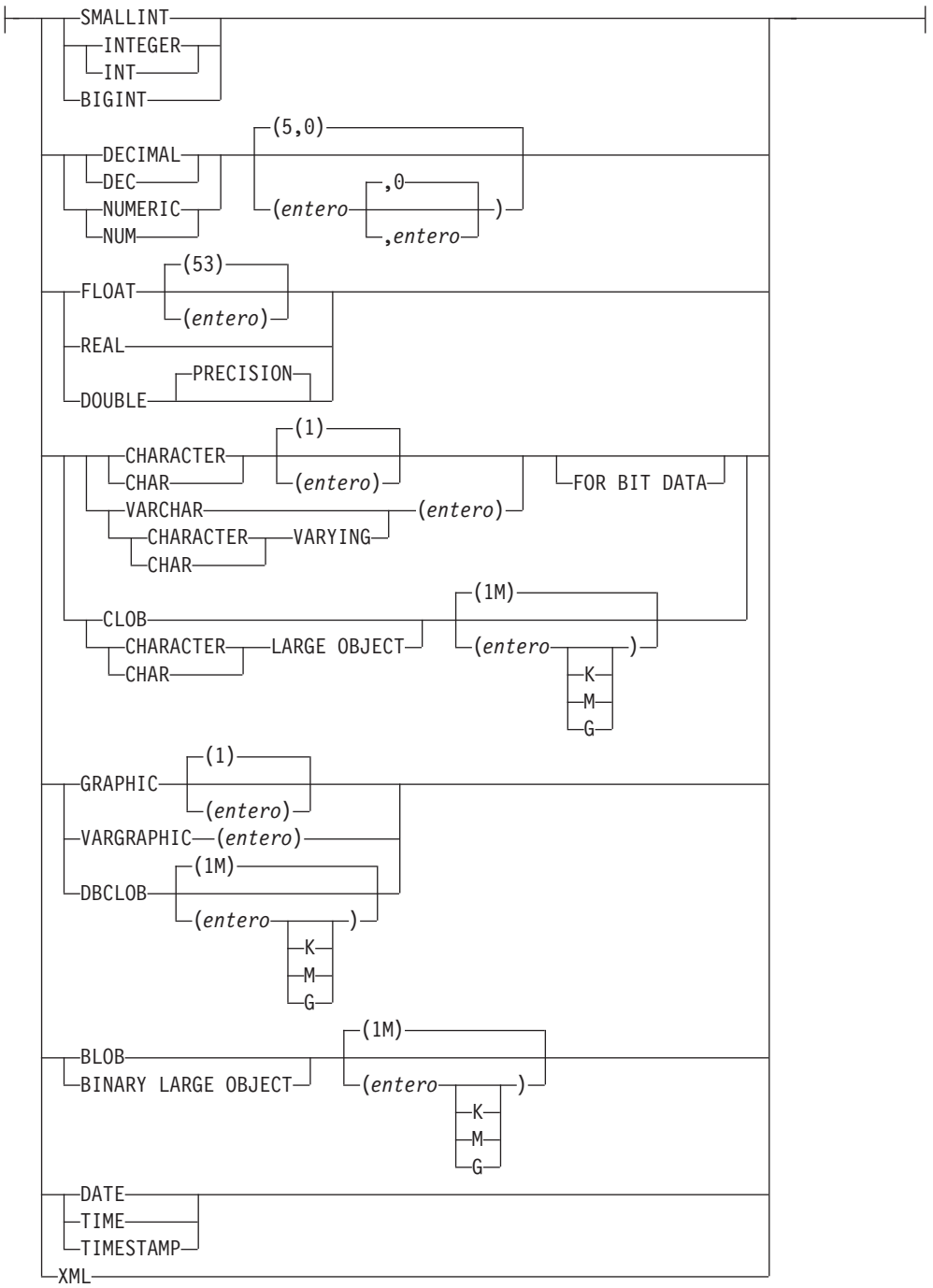




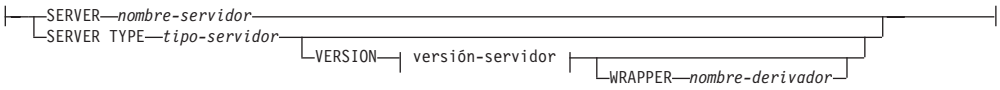
tipo-datos-locales:



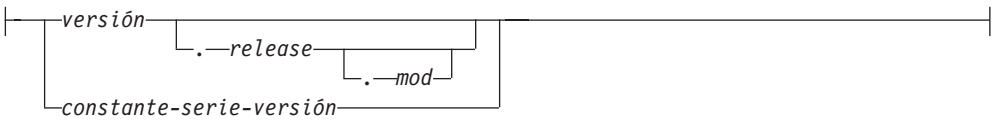
tipo-incorporado:



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 utiliza la longitud por omisión para el tipo de datos.

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

El *tipo-datos-local* no puede ser DECFLOAT, LONG VARCHAR, LONG VARGRAPHIC o un tipo definido por el usuario (SQLSTATE 42611).

SERVER *nombre-servidor*
Nombra la fuente de datos para la que se define *tipo-datos-fuente-datos*.

SERVER TYPE *tipo-servidor*
Identifica el tipo de fuente de datos para el que se define el *tipo-datos-fuente-datos*.

VERSION
Identifica la versión de la fuente de datos para el que se define el *tipo-datos-fuente-datos*.

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

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

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

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

WRAPPER *nombre-derivador*
Especifica el nombre del derivador que el servidor federado utiliza para interactuar con las fuentes de datos del tipo y versión indicados por *tipo-servidor* y *versión-servidor*.

TYPE *tipo-datos-fuente-datos*
Especifica el tipo de datos de fuente de datos que se está correlacionando con el tipo de datos local.

Se pueden utilizar paréntesis vacíos para los tipos de datos con parámetros. Si se especifican paréntesis vacíos en una correlación de tipos en avance como, por ejemplo, CHAR(), la correlación de tipos se aplica al tipo de datos con cualquier longitud. Si se especifican paréntesis vacíos en una correlación de tipo de datos invertida, la longitud se determina a partir de la longitud de columna especificada en el DDL transparente. Si se omiten los paréntesis, se utiliza la longitud por omisión para el tipo de datos.

El *tipo-datos-fuente-datos* debe ser un tipo de datos incorporado. No están permitidos los tipos definidos por el usuario.

Si se especifica el *nombre-servidor* con una correlación de tipos o los servidores existentes se ven afectados por la correlación de tipos, *tipo-datos-fuente-datos*, *p* y *s* se verifican al crear la correlación de tipos (SQLSTATE 42611).

p Si se especifica *p*, el tipo de correlación sólo afecta al tipo de datos cuya longitud o precisión sea igual a *p*.

[p1..p2]
Sólo para la correlación de tipos en avance. Para un tipo de datos decimal, *p1* y *p2* especifican el número mínimo y máximo de dígitos que un valor puede tener. Para tipos de datos de serie, *p1* y *p2* especifican el número mínimo y máximo de caracteres que un valor puede tener. En todos los casos, el máximo debe ser igual o exceder del mínimo; y ambos números deben ser válidos con respecto al tipo de datos.

s Si se especifica *s*, sólo el tipo de datos cuya escala sea igual a *s* se ve afectado por la correlación de tipos.

[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 bit. Estas palabras clave son necesarias si la columna del tipo de la fuente de datos contiene valores binarios. El gestor de bases de datos determinará este atributo si no se especifica para un tipo de datos de caracteres.

Notas

- Una sentencia CREATE TYPE MAPPING de una unidad de trabajo determinada (UOW) no se puede procesar (SQLSTATE 55007) bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- Cuando se pueden aplicar varias correlaciones de tipos, se utilizará la más reciente. Puede recuperar la hora de creación para una correlación de tipos consultando la columna CREATE_TIME de la vista del catálogo SYSCAT.TYPEMAPPINGS.

Ejemplos

Ejemplo 1: Cree una correlación de tipos en avance entre el tipo de datos DATE de Oracle y el tipo de datos SYSIBM.DATE. Para todos los apodos que se creen después de haber definido esta correlación, las columnas del tipo de datos DATE de Oracle se correlacionarán con columnas DB2 del tipo de datos DATE.

```
CREATE TYPE MAPPING MY_ORACLE_DATE
  FROM LOCAL TYPE SYSIBM.DATE
  TO SERVER TYPE ORACLE
  REMOTE TYPE DATE
```

Ejemplo 2: Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.DECIMAL(10,2) y el tipo de datos NUMBER([10..38],2) en la fuente de datos ORACLE1. Si hay una columna en la tabla Oracle del tipo de datos

NUMBER(11,2), se correlacionará con una columna de datos de tipo DECIMAL(10,2), porque 11 está entre 10 y 38.

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  FROM LOCAL TYPE SYSIBM.DECIMAL(10,2)
  TO SERVER ORACLE1
  REMOTE TYPE NUMBER([10..38],2)
```

Ejemplo 3: Cree una correlación de tipos en avance entre el tipo de datos SYSIBM.VARCHAR(*p*) y el tipo de datos CHAR(*p*) de Oracle en la fuente de datos ORACLE1 (*p* es cualquier longitud). Si hay una columna en la tabla Oracle del tipo de datos CHAR(10), se correlacionará con una columna del tipo de datos VARCHAR(10).

```
CREATE TYPE MAPPING MY_ORACLE_CHAR
  FROM LOCAL TYPE SYSIBM.VARCHAR()
  TO SERVER ORACLE1
  REMOTE TYPE CHAR()
```

Ejemplo 4: Cree una correlación de tipos invertida del tipo de datos NUMBER(10,2) de Oracle en la fuente de datos ORACLE2 y el tipo de datos SYSIBM.DECIMAL(10,2). Si utiliza un DDL transparente para crear una tabla Oracle y especifica una columna de tipo de datos DECIMAL(10,2), DB2 creará la tabla Oracle con una columna de tipo de datos NUMBER(10,2).

```
CREATE TYPE MAPPING MY_ORACLE_DEC
  TO LOCAL TYPE SYSIBM.DECIMAL(10,2)
  FROM SERVER ORACLE2
  REMOTE TYPE NUMBER(10,2)
```

CREATE USER MAPPING

La sentencia CREATE USER MAPPING define una correlación entre un ID de autorización que utiliza una base de datos federada y el ID de autorización y la contraseña que se deben utilizar en una fuente de datos especificada.

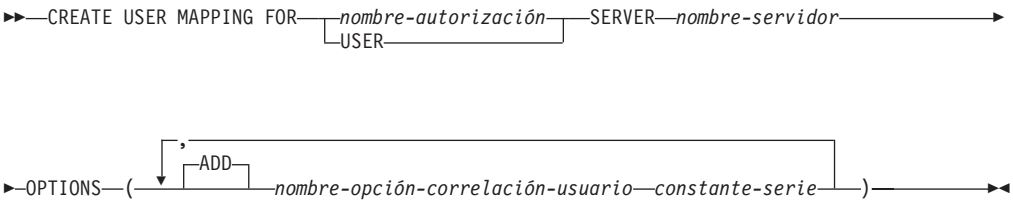
Invocación

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

Autorización

Si el ID de autorización de la sentencia es diferente del nombre de autorización que se correlaciona con la fuente de datos, los privilegios del ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM. De lo contrario, si el ID de autorización y el nombre de la autorización coinciden, no son obligatorios privilegios o 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 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')
```

CREATE VARIABLE

La sentencia CREATE VARIABLE define una variable global.

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

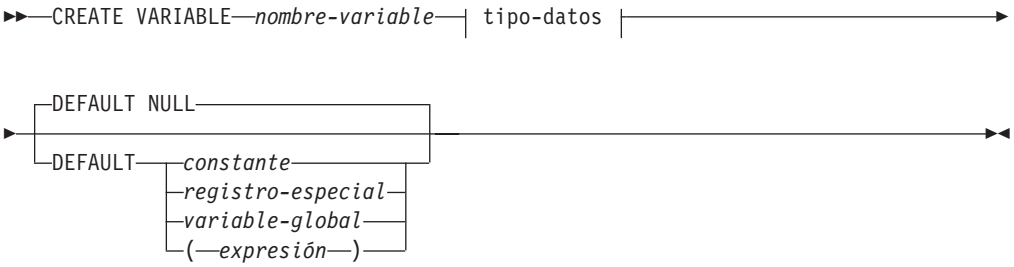
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Autorización IMPLICIT_SCHEMA en la base de datos, si el nombre de esquema implícito o explícito de la variable no existe
- Privilegio CREATEIN para el esquema, si el nombre de esquema de la variable hace referencia a un esquema existente
- Autorización SYSADM o DBADM

y cualesquiera privilegios que sean necesarios para ejecutar la expresión por omisión.

Los privilegios de grupo no se tienen en cuenta a la hora de comprobar la autorización de los objetos a los que se hace referencia en la cláusula de expresión.

Sintaxis



tipo-datos:

Descripción

nombre-variable
Da nombre a la variable global. El nombre, incluyendo un calificador implícito o explícito, no debe identificar una variable global que ya exista en el servidor actual (SQLSTATE 42710). Si no se especifica el calificador, se asigna el esquema actual de forma implícita.

tipo-datos
Especifica el tipo de datos de la variable global. El tipo de datos puede ser un tipo de datos incorporado, un tipo diferenciado o un tipo de referencia.

tipo-datos-incorporado
Especifica un tipo de datos incorporado. Consulte en "CREATE TABLE" la descripción de los tipos de datos incorporados. Tenga en cuenta que CLOB, DBCLOB, BLOB, LONG VARCHAR, LONG VARGRAPHIC, XML, ARRAY, o los tipos estructurados no pueden especificarse para las variables globales (SQLSTATE 42611).

Se puede especificar FOR BIT DATA como parte de los tipos de datos de serie de caracteres.

nombre-tipo-diferenciado
Especifica un tipo diferenciado. La longitud, precisión y la escala de la variable global son, respectivamente, la longitud, la precisión y la escala del tipo de fuente del tipo diferenciado. Si se especifica un *nombre-tipo-diferenciado* sin un nombre de esquema, el *nombre-tipo-diferenciado* se resuelve buscando los esquemas en la vía de acceso de SQL. Las mismas limitaciones que se aplican a los tipos incorporados se aplican al tipo de fuente del tipo diferenciado.

nombre-tipo-referencia
Especifica un tipo de referencia. La longitud, precisión y la escala de la variable global son, respectivamente, la longitud, la precisión y la escala del tipo de fuente del tipo de referencia. Si se especifica un *nombre-tipo-referencia* sin un nombre de esquema, el *nombre-tipo-referencia* se resuelve buscando los esquemas en la vía de acceso de SQL. Las mismas limitaciones que se aplican a los tipos incorporados se aplican al tipo de fuente del tipo de referencia.

DEFAULT
Especifica un valor por omisión para la variable global. El valor puede ser una constante, un registro especial, una variable global, una expresión o la palabra clave NULL. La expresión puede ser cualquier expresión del tipo que se describe en el apartado "Expresiones". Si no se especifica un valor por omisión, el valor de la variable se inicializa en el valor nulo. El tamaño máximo de la expresión es de 64K.

La expresión por omisión no debe modificar datos SQL (SQLSTATE 428FL) ni realizar acciones externas (SQLSTATE 42845). La expresión debe ser compatible con la asignación al tipo de datos de la variable.

Notas

- Las variables globales tienen un ámbito de sesión. Esto significa que, aunque están disponibles para todas las sesiones activas en la base de datos, su valor es privado para cada una de las sesiones.
- Las modificaciones en el valor de una variable global no están bajo el control de la transacción. El valor de la variable global se preserva cuando una transacción finaliza con una sentencia COMMIT o ROLLBACK.
- ***Privilegios para utilizar una variable global:*** Todo intento de leer o escribir en una variable global creada mediante esta sentencia requiere que el ID de autorización que intente esta acción detente el privilegio correspondiente sobre la variable global. Al definidor de la variable se le otorgan de modo implícito todos los privilegios sobre la variable.
- ***Establecimiento del valor por omisión:*** La primera vez que se hace referencia a una variable global creada dentro de un determinado ámbito se crea una instancia de la misma en su valor por omisión. Tenga en cuenta que si se hace referencia a una variable global en una sentencia, se creará una instancia de la misma con independencia del flujo de control para dicha sentencia.

- ***Utilización de una variable global de sesión recién creada:*** Si se crea una variable global en una sesión, no podrán utilizarla otras sesiones hasta que se haya confirmado la unidad de trabajo. Sin embargo, la variable global nueva puede utilizarse en la sesión que creó la variable antes de la confirmación de la unidad de trabajo.

Ejemplos

Ejemplo 1: Cree una variable global para indicar la impresora que ha de utilizarse para la sesión.

```
CREATE VARIABLE MYSCHEMA.MYJOB_PRINTER VARCHAR(30)
  DEFAULT 'Impresora por omisión'
```

Ejemplo 2: Cree una variable global para indicar el departamento en el que trabaja un empleado.

```
CREATE VARIABLE SCHEMA1.GV_DEPTNO INTEGER
  DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
    WHERE EMPUSER = SESSION_USER))
```

Ejemplo 3: Cree una variable global para indicar el nivel de seguridad del usuario actual.

```
CREATE VARIABLE SCHEMA2.GV_SECURITY_LEVEL INTEGER
  DEFAULT (GET_SECURITY_LEVEL (SESSION_USER))
```

CREATE VIEW

La sentencia CREATE VIEW define una vista para una o más tablas, vistas o apodos.

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

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

- 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 igual que el del propietario de la tabla raíz de la jerarquía de tablas
  - Tener el privilegio SELECT WITH GRANT en la tabla subyacente de la subvista o la supervista no debe tener otorgado el privilegio SELECT un usuario distinto del definidor de la vista

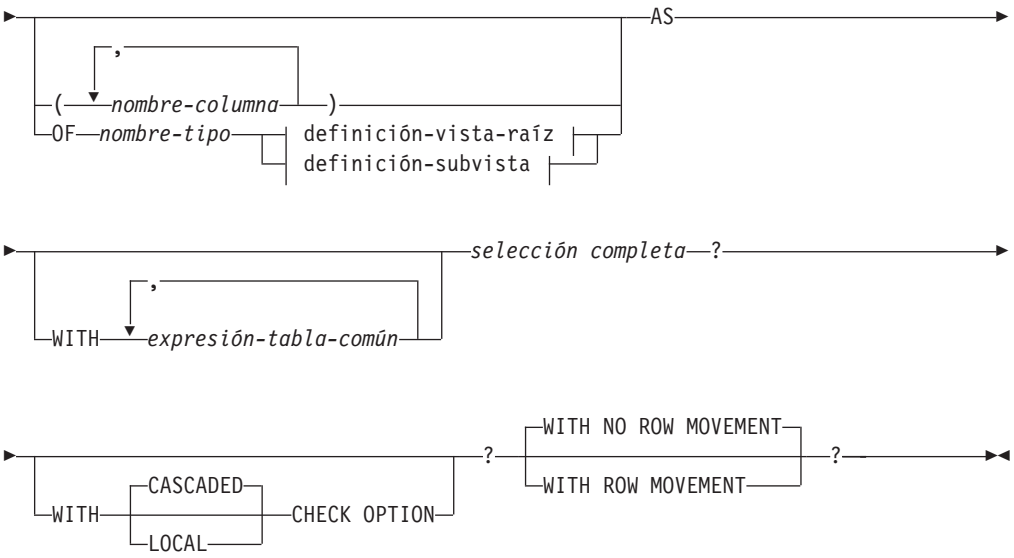
  Si se especifica WITH ROW MOVEMENT:
  - Privilegio UPDATE en la tabla o vista
- Autorización SYSADM o DBADM

No se tienen en cuenta los privilegios de grupo para cualquier tabla o vista especificada en la sentencia CREATE VIEW.

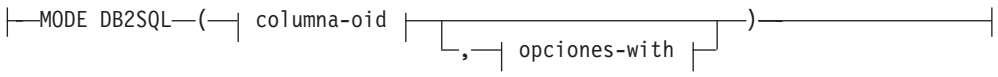
No se tienen en cuenta los privilegios al definir una vista para un apodo de base de datos federada. Los requisitos de autorización de la fuente de datos para la tabla o vista referenciada por el apodo se aplican cuando se procesa la consulta. El ID de autorización de la sentencia puede estar correlacionado con un ID de autorización remoto distinto.

Si el definidor de una vista sólo puede crear la vista porque el definidor tiene la autorización SYSADM, se le otorgará la autorización DBADM explícita con el propósito de crear la vista.

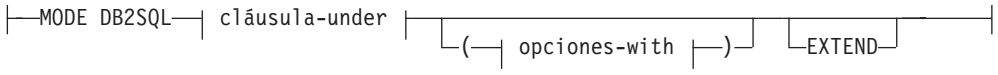
Sintaxis



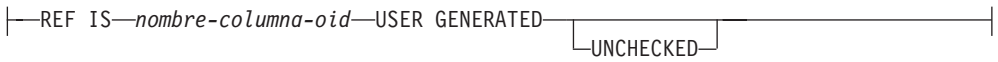
definición-vista-raíz:



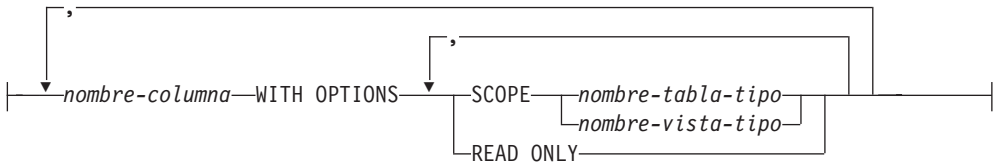
definición-subvista:



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 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 "Notas" en la página 658 para obtener más información sobre el ámbito y las columnas de tipo de referencia.

nombre-tabla-tipo
El nombre de una tabla con tipo. La tabla debe existir ya o ser la misma que el nombre de la tabla que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-tabla-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de ninguno de los valores existentes en *nombre-columna* para garantizar si los valores hacen referencia realmente a las filas existentes en el *nombre-tabla-tipo*.

nombre-vista-tipo
El nombre de una vista con tipo. La vista debe existir ya o ser la misma que el nombre de la vista que se está creando (SQLSTATE 42704). El tipo de datos de *nombre-columna* debe ser REF(*S*), donde *S* es el tipo de *nombre-vista-tipo* (SQLSTATE 428DM). No se realiza ninguna comprobación de los valores existentes de *nombre-columna* para asegurarse de si realmente los valores hacen referencia a filas existentes de *nombre-vista-tipo*.

READ ONLY
Identifica la columna como columna de sólo lectura. Esta opción se utiliza para hacer que una columna sea de sólo lectura, de manera que las definiciones de las subvistas puedan especificar una expresión para la misma columna que sea de sólo lectura implícitamente.

AS
Identifica la definición de la vista.

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 parámetro ni a tablas temporales declaradas. En cambio, una vista parametrizada se puede crear como función de tabla SQL.

La selección completa no puede incluir una sentencia de cambio de datos de SQL en la cláusula FROM (SQLSTATE 428FL).

Para subvistas y vistas con tipo: La *selección completa* debe ajustarse a las normas siguientes, de lo contrario se devolverá un error (SQLSTATE 428EA a no ser que se especifique lo contrario).

- La selección completa no debe incluir referencias a las funciones DBPARTITIONNUM o HASHEDVALUE, a las funciones no deterministas o a funciones que se han definido para que exista una acción externa.
- El cuerpo de la vista debe consistir en una sola subselección o en UNION ALL de dos o más subselecciones. Supongamos que cada una de las subselecciones que participan directamente en el cuerpo de la vista se llama una *rama* de la vista. Una vista puede tener una o más ramas.

- La cláusula FROM de cada rama debe consistir de una sola tabla o vista (no necesariamente de tipo), llamada la tabla o vista *subyacente* de esa rama.
- La tabla o vista subyacente de cada rama debe estar en una jerarquía separada (es decir, una vista no puede tener varias ramas con sus tablas o vistas subyacentes en la misma jerarquía).
- Ninguna de las ramas de una definición de vista con tipo puede que especifique GROUP BY o HAVING.
- Si el cuerpo de la vista contiene UNION ALL, la vista de la raíz en la jerarquía debe especificar la opción UNCHECKED para su columna de OID.

Para una jerarquía de vistas y subvistas: BR1 y BR2 son las ramas que aparecen en las definiciones de vistas en la jerarquía. T1 debe ser la tabla subyacente o vista de BR1 y T2 debe ser la tabla o vista subyacente de BR2. Entonces:

- Si T1 y T2 no están en la misma jerarquía, entonces la vista raíz de la jerarquía de vistas debe especificar la opción UNCHECKED para su columna OID.
- Si T1 y T2 están en la misma jerarquía, entonces BR1 y BR2 deben contener predicados o cláusulas ONLY que sean suficientes para garantizar que sus conjuntos de filas no están unidos.

Para subvistas con tipo definidas utilizando EXTEND AS: Para cada rama en el cuerpo de la subvista:

- La tabla subyacente de cada rama debe ser una subtabla (no necesariamente la correspondiente) de alguna tabla subyacente de la supervista inmediata.
- Las expresiones de la lista SELECT deben poder asignarse a las columnas no heredadas de la subvista (SQLSTATE 42854).

Para subvistas con tipo definidas utilizando AS sin EXTEND:

- Para cada rama del cuerpo de la subvista, las expresiones en la lista SELECT deben poder asignarse a los tipos declarados de las columnas heredadas y no heredadas de la subvista (SQLSTATE 42854).
- La expresión OID de cada rama de una jerarquía de la subvista debe ser equivalente (excepto para la conversión del tipo de datos) a la expresión OID en la rama en la misma jerarquía en la vista raíz.
- La expresión para una columna no definida (implícita o explícitamente) como READ ONLY en una supervista debe ser equivalente a todas las ramas en la misma jerarquía subyacente en sus subvistas.

WITH CHECK OPTION

Especifica la restricción según la cual cada fila que se haya insertado o actualizado a través de la vista debe ajustarse a la definición de dicha vista. Una fila que no se ajusta a la definición de la vista es una fila que no cumple las condiciones de búsqueda de la vista.

WITH CHECK OPTION no debe especificarse si se da alguna de las condiciones siguientes:

- La vista es de sólo lectura (SQLSTATE 42813). Si se ha especificado WITH CHECK OPTION para una vista que puede actualizarse y que no admite inserciones, la restricción se aplicará sólo a las actualizaciones.
- La vista hace referencia a la función DBPARTITIONNUM o HASHEDVALUE, a una función no determinante o a una función con acción externa (SQLSTATE 42997).
- Un apodo es el destino de actualización de la vista.
- Una vista que tiene definido un activador INSTEAD OF es el destino de actualización de la vista (SQLSTATE 428FQ).

Si se omite WITH CHECK OPTION, la definición de la vista no se utilizará en la comprobación de ninguna operación de inserción o de actualización que utilicen esa vista. Si la vista depende directa o indirectamente de otra vista que incluya WITH CHECK OPTION, durante las operaciones de inserción y actualización es posible que se siga produciendo algún tipo de comprobación. Dado que no se utiliza la definición de la vista, se podrían insertar o actualizar filas a través de la vista que no se ajustasen a la definición de la vista.

CASCADED
La restricción WITH CASCADED CHECK OPTION en una vista *V* significa que *V* hereda las condiciones de búsqueda como restricciones de cualquier vista actualizable de la que *V* depende. Además, todas las vistas actualizables que dependen de *V* también están sometidas a estas restricciones. De esta forma, las condiciones de búsqueda de *V* y todas las vistas de las que *V* depende se unen mediante AND para formar una restricción que se aplica a las inserciones o actualizaciones de *V* o de cualquier vista dependiente de *V*.

LOCAL
La restricción WITH LOCAL CHECK OPTION en una vista *V* significa que la condición de búsqueda de *V* se aplica como restricción a las inserciones o actualizaciones de *V* o de cualquier vista que sea dependiente de *V*.

La diferencia entre CASCADED y LOCAL se explica en el ejemplo siguiente. Tome en consideración las siguientes vistas actualizables (sustituyendo Y en las cabeceras de columna de la tabla que sigue):

```
V1 definida en la tabla T
V2 definida en V1 WITH Y CHECK OPTION
V3 definida en V2
V4 definida en V3 WITH Y CHECK OPTION
V5 definida en V4
```

La tabla siguiente muestra las condiciones de búsqueda con las que se comparan las filas insertadas o actualizadas:

	Y es LOCAL	**Y es CASCADED**
V1 se comprueba con:	ninguna vista	ninguna vista
V2 se comprueba con:	V2	V2, V1
V3 se comprueba con:	V2	V2, V1
V4 se comprueba con:	V2, V4	V4, V3, V2, V1
V5 se comprueba con:	V2, V4	V4, V3, V2, V1

Tome en consideración la siguiente vista actualizable que muestra el efecto de WITH CHECK OPTION utilizando la opción CASCADED por omisión:

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

La siguiente sentencia INSERT que utiliza *V1* será satisfactoria porque *V1* no tiene WITH CHECK OPTION y *V1* no depende de ninguna otra vista que lo tenga.

```
INSERT INTO V1 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V2* dará lugar a error porque *V2* tiene WITH CHECK OPTION y la inserción generaría una fila que no se ajustaría con la definición de *V2*.

```
INSERT INTO V2 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* dará lugar a un error aun no teniendo WITH CHECK OPTION porque *V3* depende de *V2*, que sí tiene especificado WITH CHECK OPTION (SQLSTATE 44000).

```
INSERT INTO V3 VALUES(5)
```

La siguiente sentencia INSERT que utiliza *V3* será satisfactoria aunque no se ajuste a la definición de *V3* (*V3* no tiene WITH CHECK OPTION); se ajusta a la definición de *V2*, que sí tiene WITH CHECK OPTION.

```
INSERT INTO V3 VALUES(200)
```

WITH NO ROW MOVEMENT o WITH ROW MOVEMENT
Especifica la acción a emprender para una vista UNION ALL actualizable cuando se actualiza una fila de modo que viola una restricción de comprobación de la tabla subyacente. El valor por omisión es WITH NO ROW MOVEMENT.

WITH NO ROW MOVEMENT
Especifica que se devolverá un error (SQLSTATE 23513) si se actualiza una fila mediante un método que infrinja restricción de comprobación de la tabla subyacente.

WITH ROW MOVEMENT
Especifica que se debe mover una fila actualizada a la tabla subyacente adecuada, aunque viole una restricción de comprobación en dicha tabla.

El movimiento de filas implica la supresión de las filas que violan la restricción de comprobación y la inserción de dichas filas de nuevo en la vista. La cláusula WITH ROW MOVEMENT sólo se puede especificar para vistas UNION ALL cuyas columnas sean actualizables (SQLSTATE 429BJ). Si se inserta una fila (quizás después de la activación de un activador) en la misma tabla subyacente desde la que se ha suprimido, se devuelve un error (SQLSTATE 23524). Una vista definida utilizando la cláusula WITH ROW MOVEMENT no debe contener operaciones UNION ALL anidadas, excepto en la selección completa más externa (SQLSTATE 429BJ).

Notas

- La creación de una vista con un nombre de esquema que todavía no existe dará como resultado la creación implícita de ese esquema siempre que el ID de autorización de la sentencia disponga de autorización IMPLICIT_SCHEMA. El propietario del esquema es SYSIBM. El privilegio CREATEIN 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
- ***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 que se pueden insertar:***
  - 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 se puede insertar en una vista (incluida una UNION ALL) solamente si cumple las restricciones de comprobación de exactamente una de las tablas base subyacentes.
  - Para insertarla en una vista que incluya columnas que no se pueden actualizar, estas columnas deben omitirse de la lista de columnas.
- ***Vistas de sólo lectura:*** una vista es de *sólo lectura* si *no* 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 tabla comunes y las expresiones de tablas anidadas deben seguir el mismo conjunto de normas para determinar si se pueden suprimir, actualizar, insertar o si son de sólo lectura.
- ***Vistas no operativas:*** una *vista no operativa* es una vista que ya no está disponible para sentencias de SQL. Una vista deja de ser operativa si:
  - Se revoca un privilegio del que depende la definición de dicha vista.
  - Se 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.

  Al definidor de la vista de le otorgan los privilegios INSERT, UPDATE, UPDATE de nivel de columna o DELETE en la vista si ésta no es de sólo lectura y el definidor dispone de los privilegios correspondientes en los objetos subyacentes.

  Para una vista definida WITH ROW MOVEMENT, el definidor adquiere el privilegio UPDATE sobre la vista sólo si el definidor tiene el privilegio UPDATE sobre todas las columnas de la vista, así como los privilegios INSERT y DELETE sobre todas las tablas o vistas subyacentes.

  El definidor de una vista sólo adquiere privilegios si los privilegios de los cuales aquéllos derivan ya existen cuando se crea la vista. El definidor debe tener estos privilegios directamente o porque PUBLIC tiene este privilegio. Los privilegios no se tienen en cuenta al definir una vista en un apodo de servidor federado. Sin embargo, al utilizar una vista en un apodo, el ID de autorización de usuario debe tener los privilegios de selección válidos en la tabla o vista a la que el apodo hace referencia en la fuente de datos. De lo contrario, se emite un error. No se tienen en cuenta los privilegios pertenecientes a grupos de los que forma parte el definidor.

  Cuando se crea una subvista, los privilegios de SELECT que tiene la supervista inmediata se otorgan automáticamente a la subvista.

- ***Columnas REF y de ámbito:***

  Cuando 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:*** Una columna de una vista se considera una columna de identidad, si el elemento de la columna correspondiente en la selección completa de la definición de la vista es el nombre de la columna de identidad de una tabla o el nombre de una columna de una vista que directa o indirectamente se correlaciona con el nombre de una columna de identidad de una tabla base.

  En todos los demás casos, las columnas de una vista no obtendrán el atributo de identidad. Por ejemplo:

  - La lista de selección de la definición de vista contiene varias instancias del nombre de una columna de identidad (es decir, se selecciona la misma columna más de una vez)
  - la definición de la vista incluye una operación de unión
  - una columna de la definición de vista incluye una expresión que hace referencia a una columna de identidad
  - la definición de la vista incluye una operación UNION

  Cuando se inserta en una vista para que la lista de selecciones de la definición de vista incluye directa o indirectamente el nombre de una columna de identidad de una tabla base, se aplican las mismas normas que si la declaración INSERT hicieran referencia directamente a la columna de identidad de la tabla base.

- ***Vistas federadas:*** Una vista federada es una vista que incluye una referencia a un apodo situado en algún lugar de la selección completa. La presencia de este apodo cambia el modelo de autorización utilizado para la vista cuando posteriormente se hace referencia a la misma en una consulta.

  Cuando se crea la vista, no se realiza ninguna comprobación de privilegios para determinar si el definidor de la vista tiene acceso a la tabla de fuente de datos subyacente o a la vista de un apodo. La comprobación de privilegios de referencias a las tablas o las vistas de la base de datos federada se lleva a cabo del modo habitual, y se requiere que el definidor de la vista tenga al menos el privilegio SELECT en dichos objetos.

  Cuando posteriormente se hace referencia en una consulta, a una vista federada, los apodos que resultan de las consultas en la fuente de datos y el ID de autorización que ha emitido la consulta (o el ID de autorización remota con el que se correlaciona) deben tener los privilegios necesarios para acceder a la tabla o vista de la fuente de datos. El ID de autorización que emite la consulta que hace referencia a la vista federada no es necesario que tenga privilegios adicionales en tablas o vistas (sin federar) que existen en el servidor federado.

- ***ROW MOVEMENT, activadores y restricciones:*** Cuando se actualiza una vista definida mediante la cláusula WITH ROW MOVEMENT, la secuencia de operaciones de activador y restricciones es la siguiente:

  1. Los activadores BEFORE UPDATE se activan para todas las filas que se están actualizando, incluidas las filas que finalmente se moverán.
  2. Se procesa la operación de actualización.

3. Se procesan las restricciones para todas las filas actualizadas.
4. 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. Puesto que de trata de una sentencia UPDATE, todos los activadores del nivel de sentencia UPDATE se activan para todas las tablas subyacentes.
5. Los activadores BEFORE DELETE se activan para todas las filas que no han cumplidos las restricciones tras la operación de actualización (estas filas son las que debe moverse).
6. Se procesa la operación de supresión.
7. Se procesan las restricciones para todas las filas suprimidas.
8. Los activadores AFTER DELETE (tanto de nivel de fila como de nivel de sentencia) se activan en orden de creación, para todas las filas suprimidas. Los activadores del nivel de sentencia se activan únicamente para las tablas implicadas en la operación de supresión.
9. Los activadores BEFORE INSERT se activan para todas las filas que se van a insertar (es decir, las filas que se van a mover). Las nuevas tablas de transición para los activadores BEFORE INSERT contienen los datos de entrada que ha proporcionado el usuario.
10. Se procesa la operación de inserción.
11. Se procesan las restricciones para todas las filas insertadas.
12. Los activadores AFTER INSERT (tanto de nivel de fila como de nivel de sentencia) se activan en el orden de creación para todas las filas insertadas. Los activadores del nivel de sentencia se activan únicamente para las tablas implicadas en la operación de inserción.

- ***Vistas UNION ALL anidadas:*** Una vista definida con UNION ALL y basada, directa o indirectamente, en una vista que también está definida con UNION ALL no se puede actualizar si alguna de las vistas se ha definido utilizando la cláusula WITH ROW MOVEMENT (SQLSTATE 429BK).
- ***Consideraciones para columnas implícitamente ocultas:*** es posible que la tabla resultante de la selección completa incluya una columna de la tabla base que está definida como implícitamente oculta. Esto puede suceder cuando se hace referencia explícitamente a la columna implícitamente oculta en la selección completa de la definición de vista. Sin embargo, la columna correspondiente de la vista no hereda el atributo de implícitamente oculta. Las columnas de una vista no pueden definirse como ocultas.
- ***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.

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:

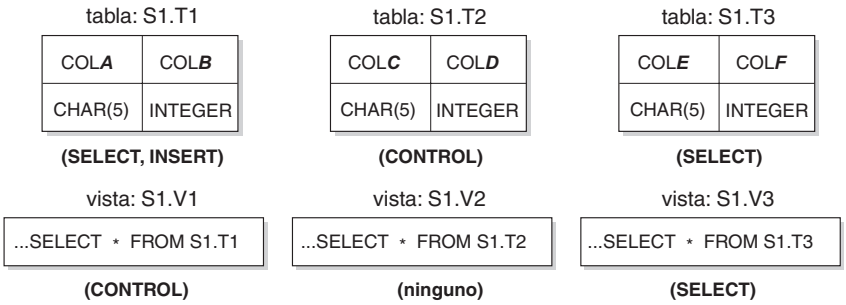


Figura 1. Tablas y vistas para el ejemplo 6

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.

CREATE WORK ACTION SET

La sentencia CREATE WORK ACTION SET define un conjunto de acciones de trabajo y acciones de trabajo en el conjunto de acciones de trabajo.

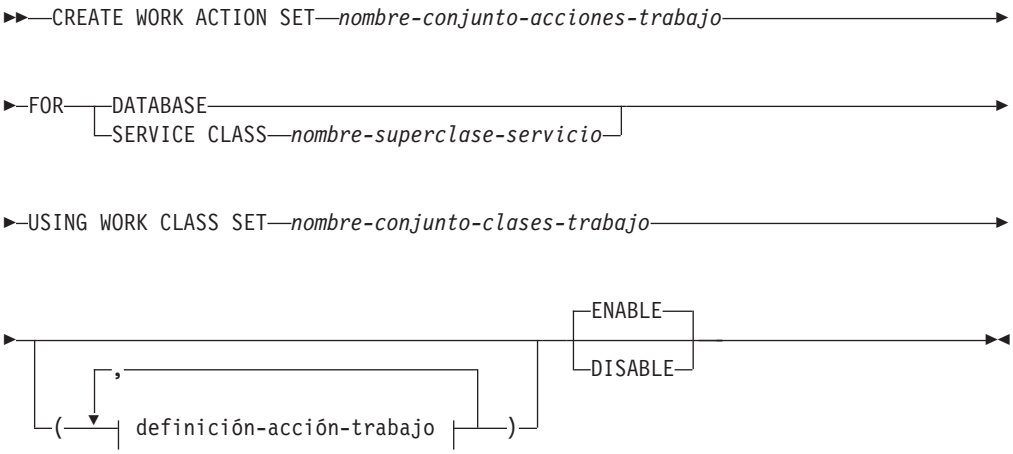
Invocación

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

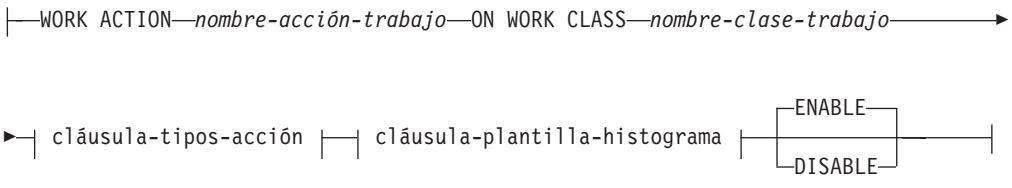
Autorización

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

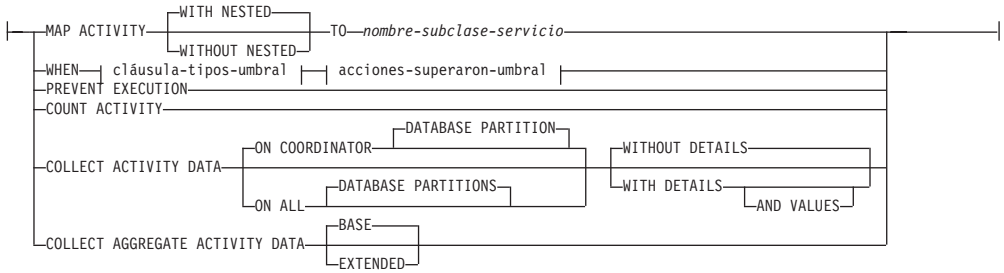
Sintaxis



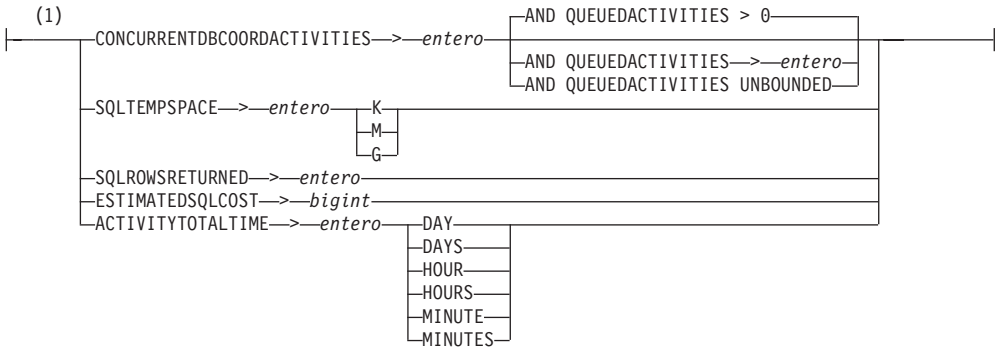
definición-acción-trabajo:



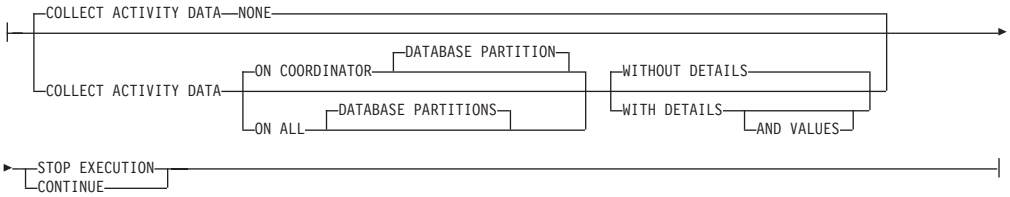
cláusula-tipos-acción:



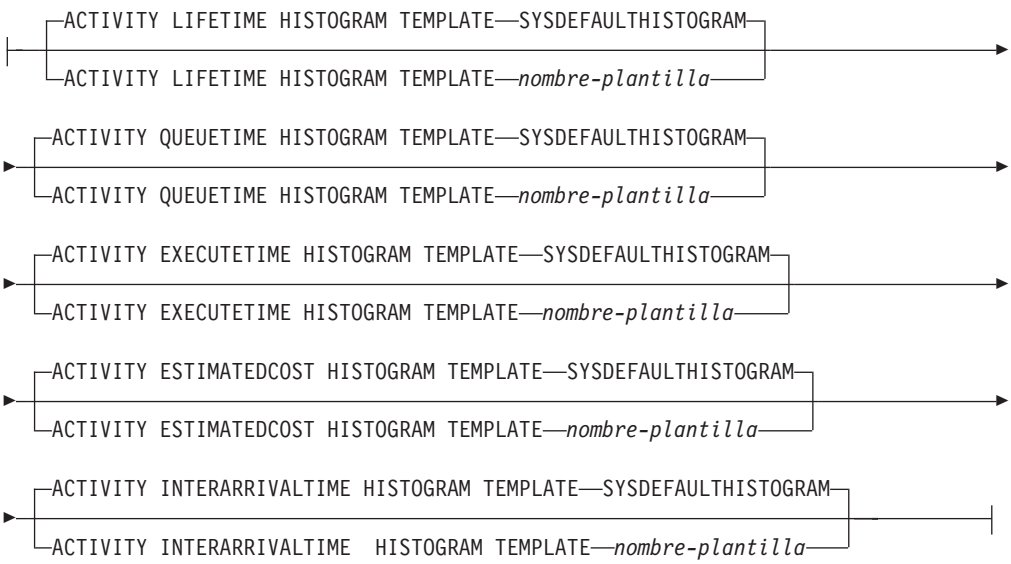
cláusula-tipos-umbral:



acciones-superaron-umbral:



cláusula-plantilla-histograma:



Notas:

1 Sólo puede aplicarse una acción de trabajo del mismo tipo de umbral a una única clase de trabajo a la vez.

Descripción

nombre-conjunto-acciones-trabajo
Da nombre al conjunto de acciones de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-acciones-trabajo* no debe identificar un conjunto de acciones de

trabajo que exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

FOR
Especifica el objeto de gestor de base de datos al que se aplicarán las acciones de este conjunto de acciones de trabajo. Cada uno de los objetos de gestor de base de datos sólo puede tener definido un conjunto de acciones de trabajo definida para el mismo (SQLSTATE 5U017).

DATABASE
Las acciones de este conjunto de acciones de trabajo van a aplicarse a la base de datos. Si se especifica DATABASE, la cláusula MAP ACTIVITY no puede especificarse (SQLSTATE 5U034).

SERVICE CLASS *nombre-superclase-servicio*
Las acciones de este conjunto de acciones de trabajo van a aplicarse a *nombre-superclase-servicio*. Si se especifica SERVICE CLASS, no pueden especificarse acciones de umbral (SQLSTATE 5U034). *nombre-superclase-servicio* debe existir en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no debe ser una subclase de servicio y no puede ser ninguna de las clases siguientes (SQLSTATE 5U032):

- La clase de servicio de sistema (SYSDEFAULTSYSTEMCLASS)
- La clase de servicio de mantenimiento (SYSDEFAULTMAINTENANCECLASS)
- la clase de servicio de usuario por omisión (SYSDEFAULTUSERCLASS)

USING WORK CLASS SET *nombre-conjunto-clases-trabajo*
Especifica el conjunto de clases de trabajo que contiene las clases de trabajo que clasificarán actividades de base de datos en las que realizar acciones. *nombre-conjunto-clases-trabajo* debe existir en el servidor actual (SQLSTATE 42704).

definición-acción-trabajo
Especifica la definición de la acción de trabajo.

WORK ACTION *nombre-acción-trabajo*
Da nombre a la acción de trabajo. El *nombre-acción-trabajo* no debe identificar una acción de trabajo que ya exista en el servidor actual en este conjunto de acciones de trabajo (SQLSTATE 42710). El *nombre-acción-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

ON WORK CLASS *nombre-clase-trabajo*
Especifica la clase de trabajo que identifica las actividades de base de trabajo a las que se aplicará esta acción de trabajo. El *nombre-clase-trabajo* debe existir en el *nombre-conjunto-clases-trabajo* en el servidor actual (SQLSTATE 42704).

MAP ACTIVITY
Especifica una acción de trabajo de correlación de la actividad. Esta acción sólo puede especificarse si el objeto para el que se ha definido este conjunto de acciones de trabajo está definido como una superclase de servicio (SQLSTATE 5U034).

WITH NESTED o WITHOUT NESTED
Especifica si las actividades anidadas bajo esta actividad se van a correlacionar o no a la subclase de servicio. El valor por omisión es WITH NESTED.

WITH NESTED
Todas las actividades de base de datos que tengan un nivel de

anidamiento de cero que estén clasificadas bajo la clase de trabajo y todas las actividades de base de datos anidadas bajo esta actividad, se correlacionarán a la subclase de servicio; es decir, las actividades con un nivel de anidamiento superior a cero se ejecutarán en la misma clase de servicio que las actividades con un nivel de anidamiento de cero.

WITHOUT NESTED
Sólo las actividades que tengan un nivel de anidamiento de cero que estén clasificadas bajo la clase de trabajo se correlacionarán con la subclase de servicio. Las actividades anidadas bajo esta actividad se manejarán con arreglo a su tipo de actividad.

TO *nombre-subclase-servicio*
Especifica la subclase de servicio a la que están correlacionadas las actividades. El *nombre-subclase-servicio* debe existir en el *nombre-superclase-servicio* en el servidor actual (SQLSTATE 42704). El *nombre-subclase-servicio* no puede ser la subclase de servicio por omisión, SYSDEFAULTSUBCLASS (SQLSTATE 5U018).

WHEN
Especifica el umbral que se aplicará a la actividad de base de datos asociada a la clase de trabajo para la que se define esta acción de trabajo. Un umbral sólo puede especificarse si el objeto de gestor de base de datos para el que se ha definido esta acción de trabajo está definido como base de datos (SQLSTATE 5U034). Ninguno de estos umbrales se aplican a las actividades de base de datos internas iniciadas por medio del gestor de base de datos o a las actividades de base de datos generadas por medio de rutinas SQL administrativas.

cláusula-tipos-umbral
Para obtener una descripción de los tipos de umbral válidos, consulte la sentencia “CREATE THRESHOLD”.

acciones-superaron-umbral
Para obtener una descripción de las acciones de excedido umbral válidas, consulte la sentencia “CREATE THRESHOLD”.

PREVENT EXECUTION
Especifica que no se permitirá la ejecución de ninguna de las actividades de base de datos asociada a la clase de trabajo para la que se ha definido esta acción de trabajo (SQLSTATE 5U033).

COUNT ACTIVITY
Especifica que se ejecuten todas las actividades de base de datos asociadas a la clase de trabajo para las que se define esta acción de trabajo y que cada vez que se ejecute una de ellas, aumente el contador para la clase de trabajo.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo van a enviarse al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. El valor por omisión es COLLECT ACTIVITY DATA WITHOUT DETAILS.

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en las que se procesa la actividad. Para umbrales de predicción, la información sobre actividades sólo se recopila en todas las particiones si también se especifica la acción CONTINUE para umbrales excedidos. Para umbrales de reacción, la cláusula ON ALL DATABASE PARTITIONS no tiene efecto y la información sobre actividades siempre se recopila únicamente en la partición coordinadora. Para umbrales de predicción y de reacción, los detalles o valores de actividades únicamente se recopilarán en la partición coordinadora.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo van a enviarse al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. La información del entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia va a enviarse al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

COLLECT AGGREGATE ACTIVITY DATA
Especifica que los datos de actividad agregados se van a capturar para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y se van a enviar al supervisor de sucesos de estadísticas, si hay alguno activo. Esta información se recopila periódicamente en un intervalo que se especifica por medio del parámetro de configuración de base de datos **wlm_collect_int**. El valor por omisión es COLLECT AGGREGATE ACTIVITY DATA BASE. Esta cláusula no puede especificarse para una acción de trabajo definida en un conjunto de acciones de trabajo aplicado a una base de datos.

BASE
Especifica que los datos de actividades agregados básicos deben capturarse para las actividades asociadas con la clase de trabajo para la que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Los datos de actividad agregada básica incluyen:

- Marca de límite superior del coste de actividad estimado
- Marca de límite superior de las filas devueltas
- Marca de límite superior del uso de espacio de tablas temporal
- Histograma de tiempo de vida de la actividad
- Histograma de tiempo de cola de la actividad
- Histograma de tiempo de ejecución de la actividad

EXTENDED
Especifica que todos los datos de actividades agregados deben capturarse para las actividades asociadas con la clase de trabajo

para la que se define esta acción de trabajo y enviarse al supervisor de sucesos de estadísticas, si hay alguno activo. Ello incluye todos los datos de actividad agregados básicos más:

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

ENABLE o DISABLE
Especifica si la acción de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

ENABLE
Especifica que la acción de trabajo está habilitada y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE
Especifica que la acción de trabajo está inhabilitada y no va a tomarse en consideración cuando se sometan actividades de base de datos.

ENABLE o DISABLE
Especifica si el conjunto de acciones de trabajo va a tomarse en consideración o no cuando se sometan actividades de base de datos. El valor por omisión es ENABLE.

ENABLE
Especifica que el conjunto de acciones de trabajo está habilitado y va a tomarse en consideración cuando se sometan actividades de base de datos.

DISABLE
Especifica que el conjunto de acciones de trabajo está inhabilitado y no va a tomarse en consideración cuando se sometan actividades de base de datos.

cláusula-plantilla-histograma
Especifica las plantillas de histograma que han de utilizarse al recopilar datos de actividad agregados para las actividades asociadas a la clase de trabajo a la que se asigna esta acción de trabajo. Los datos de actividades agregadas sólo se recopilan para la clase de trabajo cuando el tipo de acción de trabajo es COLLECT AGGREGATE ACTIVITY DATA.

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos de las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- en ejecución durante un intervalo específico. Este tiempo incluye tanto el tiempo que está en cola como el tiempo de ejecución. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en cola durante un intervalo específico. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila

cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre la duración, en microsegundos que las actividades de DB2 -asociadas a la clase de trabajo a la que está asignada esta acción de trabajo- están en ejecución durante un intervalo específico. Este tiempo no incluye el tiempo que se pasa en la cola. El tiempo de ejecución de la actividad se recopila en este histograma en cada una de las particiones de base de datos en las que se ejecuta la actividad. En la partición de base de datos del coordinador de la actividad, este es el tiempo de ejecución integral (es decir, el tiempo de vida menos el tiempo que se ha pasado en la cola). En particiones de base de datos no de coordinador, este es el tiempo que estas particiones pasan trabajando en nombre de la actividad. Durante la ejecución de una determinada actividad, es posible que DB2 presente el trabajo en una partición de base de datos remota más de una vez y cada una de las veces la partición remota recopilará el tiempo de ejecución para dicha aparición de la actividad. Por tanto, es posible que los números del histograma de tiempo de ejecución no representen el número real de actividades exclusivas que se ejecutan en una partición de base de datos. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción BASE o EXTENDED.

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el coste estimado, en activaciones de temporización, de actividades DML asociadas a la clase de trabajo a la que se asigna esta clase de servicio. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *nombre-plantilla*
Especifica la plantilla que describe el histograma utilizado para recopilar datos estadísticos sobre el espacio de tiempo, en microsegundos, entre la llegada de una actividad DML y la llegada de la siguiente actividad DML, para cualquier actividad asociada a la clase de trabajo a la que se asigna esta acción de trabajo. El valor por omisión es SYSDEFAULTHISTOGRAM. Esta información sólo se recopila cuando se especifica la cláusula COLLECT AGGREGATE ACTIVITY DATA, con la opción EXTENDED.

Normas

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

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

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Cree un conjunto de acciones de trabajo denominado DATABASE_ACTIONS que se aplique a todas las actividades de base de datos. Utilice el conjunto de clases de trabajo LARGE_QUERIES y defina las siguientes acciones de trabajo. La acción de trabajo ONE_CONCURRENT_QUERY tiene una acción de umbral que permita que una consulta simultánea se ejecute en el sistema a la vez para las consultas que caigan en la clase de trabajo LARGE_ESTIMATED_COST. Si se supera dicho umbral, el gestor de base de datos va a poner en cola la actividad, pero no va a permitir que se ponga en cola más de una actividad la vez. Si se supera el umbral, no se permite la ejecución de la actividad de base de datos. La acción de trabajo TWO_CONCURRENT_QUERIES tiene una acción de umbral que permite que dos consultas simultáneas se ejecuten a la vez para las consultas que caigan dentro de la clase de trabajo LARGE_CARDINALITY y no permite que se pongan en cola más de dos. Si se van a poner en cola más de dos consultas, la actividad de base de datos va a seguir colocando las consultas en la cola y va a recopilar los datos de actividad de base de datos en el supervisor de sucesos de actividades, si hay alguno activo.

```
CREATE WORK ACTION SET DATABASE_ACTIONS
  FOR DATABASE USING WORK CLASS SET LARGE_QUERIES
    (WORK ACTION ONE_CONCURRENT_QUERY ON WORK CLASS LARGE_ESTIMATED_COST
     WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1
       STOP EXECUTION,
     WORK ACTION TWO_CONCURRENT_QUERIES ON WORK CLASS LARGE_CARDINALITY
     WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 2
       COLLECT ACTIVITY DATA CONTINUE)
```

Ejemplo 2: Cree un conjunto de acciones de trabajo denominada ADMIN_APPS_ACTIONS con una acción de trabajo denominada MAP_SELECTS que va a aplicarse a las actividades de base de datos que se ejecutan bajo la superclase de servicio ADMIN_APPS. La acción de trabajo va a correlacionar toda la actividad de base de datos que caiga dentro de la clase de trabajo SELECT_CLASS con la subclase de servicio SELECTS_SERVICE_CLASS, que está en el conjunto de clases de trabajo DML_SELECTS.

```
CREATE WORK ACTION SET ADMIN_APPS_ACTIONS
  FOR SERVICE CLASS ADMIN_APPS USING
    WORK CLASS SET DML_SELECTS
      (WORK ACTION MAP_SELECTS ON WORK CLASS SELECT_CLASS
       MAP TO SELECTS_SERVICE_CLASS)
```

CREATE WORK CLASS SET

La sentencia CREATE WORK CLASS SET define un conjunto de clases de trabajo.

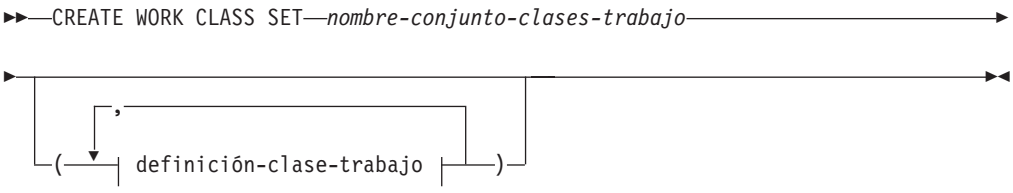
Invocación

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

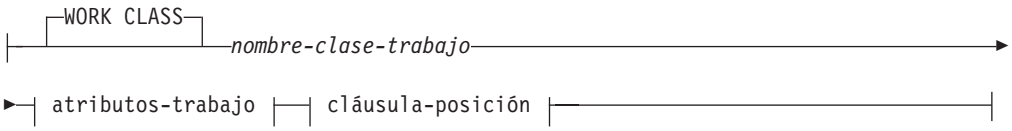
Autorización

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

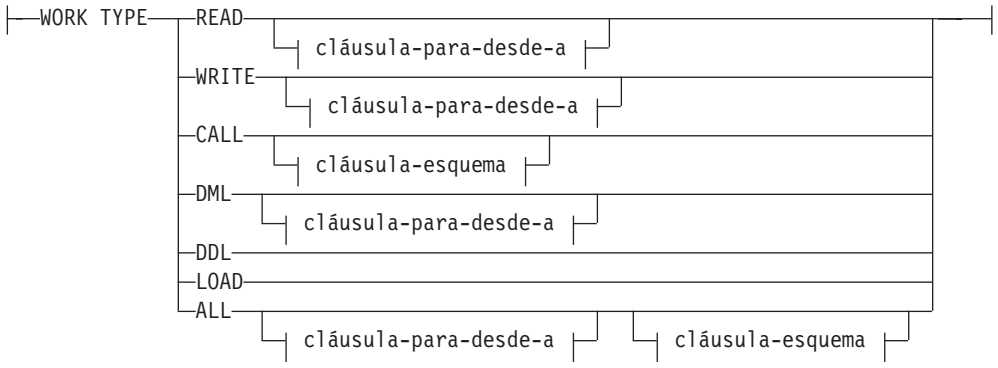
Sintaxis



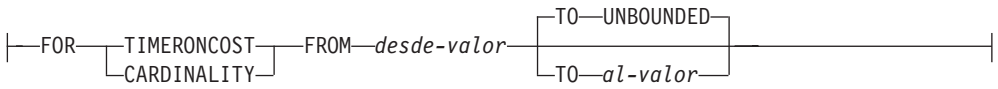
definición-clase-trabajo:



atributos-trabajo:

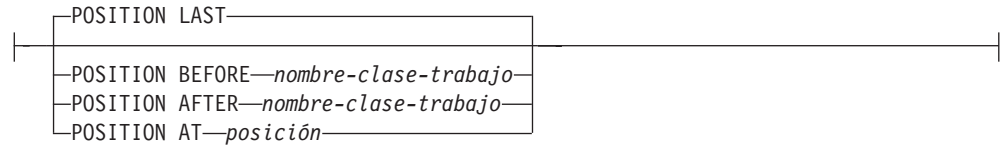


cláusula-para-desde-a:



cláusula-esquema:

cláusula-posición:



Descripción

nombre-conjunto-clases-trabajo
Da nombre al conjunto de clases de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-conjunto-clases-trabajo* no debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

definición-clase-trabajo
Especifica la definición de la clase de trabajo.

WORK CLASS *nombre-clase-trabajo*
Da nombre a la clase de trabajo. El *nombre-clase-trabajo* no debe identificar una clase de trabajo que ya existe en el conjunto de clases de trabajo del servidor actual (SQLSTATE 42710). El *nombre-clase-trabajo* no puede comenzar por 'SYS' (SQLSTATE 42939).

atributos-trabajo
Los atributos de la actividad de base de datos deben corresponderse con todos los atributos especificados en esta clase de trabajo si va a asociarse dicha actividad con esta clase de trabajo.

WORK TYPE
Especifica el tipo de actividad de base de datos.

READ
Esta actividad incluye las sentencias siguientes:

- Todas las sentencias SELECT o SELECT INTO que no contengan una sentencia DELETE, INSERT, MERGE o UPDATE y todas las sentencias VALUES INTO
- Todas las sentencias XQuery

WRITE
Esta actividad incluye las sentencias siguientes:

- UPDATE
- DELETE
- INSERT
- MERGE
- Todas las sentencias SELECT que contengan una sentencia DELETE, INSERT o UPDATE y todas las sentencias VALUES INTO
- Todas las sentencias XQuery

CALL
Incluye la sentencia CALL. Se toma en consideración una sentencia CALL para una clase de trabajo con un tipo de trabajo de CALL o ALL.

DML
Incluye las sentencias listadas bajo READ y WRITE.

DDL
Esta actividad incluye las sentencias siguientes:

- ALTER
- CREATE
- COMMENT
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- FLUSH PACKAGE CACHE
- GRANT
- REFRESH TABLE
- RENAME
- REVOKE
- SET INTEGRITY

LOAD
Operaciones de carga DB2.

ALL
Toda actividad de gestión de carga de trabajo (WLM) reconocida que caiga bajo una de las palabras clave descritas anteriormente.

FOR
Indica el tipo de información que está especificándose en la cláusula FROM *desde-valor* TO *al-valor*. La cláusula FOR sólo se utiliza para los siguientes tipos de trabajo:

- READ
- WRITE
- DML
- ALL

TIMERONCOST
El coste estimado del trabajo, en acciones de temporización. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

CARDINALITY
La cardinalidad estimada del trabajo. Este valor se utiliza para determinar si el trabajo cae dentro del rango especificado en la cláusula FROM *desde-valor* TO *al-valor*.

FROM *desde-valor* **TO UNBOUNDED o FROM** *desde-valor* **TO** *al-valor*
Especifica el rango de valor de activación de temporizador (para el coste estimado) o cardinalidad en la que debe caer la actividad de la base de datos si va a formar parte de esta clase de trabajo. El rango incluye *desde-valor* y *al-valor*. Si esta cláusula no se especifica para la clase de trabajo, se incluirá todo trabajo que caiga en el tipo de trabajo

especificado (es decir, el valor por omisión es FROM 0 TO UNBOUNDED). Este rango sólo se utiliza para los siguientes tipos de trabajo:

- READ
- WRITE
- DML
- ALL

FROM *desde-valor* **TO UNBOUNDED**
desde-valor debe ser cero o un valor DOUBLE positivo (SQLSTATE 5U019). El rango no tiene vinculación superior.

FROM *desde-valor* **TO** *al-valor*
desde-valor debe ser cero o un valor DOUBLE positivo y *al-valor* debe ser un valor DOUBLE positivo. *desde-valor* debe ser igual o más pequeño que *al-valor* (SQLSTATE 5U019).

cláusula-esquema

ROUTINES IN SCHEMA *nombre-esquema*
Especifica el nombre de esquema del procedimiento que estará llamando la sentencia CALL. Esta cláusula sólo se utiliza si el tipo de trabajo es CALL o ALL y la actividad de base de datos es una sentencia CALL. Si no se especifica ningún valor, se incluirán todos los esquemas.

cláusula-posición

POSITION
Especifica el lugar en el que va a colocarse esta clase de trabajo dentro del conjunto de clases de trabajo, lo cual determina el orden en el que se evalúan estas clases de trabajo. Al efectuar la asignación de clases de trabajo en tiempo de ejecución, el gestor de base de datos determina en primer lugar el conjunto de clases de trabajo asociado al objeto, la base de datos o una superclase de servicio. Se selecciona la primera clase de trabajo coincidente en dicho conjunto de clases de trabajo. Si no se especifica esta palabra clave, la clase de trabajo se ubica en la última posición.

LAST
Especifica que la clase de trabajo va a colocarse en última posición en la lista ordenada de las clases de trabajo del conjunto de clases de trabajo. Éste es el valor por omisión.

BEFORE *nombre-clase-trabajo*
Especifica que la clase de trabajo va a colocarse antes de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AFTER *nombre-clase-trabajo*
Especifica que la clase de trabajo va a colocarse detrás de la clase de trabajo *nombre-clase-trabajo* de la lista. El *nombre-clase-trabajo* debe identificar una clase de trabajo del conjunto de clases de trabajo que existe en el servidor actual (SQLSTATE 42704).

AT *posición*
Especifica la posición absoluta en la que la carga de trabajo va a colocarse en el conjunto de clases de trabajo en la lista ordenada de clases de trabajo. Este valor puede ser cualquier valor entero

positivo (SQLSTATE 42615). Si *posición* es mayor que el número de clases de trabajo existentes más una, la clase de trabajo se sitúa en la última posición del conjunto de clases de trabajo.

Normas

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

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.

Ejemplos

Ejemplo 1: Cree un conjunto de clases de trabajo denominado LARGE_QUERIES que tenga un conjunto de clases de trabajo que representen todo el DML con un coste estimado superior a 9999 y una cardinalidad estimada superior a 1000.

```
   CREATE WORK CLASS
SET LARGE_QUERIES
     (WORK CLASS LARGE_ESTIMATED_COST WORK TYPE DML
      FOR TIMERONCOST FROM 9999 TO UNBOUNDED,
      WORK CLASS LARGE_CARDINALITY WORK TYPE DML
      FOR CARDINALITY FROM 1000 TO UNBOUNDED)
```

Ejemplo 2: Cree una clase de trabajo denominada DML_SELECTS que tenga una clase de trabajo que represente a todas las sentencias DML SELECT que no contengan una sentencia DELETE, INSERT, MERGE o UPDATE.

```
   CREATE WORK CLASS SET DML_SELECTS
     (WORK CLASS SELECT_CLASS WORK TYPE READ)
```

CREATE WORKLOAD

La sentencia CREATE WORKLOAD define una carga de trabajo.

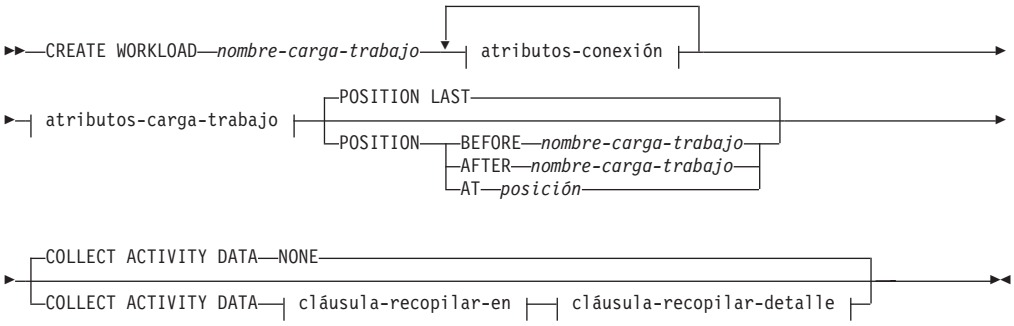
Invocación

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

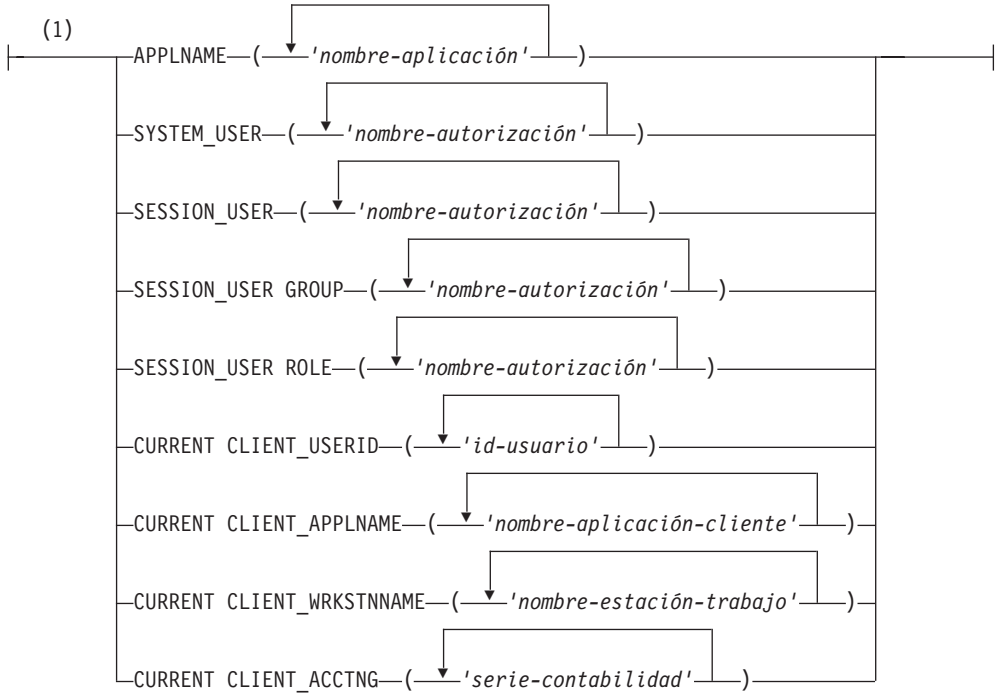
Autorización

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

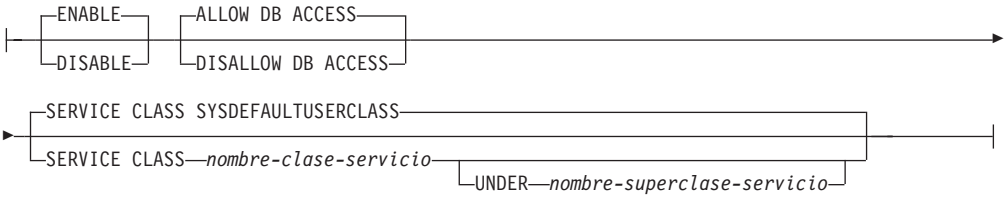
Sintaxis



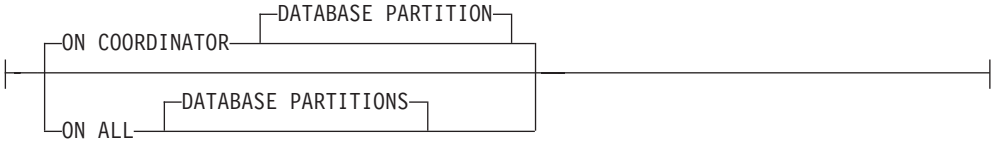
atributos-conexión:



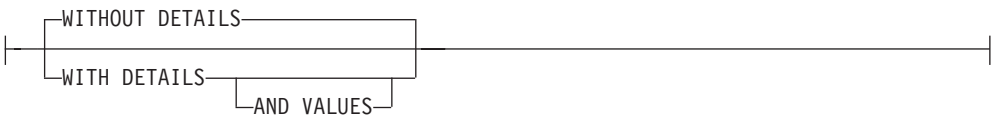
atributos-carga-trabajo:



recopilar-en-cláusula:



cláusula-detalle-recopilación:



Notas:

1 Cada cláusula de atributo de conexión sólo puede especificarse una vez.

Descripción

nombre-carga-trabajo
Da nombre a la carga de trabajo. Este nombre consta de una sola parte. Se trata de un identificador de SQL (ordinario o delimitado). El *nombre-carga-trabajo* no debe identificar una carga de trabajo que ya exista en el servidor actual (SQLSTATE 42710). El nombre no debe empezar por los caracteres 'SYS' (SQLSTATE 42939).

atributos-conexión
Los atributos de la conexión deben corresponderse con todos los atributos especificados en esta definición de carga de trabajo si va a asociarse con esta carga de trabajo cuando se establezca la conexión. Si se especifica una lista de valores para un atributo de conexión de la definición de carga de trabajo, el atributo correspondiente de la conexión debe corresponderse con al menos uno de los valores de la lista. Si no se especifica un atributo de conexión en la definición de carga de trabajo, la conexión puede tener cualquier valor para el atributo de conexión correspondiente.

Nota: Todos los atributos de conexión son sensibles a las mayúsculas y minúsculas.

APPLNAME ('*nombre-aplicación***', ...)**
Especifica una o más aplicaciones para el atributo de conexión de APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SYSTEM_USER ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SYSTEM USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION USER. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER GROUP ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER GROUP. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

SESSION_USER ROLE ('*nombre-autorización***', ...)**
Especifica uno o más ID de autorización para el atributo de conexión SESSION_USER ROLE. Los roles de un ID de autorización de sesión en este contexto hacen referencia a todas las funciones a disposición del ID de autorización de sesión, sin tener en cuenta el modo en que se obtuvieron las funciones. Un ID de autorización no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_USERID ('*id-usuario***', ...)**
Especifica uno o más ID de usuario cliente para el atributo de conexión CURRENT CLIENT_USERID. Un ID de usuario cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_APPLNAME ('*nombre-aplicación-cliente***', ...)**
Especifica una o más aplicaciones para el atributo de conexión de CURRENT CLIENT_APPLNAME. Un nombre de aplicación no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_WRKSTNNAME ('*nombre-estación-trabajo***', ...)**
Especifica uno o más nombres de estación de trabajo cliente para el atributo de conexión CURRENT CLIENT_WRKSTNNAME. Un nombre de estación de trabajo cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

CURRENT CLIENT_ACCTNG ('*serie-contabilidad***', ...)**
Especifica uno o más series de contabilidad cliente para el atributo de conexión CURRENT CLIENT_ACCTNG. Una serie de contabilidad cliente no puede aparecer más de una vez en la lista (SQLSTATE 42713).

atributos-carga-trabajo
Especifica los atributos de la carga de trabajo.

ENABLE o DISABLE
Especifica si esta carga de trabajo se tomará o no en consideración cuando se seleccione una carga de trabajo. El valor por omisión es ENABLE.

ENABLE
Especifica que la carga de trabajo se habilitará y se tomará en consideración cuando se seleccione una carga de trabajo.

DISABLE
Especifica que la carga de trabajo se inhabilitará y no se tomará en consideración cuando se seleccione una carga de trabajo.

ALLOW DB ACCESS o DISALLOW DB ACCESS
Especifica si se permitirá o no la aparición de carga de trabajo asociada a esta carga de trabajo acceder a la base de datos. El valor por omisión es ALLOW DB ACCESS.

ALLOW DB ACCESS
Especifica que se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos.

DISALLOW DB ACCESS
Especifica que no se permitirá a las apariciones de carga de trabajo asociadas a esta carga de trabajo acceder a la base de datos. La siguiente unidad de trabajo asociada a esta carga de trabajo se rechazará (SQLSTATE 5U020). Se permite que se completen las apariciones de carga de trabajo que ya se estén ejecutando.

SERVICE CLASS *nombre-clase-servicio*
Especifica que las peticiones asociadas a esta carga de trabajo se ejecuten en la clase de servicio *nombre-clase-servicio*. El *nombre-clase-servicio* debe identificar una clase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-clase-servicio* no puede ser 'SYSDEFAULTSUBCLASS', 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032). El valor por omisión es SYSDEFAULTUSERCLASS.

UNDER *nombre-superclase-servicio*
La cláusula se utiliza al especificar una subclase de servicio. El *nombre-superclase-servicio* identifica la superclase de servicio de *nombre-superclase-servicio*. El *nombre-superclase-servicio* debe identificar una superclase de servicio que exista en el servidor actual (SQLSTATE 42704). El *nombre-superclase-servicio* no puede ser 'SYSDEFAULTSYSTEMCLASS' o 'SYSDEFAULTMAINTENANCECLASS' (SQLSTATE 5U032).

POSITION
Especifica el lugar en el que va a situarse esta carga de trabajo en la lista ordenada de cargas de trabajo. En tiempo de ejecución, se examina esta lista por orden para buscar la primera carga de trabajo que coincida con los atributos de conexión necesarios. El valor por omisión es LAST.

LAST
Especifica que la carga de trabajo va a ser la última de la lista, antes de las cargas de trabajo por omisión SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

BEFORE *nombre-carga-trabajo-relativa*
Especifica que la carga de trabajo va a colocarse antes de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción BEFORE no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AFTER *nombre-carga-trabajo-relativa*
Especifica que la carga de trabajo va a colocarse detrás de la carga de trabajo *nombre-carga-trabajo-relativa* de la lista. El *nombre-carga-trabajo-relativa* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). La opción AFTER no puede especificarse si *nombre-carga-trabajo-relativa* es 'SYSDEFAULTUSERWORKLOAD' o 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

AT *posición*
Especifica la posición absoluta en la que la carga de trabajo va a colocarse en la lista. Este valor puede ser cualquier valor entero positivo (SQLSTATE 42615). Si *posición* es mayor que el número de cargas de trabajo existentes más una, la carga de trabajo se sitúa en la última posición justo antes de SYSDEFAULTUSERWORKLOAD y SYSDEFAULTADMWORKLOAD.

COLLECT ACTIVITY DATA
Especifica que los datos sobre cada una de las actividades asociadas con esta carga de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza. El valor por omisión es COLLECT ACTIVITY DATA NONE.

recopilar-en-cláusula
Especifica el lugar en el que se recopilarán los datos de actividad. El valor por omisión es ON COORDINATOR DATABASE PARTITION.

ON COORDINATOR DATABASE PARTITION
Especifica que los datos de actividad sólo van a recopilarse en la partición de la base de datos del coordinador de la actividad.

ON ALL DATABASE PARTITIONS
Especifica que los datos de actividad van a recopilarse en todas las particiones de la base de datos en los que se procesa la actividad; sin embargo los valores o detalles de la actividad sólo se recopilarán en la partición de la base de datos del coordinador de la actividad.

NONE
Especifica que los datos de actividad no deberían recopilarse para cada una de las actividades que esté asociada con esta carga de trabajo.

cláusula-detalle-recopilación
Especifica el tipo de datos de actividad que se van a recopilar. El valor por omisión es WITHOUT DETAILS.

WITHOUT DETAILS
Especifica que los datos sobre cada una de las actividades asociadas con esta carga de trabajo se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, cuando la actividad finaliza su ejecución. El entorno de compilación y sentencia no se envía al supervisor de sucesos.

WITH DETAILS
Especifica que la información de entorno de compilación y sentencia se va a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

AND VALUES
Especifica que los valores de datos de entrada se van a enviar al supervisor de sucesos de actividades, si hay alguno activo, para las actividades que disponen de ellos.

Normas

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

  - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
  - CREATE WORK ACTION SET, ALTER WORK ACTION SET o DROP (WORK ACTION SET)
  - CREATE WORK CLASS SET, ALTER WORK CLASS SET o DROP (WORK CLASS SET)
  - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
  - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- Una sentencia de SQL exclusiva de WLM no puede emitirse en una transacción global (SQLSTATE 51041) como por ejemplo, una transacción XA.

Notas

- Los cambios se graban en el catálogo del sistema, pero no surten efecto hasta que se confirmen, incluso para la conexión que emite la sentencia.
- Cuando se establece una conexión de base de datos, el gestor de base de datos busca una carga de trabajo coincidente basada en los atributos de conexión especificados en la cláusula POSITION (por orden de especificación). Si se encuentra una carga de trabajo coincidente, el gestor de base de datos compruebe si el usuario de sesión actual tiene privilegio de USAGE en dicha carga de trabajo. Si el usuario de sesión no tiene privilegio de USAGE en la carga de trabajo, el gestor de base de datos busca la siguiente carga de trabajo coincidente. Si el usuario de sesión tiene privilegio de USAGE en esta carga de trabajo, la conexión se asociará a la carga de trabajo. Si no se encuentra una carga de trabajo coincidente, la conexión se asocia a la carga de trabajo de usuario por omisión, SYSDEFAULTUSERWORKLOAD. Si el usuario de sesión no tiene privilegio de USAGE en SYSDEFAULTUSERWORKLOAD, se devuelve un error (SQLSTATE 42501).
- La asociación de carga de trabajo vuelve a evaluarse al principio de cada unidad de trabajo nueva si el gestor de base de datos detecta una de las siguientes condiciones:
  - Han cambiado los atributos de conexión. Esto puede suceder si se han producido alguno de los sucesos siguientes:
    - Se ha invocado la API de información de cliente establecido (sqleseti) y se ha cambiado los atributos de conexión incluidos en la definición de carga de trabajo. Tenga en cuenta que aunque la información de cliente puede establecerse por medio del usuario final de modo que pueda iniciarse una reevaluación de carga de trabajo, la propia acción de correlación de carga de trabajo podría producirse si el usuario de sesión no tiene el privilegio de USAGE en la carga de trabajo.
    - La sentencia SET SESSION AUTHORIZATION se ha invocado y ha cambiado el usuario de sesión actual.
    - Han cambiado los roles que están disponibles para un usuario de sesión.
  - Se crea una carga de trabajo.
  - Se descarta una carga de trabajo.
  - Se modifica una carga de trabajo.
  - El privilegio de USAGE de una carga de trabajo se otorga a un usuario, grupo o función.
  - El privilegio de USAGE de una carga de trabajo se revoca de un usuario, grupo o función.

Si la reevaluación de la carga de trabajo da como resultado que no se reasigne ninguna carga de trabajo, la aparición de carga de trabajo actual sigue ejecutándose; es decir, no se iniciará una aparición de carga de trabajo nueva.

- No puede reasignarse una conexión a una carga de trabajo diferente cuando una actividad siga activa. Una operación de carga, un procedimiento de ejecución o las sentencias que mantengan recursos entre varias unidades de trabajo, por ejemplo un cursor de WITH HOLD de apertura son ejemplos de tales actividades. La aparición de carga de trabajo actual sigue ejecutándose hasta que se completan todas las actividades de ejecución. La reasignación de la carga de trabajo se produce al principio de la siguiente unidad de trabajo.
- Una vez se haya hecho referencia a una clase de servicio por medio de una carga de trabajo, no podrá descartarse hasta que ya no se haga referencia al mismo por medio de ninguna carga de trabajo. Para eliminar una referencia de clase de servicio desde una carga de trabajo puede adoptarse cualquiera de las acciones siguientes:
  - Modificar la carga de trabajo para cambiar el nombre de clase de servicio
  - Descartar la carga de trabajo
- Una vez se haya hecho referencia a una función por medio de una carga de trabajo, no podrá descartarse hasta que ya no se haga referencia al mismo por medio de ninguna carga de trabajo. Para eliminar una referencia de función desde una carga de trabajo puede adoptarse cualquiera de las acciones siguientes:
  - Modificar la carga de trabajo para eliminar la función
  - Descartar la carga de trabajo

Ejemplos

Ejemplo 1: Cree una carga de trabajo denominada CAMPAIGN para las peticiones que envía un usuario de sesión que pertenece al grupo FINANCE. Estas peticiones van a ejecutarse en la clase de servicio de usuario por omisión SYSDEFAULTUSERCLASS.

```
CREATE WORKLOAD CAMPAIGN
  SESSION_USER GROUP ('FINANCE')
```

Ejemplo 2: Cree una carga de trabajo denominada PAYROLL para un usuario de sesión con la función HR que tenga el registro especial CURRENT CLIENT_APPLNAME establecido en SALARYSYS. Las unidades de trabajo asociadas a esta carga de trabajo van a ejecutarse en la clase de servicio MEDIUMSC que está bajo la superclase de servicio HRSC. Cuando se selecciona una carga de trabajo en tiempo de ejecución, esta carga de trabajo debería evaluarse únicamente después de que se haya evaluado la carga de trabajo CAMPAIGN y se haya determinado que no coinciden.

```
CREATE WORKLOAD PAYROLL
  SESSION_USER ROLE ('HR')
  CURRENT CLIENT_APPLNAME ('SALARYSYS') SERVICE CLASS MEDIUMSC
  UNDER HRSC POSITION AFTER CAMPAIGN
```

Ejemplo 3: Una aparición de la carga de trabajo CAMPAIGN (del ejemplo 1) se está ejecutando en la actualidad en el sistema. Cree una carga de trabajo denominada NEWCAMPAIGN, también para las peticiones que envía un usuario de sesión que pertenezca al grupo FINANCE, pero sólo para las peticiones que se envían por medio de la aplicación DB2BP.EXE. Las peticiones asociadas a esta carga de trabajo van a ejecutarse en la clase de servicio MARKETINGSC. NEWCAMPAIGN debería evaluarse antes de CAMPAIGN.

```
CREATE WORKLOAD NEWCAMPAIGN
  SESSION_USER GROUP ('FINANCE')
  APPLNAME ('DB2BP.EXE') SERVICE CLASS MARKETINGSC
  POSITION BEFORE CAMPAIGN
```

La aparición de carga de trabajo en ejecución de CAMPAIGN continúa ejecutándose hasta que se complete la unidad de trabajo actual, en cuyo momento tiene lugar una reevaluación de carga de trabajo y a continuación, la conexión puede volver a correlacionarse con la carga de trabajo NEWCAMPAIGN.

Ejemplo 4: Cree una carga de trabajo denominada REPORTS para las peticiones sometidas por medio de la aplicación appl1, appl2 o appl3 por medio del usuario del sistema BOB o MARY.

```
CREATE WORKLOAD REPORTS
  APPLNAME ('appl1', 'appl2', 'appl3')
  SYSTEM_USER ('BOB', 'MARY')
```

CREATE WRAPPER

La sentencia CREATE WRAPPER registra un derivador en un servidor federado. Un derivador es un mecanismo mediante el cual un servidor federado puede interactuar con determinados tipos de fuentes de datos.

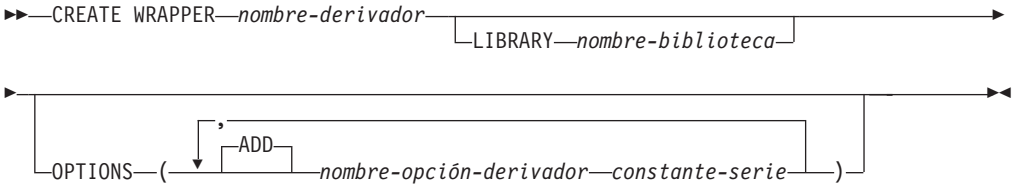
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-derivador
Nombra el derivador. Puede ser:

- Un nombre predefinido. Si se especifica un nombre predefinido, el servidor federado asigna automáticamente un valor por omisión a *nombre-biblioteca*.
- Un nombre suministrado por el usuario. Si se proporciona un nombre suministrado por el usuario, también se debe especificar el *nombre-biblioteca* adecuado que se debe utilizar con ese derivador y sistema operativo.

LIBRARY *nombre-biblioteca*
Nombra el archivo que contiene el módulo de biblioteca del derivador.

El nombre de biblioteca puede especificarse como un nombre de vía de acceso absoluta o, simplemente, puede especificarse el nombre base (sin la vía de acceso). Si sólo se especifica el nombre base, la biblioteca debe residir en el subdirectorio `lib` (UNIX) o en el subdirectorio `bin` (Windows) de la vía de acceso de DB2. El *nombre-biblioteca* se debe especificar entre comillas simples.

La opción LIBRARY sólo es necesaria si se utiliza un *nombre-derivador* proporcionado por el usuario. Esta opción no debe utilizarse cuando se proporciona un *nombre-derivador* predefinido.

OPTIONS (ADD *nombre-opción-derivador constante-serie*, **...)**
Las opciones de derivador se utilizan para configurar el derivador o para definir cómo DB2 utiliza el derivador. El *nombre-opción-derivador* es el nombre de la opción. La *constante-serie* especifica el valor para la opción de derivador.

La *constante-serie* se debe especificar entre comillas simples. Algunas opciones de derivador pueden utilizarlas todos los derivadores y otras son específicas de un derivador en particular.

Ejemplos

Ejemplo 1: Registre el derivador NET8 en un servidor federado para acceder a fuentes de datos Oracle. *NET8* es el nombre predefinido para el derivador que puede utilizar para acceder a las fuentes de datos Oracle.

```
CREATE WRAPPER NET8
```

Ejemplo 2: Registre un derivador en un servidor federado DB2 que utiliza el sistema operativo Linux para acceder a fuentes de datos ODBC. Asigne el nombre *odbc* al derivador que se registra en la base de datos federada. La vía de acceso completa de la biblioteca que contiene el gestor de controladores ODBC se define en la opción de derivador *MODULE '/usr/lib/odbc.so'*.

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

Ejemplo 3: Registre un derivador en un servidor federado DB2 que utiliza el sistema operativo Windows para acceder a fuentes de datos ODBC. El nombre de biblioteca para el derivador de ODBC es *'db2rcodbc.dll'*.

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

Ejemplo 4: Registre un derivador 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 derivador. En los servidores federados AIX, *libdb2lsentrez.a* es el archivo de biblioteca para el derivador Entrez. La opción EMAIL se necesita cuando el derivador 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 mediante el *nombre-sentencia*
- La *sentencia-select* especificada

Para cada tabla o vista que esté identificada (ya sea directamente o mediante un alias) en la sentencia SELECT del cursor, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

- Para cada tabla o vista identificada en la *sentencia-select*:
  - Privilegio SELECT para la tabla o vista, o bien
  - Privilegio CONTROL sobre la tabla o vista
- Autorización SYSADM o DBADM

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.

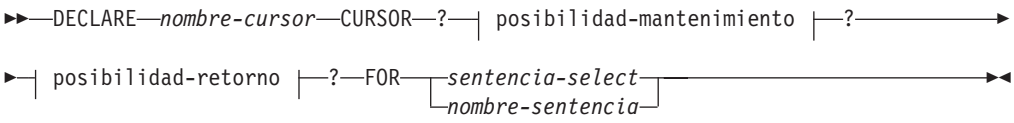
Si se especifica el *nombre-sentencia*:

- El ID de autorización de la sentencia es el ID de autorización de ejecución.
- La comprobación de autorización se lleva a cabo cuando se prepara la sentencia de selección.
- El cursor no puede abrirse si la sentencia-select no está preparada correctamente.

Si se especifica la *sentencia-select*:

- No se comprueban los privilegios GROUP.
- El ID de autorización de la sentencia es el ID de autorización especificado durante la preparación del programa.

Sintaxis



posibilidad de mantenimiento:

posibilidad de retorno:

Descripción

nombre-cursor
Especifica el nombre del cursor que se ha creado al ejecutar el programa fuente. Este nombre no debe ser el mismo que el de ningún otro cursor que esté declarado en el programa fuente. El cursor deberá abrirse para poder utilizarlo.

WITHOUT HOLD o WITH HOLD
Especifica si se debe impedir que el cursor se cierre como consecuencia de una operación de confirmación.

WITHOUT HOLD
No impide que el cursor se cierre como consecuencia de una operación de confirmación. Éste es el valor por omisión.

WITH HOLD
Mantiene recursos en varias unidades de trabajo. El efecto del atributo del cursor WITH HOLD es el siguiente:

- En las unidades de trabajo que finalizan con COMMIT:
  - Los cursores abiertos definidos con WITH HOLD permanecen abiertos. El cursor se sitúa antes de la siguiente fila lógica de la tabla de resultados.

    Si se emite la sentencia DISCONNECT después de la sentencia COMMIT para una conexión con cursores WITH HOLD, los cursores retenidos deben cerrarse explícitamente, porque si no se supondrá que la conexión ha realizado cierto trabajo (simplemente por tener abiertos cursores WITH HOLD aun sin haber emitido sentencias de SQL) y fallará la sentencia DISCONNECT.
  - Se liberan todos los bloqueos, excepto los bloqueos que protegen la posición actual del cursor, de los cursores WITH HOLD abiertos. Los bloqueos retenidos incluyen los bloqueos sobre la tabla y para los entornos paralelos, los bloqueos sobre las filas en las que los cursores están situados actualmente. Los bloqueos sobre paquetes y secciones de SQL dinámico (si las hay) se mantienen.
  - Las operaciones válidas en los cursores definidos WITH HOLD que están inmediatamente a continuación de una petición COMMIT son:
    - FETCH: Lee la siguiente fila del cursor.
    - CLOSE: Cierra el cursor.
  - UPDATE y DELETE CURRENT OF CURSOR sólo son válidas para aquellas filas que se recuperan dentro de la misma unidad de trabajo.

  - Se liberan los localizadores de LOB.
  - El conjunto de filas modificado por:
    - Una sentencia de cambio de datos
    - Rutinas que modifican datos SQL incorporados dentro de cursores WITH HOLD abiertos

    se confirma.
- Para las unidades de trabajo que finalizan con ROLLBACK:
  - Se cierran todos los cursores abiertos.
  - Se liberan todos los bloqueos adquiridos durante la unidad de trabajo.
  - Se liberan los localizadores de LOB.
- Para el caso especial de COMMIT:
  - Los paquetes se pueden volver a crear explícitamente, vinculándolos, o implícitamente, porque se han invalidado y, a continuación, se han vuelto a crear dinámicamente la primera vez que se ha hecho referencia a ellos. Todos los cursores retenidos se cierran cuando se vuelve a enlazar el paquete. Esto puede producir errores en una posterior ejecución.

WITHOUT RETURN o WITH RETURN
Especifica si la tabla de resultados del cursor está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento.

WITHOUT RETURN
Especifica que la tabla de resultados del cursor no está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento.

WITH RETURN
Especifica que la tabla de resultados del cursor está destinada a utilizarse como un conjunto de resultados a devolver de un procedimiento. WITH RETURN sólo es relevante si la sentencia DECLARE CURSOR está incluida junto con el código fuente de un procedimiento. En los demás casos, es posible que el precompilador acepte la cláusula, pero esto no tiene ningún efecto.

Dentro de un procedimiento de SQL, los cursores declarados mediante la cláusula WITH RETURN que todavía están abiertos cuando finaliza el procedimiento de SQL, definen los conjuntos de resultados del procedimiento de SQL. Todos los demás cursores abiertos de un procedimiento de SQL se cierran cuando finaliza el procedimiento de SQL. Dentro de un procedimiento externo (un procedimiento no definido utilizando LANGUAGE SQL), el valor por omisión para todos los cursores es WITH RETURN TO CALLER. Por lo tanto, todos los cursores que estén abiertos cuando el procedimiento finalice se considerarán conjuntos de resultados. Los cursores que se devuelven de un procedimiento no se pueden declarar como cursores desplazables.

TO CALLER
Especifica que el cursor puede devolver un conjunto de resultados al llamador. Por ejemplo, si el llamador es otro procedimiento, el conjunto de resultados se devuelve a ese procedimiento. Si el llamador es una aplicación cliente, el conjunto de resultados se devuelve a la aplicación cliente.

TO CLIENT
Especifica que el cursor puede devolver un conjunto de resultados a la aplicación cliente. Este cursor es invisible para cualquier

procedimiento anidado intermedio. Si una función, un método o un activador ha llamado al procedimiento directa o indirectamente, no se podrán devolver conjuntos de resultados al cliente y el cursor se cerrará tras completarse el procedimiento.

sentencia-select
Identifica la sentencia SELECT del cursor. La *sentencia-select* no debe incluir marcadores de parámetro, pero puede incluir referencias a las variables del lenguaje principal. Las declaraciones de variables del lenguaje principal deben preceder a la sentencia DECLARE CURSOR en el programa fuente.

nombre-sentencia
La sentencia SELECT del cursor es la sentencia SELECT preparada que se indica por el *nombre-sentencia* cuando está abierto el cursor. El *nombre-sentencia* no debe ser igual a ningún otro *nombre-sentencia* que esté especificado en otra sentencia DECLARE CURSOR del programa fuente.

Para obtener información acerca de las sentencias SELECT preparadas, consulte "PREPARE".

Notas

- Un programa invocado desde otro programa o desde otro archivo fuente dentro del mismo programa no puede utilizar el cursor que fue abierto por el programa llamador.
- Los procedimientos no anidados que no estén definidos con LANGUAGE SQL tendrán WITH RETURN TO CALLER como comportamiento por omisión si se especifica DECLARE CURSOR sin una cláusula WITH RETURN y el cursor se deja abierto en el procedimiento. Esto proporciona compatibilidad con procedimientos de versiones anteriores, en las cuales los procedimientos pueden devolver conjuntos de resultados a las aplicaciones cliente correspondientes. Para evitar este comportamiento, cierre todos los cursores abiertos en el procedimiento.
- Si la sentencia SELECT de un cursor contiene CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP, todas las referencias a estos registros especiales producirán el mismo valor "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 a los que llaman programas de aplicación que se han escrito mediante la CLI pueden utilizarse para definir conjuntos de resultados que se devuelvan directamente a la aplicación cliente. Los cursores de procedimientos SQL también pueden devolver resultados a un procedimiento de SQL llamador sólo si están definidos utilizando la cláusula WITH RETURN.
- Los cursores que se han declarado en rutinas que se invocan directa o indirectamente desde un cursor declarado como WITH HOLD, no heredan la opción WITH HOLD. De esta forma, a menos que el cursor de la rutina se haya definido explícitamente como WITH HOLD, una ejecución de COMMIT en la aplicación hará que se cierre.

  Consideremos la aplicación y los dos UDF siguientes:

```
Aplicación:

   DECLARE APPCUR CURSOR WITH HOLD FOR SELECT UDF1() ...
   OPEN APPCUR
   FETCH APPCUR ...
   COMMIT

UDF1:

   DECLARE UDF1CUR CURSOR FOR SELECT UDF2() ...
   OPEN UDF1CUR
```

```
   FETCH UDF1CUR ...

UDF2:

   DECLARE UDF2CUR CURSOR WITH HOLD FOR SELECT UDF2() ...
   OPEN UDF2CUR
   FETCH UDF2CUR ...
```

Después de que la aplicación haya buscado el cursor APPCUR, se abrirán los tres cursores. Cuando la aplicación emite la sentencia COMMIT, APPCUR sigue abierto, ya que se ha declarado como WITH HOLD. Sin embargo, en UDF1, el cursor UDF1CUR está cerrado, ya que no se ha definido con la opción WITH HOLD. Cuando se cierra el cursor UDF1CUR, se completan todas las invocaciones de rutina de la sentencia-select correspondiente (recibiendo una llamada final, si así se ha definido). UDF2 se completa, lo que da lugar a que UDF2CUR se cierre.

Ejemplos

Ejemplo 1: La sentencia DECLARE CURSOR asocia el nombre del cursor C1 con los resultados de la sentencia SELECT.

```
EXEC SQL  DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPARTMENT
  WHERE ADMRDEPT = 'A00';
```

Ejemplo 2: Supongamos que la tabla EMPLOYEE se ha modificado para añadir una columna generada, WEEKLYPAY, que calcula la paga semanal basada en el salario anual. Declare un cursor para recuperar el valor de la columna generada por el sistema de una fila que se va a insertar.

```
EXEC SQL DECLARE C2 CURSOR FOR
  SELECT E.WEEKLYPAY
  FROM NEW TABLE
    (INSERT INTO EMPLOYEE
     (EMPNO, FIRSTNME, MIDINIT, LASTNAME, EDLEVEL, SALARY)
     VALUES('000420', 'Peter', 'U', 'Bender', 16, 31842) AS E;
```

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 a 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

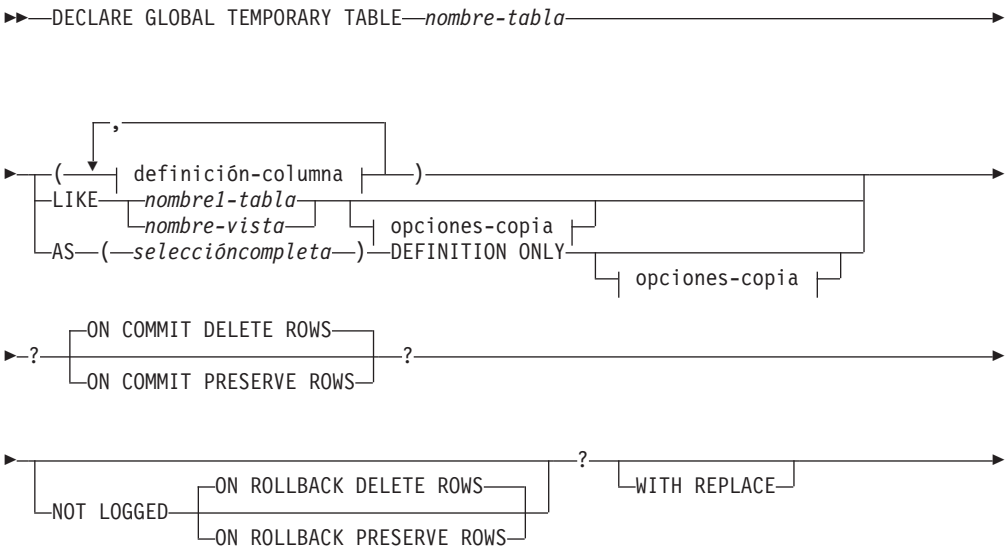
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

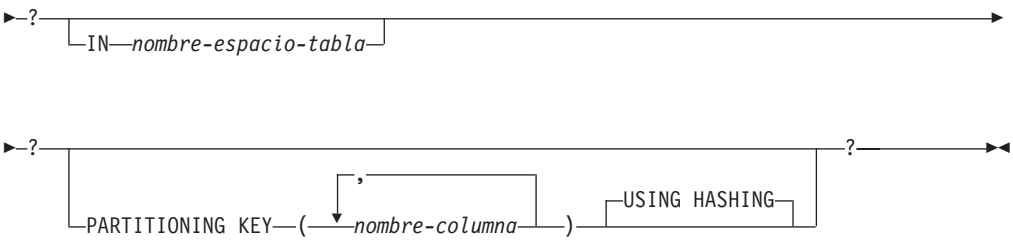
- Privilegio USE para el espacio de tablas USER TEMPORARY
- Autorización SYSADM o DBADM

Cuando se define una tabla utilizando LIKE o una selección completa, los privilegios del ID de autorización de la sentencia también deben incluir, como mínimo, uno de los siguientes para cada tabla o vista identificada:

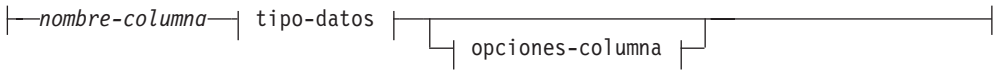
- Privilegio SELECT para la tabla o vista
- Privilegio CONTROL sobre la tabla o vista
- Autorización SYSADM o DBADM

Sintaxis

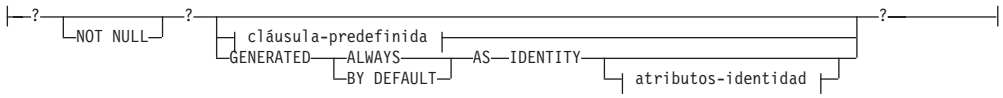




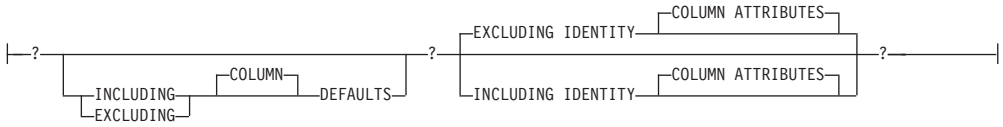
definición-columna:



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 debe ser superior a 4.005.
- Un tamaño de página de 8K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 8.101.
- Un tamaño de página de 16K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 16.293.
- Un tamaño de página de 32K con un máximo de 1.012 columnas, donde el número total de bytes de las columnas no debe ser superior a 32.677.

Para ver 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". Tenga en cuenta que los tipos BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC, XML, de referencia y estructurados no se pueden utilizar en tablas temporales globales declaradas (SQLSTATE 42962). Esta restricción incluye distintos tipos que tienen su fuente en estos tipos.

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 *nombre1-tabla* **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 identificada (incluidas las columnas implícitamente ocultas) o vista. También se definirá como implícitamente oculta una columna de la nueva tabla que se corresponde a una columna implícitamente oculta de la tabla existente.

- 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.

Si se identifica una tabla protegida en la cláusula LIKE, la tabla nueva se convierte en una tabla protegida.

Cuando una tabla se identifica en la cláusula LIKE y dicha tabla contiene una columna ROW CHANGE TIMESTAMP, la columna correspondiente de la nueva tabla hereda sólo el tipo de datos de la columna ROW CHANGE TIMESTAMP. La nueva columna no se considera una columna generada. La tabla identificada en la cláusula LIKE no debe contener una columna ROW CHANGE TIMESTAMP que también esté definida como IMPLICITLY HIDDEN (SQLSTATE 42867).

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 (*selecciónc**ompleta***) DEFINITION ONLY**
Especifica que las columnas de la tabla tienen el mismo nombre y descripción que las columnas que aparecerían en la tabla de resultados derivada de la selección completa si ésta tuviera que ejecutarse. 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.

La definición implícita incluye los atributos siguientes de las *n* columnas (si son aplicables al tipo de datos):

- Nombre de columna
- Tipo de datos, longitud, precisión y escala
- Capacidad de nulos

Los atributos siguientes no se incluyen (el valor por omisión y los atributos de identidad pueden incluirse utilizando las *opciones-copia*):

- Valor por omisión
- Atributos de identidad
- ROW CHANGE TIMESTAMP

La definición implícita no incluye ningún otro atributo opcional de las tablas o vistas a las que se hace referencia en la selección completa.

Cada elemento de la lista de selección debe tener un nombre exclusivo (SQLSTATE 42711). La cláusula AS puede utilizarse en la cláusula select para proporcionar nombres exclusivos. La selección completa no debe dar como resultado una columna que tenga un tipo de datos LOB. La selección completa no debe hacer referencia a variables del lenguaje principal ni incluir marcadores de parámetro.

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. Éste es el valor por omisión.

PRESERVE ROWS
Las filas de la tabla se conservan.

NOT LOGGED
Especifica que las operaciones de inserción, actualización o supresión efectuadas en la tabla no deben anotarse cronológicamente, pero que la creación o descarte de la tabla sí debe anotarse cronológicamente. Durante una operación ROLLBACK (o ROLLBACK TO SAVEPOINT):

- Si la tabla se había creado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla se descarta
- Si la tabla se había descartado dentro de una unidad de trabajo (o punto de salvaguarda), la tabla vuelve a crearse, pero sin datos

ON ROLLBACK
Especifica la acción que va a tener lugar en la tabla temporal global sin anotaciones cronológicas cuando se realice una operación ROLLBACK (o ROLLBACK TO SAVEPOINT).

DELETE ROWS
Si se han cambiado los datos de la tabla, se suprimirán todas las filas. Éste 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 descartan 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 ver más detalles, consulte el apartado “Tamaño de fila” en “CREATE TABLE”.

PARTITIONING KEY (*nombre-columna,...***)**
Especifica la clave de distribución utilizada cuando se distribuyen 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 distribución se define como la primera columna de la tabla temporal declarada.

Para las tablas temporales declaradas, en espacios de tablas definidos en grupos de particiones de base de datos de una sola partición, puede utilizarse cualquier colección de columnas para definir la clave de distribución. Si no especifica este parámetro, no se crea ninguna clave de distribución.

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

Notas

- 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 de 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 de SESSION.T:
    - Si se ha especificado NOT LOGGED, la operación de retroacción incluye la operación DELETE de SESSION.T a menos que también se haya especificado ON ROLLBACK PRESERVE ROWS
    - Si no se ha especificado NOT LOGGED, los cambios realizados en T se deshacen

    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).
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para z/OS:
    - La sintaxis siguiente se acepta como comportamiento por omisión:
      - CCSID ASCII
      - CCSID UNICODE

DELETE

La sentencia DELETE suprime filas de una tabla, un apodo o una vista o las tablas, los apodos o las 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 de donde van a suprimirse filas
- Privilegio CONTROL sobre la tabla, vista o apodo de donde 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

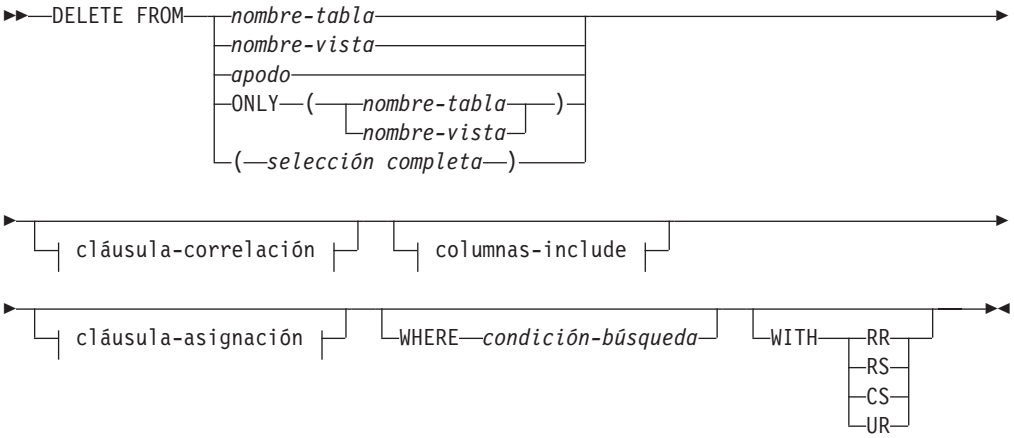
Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

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.

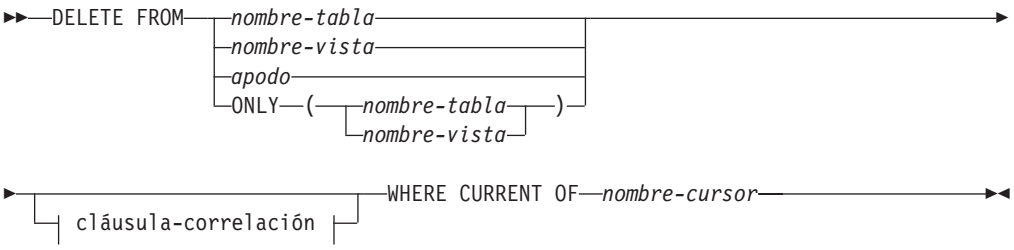
Sintaxis

supresión-búsqueda:

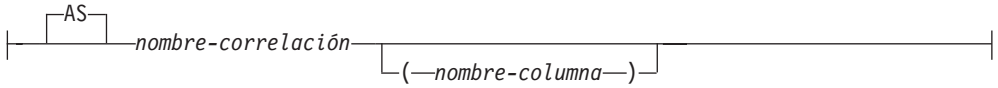


columnas-include:

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 efecto en la sentencia.

FROM ONLY (*nombre-vista***)**
Aplicable a las vistas con tipo, la palabra clave ONLY especifica que la sentencia sólo se debe aplicar a los datos de la vista especificada y no puede suprimir las filas de las subvistas correspondientes. Para una sentencia DELETE con posición, el cursor asociado también debe haber especificado la vista en la cláusula FROM utilizando ONLY. Si *nombre-vista* no es una vista con tipo, la palabra clave ONLY no tiene efecto en la sentencia.

cláusula-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 normas. Las *columnas-include* son las únicas columnas que se pueden definir utilizando la *cláusula-asignación* (SQLSTATE 42703).

WHERE
Especifica una condición que selecciona las filas que se deben suprimir. Puede omitirse la cláusula, se puede especificar una condición de búsqueda o nombrar un cursor. Si se omite la cláusula, se suprimen todas las filas de la tabla o vista.

condición-búsqueda
Cada *nombre-columna* de la condición de búsqueda, que no sea una subconsulta, debe identificar una columna de la tabla o vista.

La *condición-búsqueda* se aplica a cada fila de la tabla, vista o apodo y las filas que se suprimen son aquellas para las que el resultado de la *condición-búsqueda* es verdadero.

Si la condición de búsqueda contiene una subconsulta, puede considerarse que 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. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

Normas

- ***Activadores:*** Las sentencias DELETE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en las filas suprimidas. Si una sentencia DELETE de una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobará la integridad referencial de las actualizaciones que se han realizado en el activador y no la de las tablas subyacentes de la vista que ha dado lugar a que se ejecutara el activador.
- ***Integridad referencial:*** Si la tabla identificada o la tabla base de la vista identificada es padre, las filas seleccionadas para la supresión no deberán tener ningún dependiente en una relación con una norma de supresión de RESTRICT, y la sentencia DELETE no deberá aplicarse en cascada a las filas descendentes que tengan dependientes en una relación con una norma de supresión de RESTRICT.

  Si no se impide la operación de supresión por una norma de supresión RESTRICT, se suprimen las filas seleccionadas. Las filas que son dependientes de las filas seleccionadas también se ven afectadas:

  – Las columnas con posibilidad de contener nulos de las claves foráneas de cualquier fila que sea dependiente en una relación con una norma de supresión de SET NULL se establecen en el valor nulo.
  – Las filas que sean sus dependientes en una relación con una norma de supresión de CASCADE también se suprimen y se aplican, a su vez, las normas anteriores a estas filas.

  Se comprueba la norma de supresión de NO ACTION para imponer que las claves foráneas no nulas hagan referencia a una fila padre existente después de que se hayan impuesto otras restricciones de referencia.
- ***Política de seguridad:*** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
  – Acceso de grabación a todas las columnas protegidas (SQLSTATE 42512)
  – Acceso de grabación y de lectura a todas las filas que se han seleccionado para la supresión (SQLSTATE 42519)

Notas

- Si se produce un error durante la ejecución de una sentencia DELETE de múltiples filas, no se realiza ningún cambio en la base de datos.
- Salvo que ya existan los bloqueos adecuados, se adquieren uno o más bloqueos exclusivos durante la ejecución de una sentencia DELETE satisfactoria. La emisión de una sentencia COMMIT o ROLLBACK liberará los bloqueos. Hasta que se liberen los bloqueos por una operación de confirmación o de 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 las sentencias activadas, incluye el número de filas que se han insertado, actualizado o suprimido.
- Si se produce un error que impide la supresión de todas las filas que coincidan con la condición de búsqueda y todas las operaciones necesarias para las restricciones de referencia existentes, no se realiza ningún cambio en la tabla y se devuelve un error.
- Para los apodos, la opción del servidor externo `iud_app_svpt_enforce` plantea una limitación adicional. Consulte la documentación acerca de los objetos federados para obtener más información.
- Para algunas fuentes de datos, SQLCODE -20190 podría devolverse en una supresión realizada para un apodo debido a una posible falta de coherencia en los datos. Consulte la documentación acerca de los objetos federados para obtener más información.

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 (PARTITION BY LASTNAME ORDER BY FIRSTNME)
    FROM EMPLOYEE) AS E(RN)
    WHERE RN > 1
```

DESCRIBE

La sentencia DESCRIBE obtiene información acerca de un objeto. Con esta sentencia se pueden obtener dos tipos de información. Cada una de ellos se describe por separado.

- Marcadores de parámetro de entrada de una sentencia preparada. Obtiene información sobre los marcadores de parámetro de entrada en una sentencia preparada. Esta información se pone en un descriptor.
- La salida de una sentencia preparada. Obtiene información sobre una sentencia preparada o información sobre las columnas de lista de selección de una sentencia SELECT preparada. Esta información se pone en un descriptor.

DESCRIBE INPUT

La sentencia DESCRIBE INPUT obtiene información sobre los marcadores de parámetro de entrada de una sentencia preparada.

Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis

Descripción

nombre-sentencia
Identifica la sentencia preparada. Cuando se ejecuta la sentencia DESCRIBE INPUT, el nombre debe identificar una sentencia que el proceso de aplicación haya preparado en el servidor actual.

Para una sentencia CALL, la información devuelta describe los parámetros de entrada, definidos como IN o INOUT, del procedimiento. Los marcadores de parámetro de entrada se consideran siempre anulables, independientemente de su uso.

INTO *nombre-descriptor*
Identifica un área de descriptor de SQL (SQLDA). Antes de ejecutar la sentencia DESCRIBE INPUT, se debe establecer la variable siguiente en la SQLDA:

SQLN Especifica el número de apariciones de SQLVAR proporcionadas en la SQLDA. SQLN se debe establecer en un valor mayor que o igual a cero antes de que se ejecute la sentencia DESCRIBE INPUT.

Cuando se ejecuta la sentencia DESCRIBE INPUT, el gestor de bases de datos asigna valores a las variables de la SQLDA como se indica a continuación:

SQLDAID
Los 6 primeros bytes se establecen en 'SQLDA ' (es decir, 5 letras seguidas del carácter de espacio).

El séptimo byte, definido como SQLDOUBLED, se establece basándose en los marcadores de parámetro descritos:

- Si la SQLDA contiene dos entradas SQLVAR para cada parámetro de entrada, el séptimo byte se establece en '2'. Se utiliza esta técnica para acomodar los parámetros de entrada de tipo estructurado o LOB.
- De lo contrario, el séptimo byte se establece en el carácter de espacio.

El séptimo byte se establece en el carácter de espacio si no hay suficiente espacio en la SQLDA para incluir la descripción de todos los marcadores de parámetro de entrada.

El octavo byte se establece en el carácter de espacio.

SQLDABC
Longitud de la SQLDA en bytes.

SQLD Número de parámetros IN e INOUT del procedimiento.

SQLVAR
Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las apariciones de SQLVAR.

Si el valor de SQLD es *n*, donde *n* es mayor que 0 pero menor que o igual al valor de SQLN, se asignan valores a las primeras *n* apariciones de SQLVAR. Los valores describen los marcadores de los parámetros de entrada del procedimiento. La primera aparición de SQLVAR describe el primer marcador de parámetro de entrada, la segunda aparición de SQLVAR describe el segundo marcador de parámetro de entrada y así sucesivamente.

SQLVAR base

SQLTYPE
Código que muestra el tipo de datos del parámetro e indica si puede o no puede contener valores nulos.

SQLLEN
Valor de longitud que depende del tipo de datos del parámetro. SQLLEN es 0 para tipos de datos LOB.

SQLNAME
Si SQLVAR corresponde a un marcador de parámetro que no forma parte de una expresión de la lista de parámetros de un procedimiento, sqlname contiene el nombre del parámetro si se ha especificado alguno en la sentencia CREATE PROCEDURE.

SQLVAR secundaria

Estas variables sólo se utilizan si el número de entradas SQLVAR se dobla para acomodar parámetros LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN
Atributo de longitud de un parámetro BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME
Para cualquier parámetro de tipo definido por el usuario (diferenciado o estructurado), el gestor de base de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado. Para un parámetro de tipo de referencia, el gestor de base de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado del tipo de destino de la referencia. De lo contrario, el nombre de esquema es SYSIBM y el nombre de tipo es el nombre que aparece en la columna TYPENAME de la vista de catálogo SYSCAT.DATATYPES.

Notas

- **Preparación de la SQLDA:** Antes de que se ejecute la sentencia DESCRIBE INPUT, se debe asignar la SQLDA y el valor de SQLN se debe establecer en un valor mayor que o igual a cero para indicar cuántas apariciones de SQLVAR se proporcionan en la SQLDA. Se debe asignar suficiente almacenamiento para que contenga las apariciones de SQLN. Para obtener la descripción de los marcadores de parámetro de entrada de la sentencia preparada, el número de apariciones de SQLVAR no debe ser menor que el número de marcadores de

parámetro de entrada. Además, si los marcadores de parámetro de entrada incluyen LOB o tipos estructurados, el número de apariciones de SQLVAR debe ser el doble del número de marcadores de parámetro de entrada.

- Las conversiones de páginas de códigos entre las páginas de código UNIX ampliado (EUC) y las páginas de códigos DBCS o entre las páginas de código Unicode y no Unicode pueden producir la expansión o la contracción de las longitudes de caracteres.
- Si se está seleccionando un tipo estructurado, pero no se ha definido ninguna transformación FROM SQL (porque no se ha especificado ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 428EM) o porque el grupo mencionado no tiene una función de transformación FROM SQL definida (SQLSTATE 42744)), se devuelve 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 apariciones de SQLVAR para acomodar cualquier lista de selección que la aplicación vaya a tener que procesar. Si la tabla contiene cualquier columna de tipo LOB, tipo diferenciado, tipo estructurado o de tipo de referencia, el número de las SQLVAR debe ser el doble que el número máximo de columnas; de lo contrario, el número debe ser igual al número máximo de columnas. Después de realizar la asignación, la aplicación puede utilizar esta SQLDA repetidas veces.

  Esta técnica utiliza una gran cantidad de almacenamiento que nunca se desasigna, incluso cuando la mayor parte de su almacenamiento no se utilice para una lista de selección en particular.

  Segunda técnica: Repita los dos siguientes pasos para cada lista de selección procesada:

  1. Ejecute una sentencia DESCRIBE INPUT con una SQLDA que no tenga apariciones de SQLVAR; es decir, una SQLDA para la que SQLN sea cero. El valor devuelto para SQLD es el número de columnas de la tabla de resultados. Este es el número necesario de apariciones de SQLVAR o la mitad del número necesario. Puesto que no había ninguna entrada SQLVAR, se emitirá un aviso con SQLSTATE 01005. Si el SQLCODE que se asocia a ese aviso es +237, +238 o +239, el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD. (En la devolución de estos SQLCODE positivos se da por supuesto que el valor de la opción de enlace SQLWARN era YES (devolución de SQLCODE positivos). Si SQLWARN se había establecido en NO, sigue devolviéndose +238 para indicar que el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD.)
  2. Asigne una SQLDA con suficientes apariciones de SQLVAR. Después ejecute la sentencia DESCRIBE de nuevo, utilizado esta SQLDA nueva.

  Esta técnica permite una mejor gestión de almacenamiento que la primera técnica, pero dobla el número de sentencias DESCRIBE INPUT.

  Tercera técnica: Asigne una SQLDA que sea lo suficientemente grande para manejar la mayoría de, y quizá todas, las listas de selección pero que también sea razonablemente pequeña. Ejecute DESCRIBE INPUT y compruebe el valor SQLD. Utilice el valor SQLD para el número de apariciones de SQLVAR para asignar una SQLDA mayor, si es necesario.

  Esta técnica está comprendida entre las dos primeras técnicas. Su eficacia depende de una buena elección del tamaño de la SQLDA original.

Ejemplo

Ejecute una sentencia DESCRIBE INPUT con una SQLDA que tenga suficientes apariciones SQLVAR para describir cualquier número de parámetros de entrada que una sentencia preparada pueda tener. Suponga que es necesario describir cinco marcadores de parámetro como máximo y que los datos de entrada no contienen LOB.

```
     /* STMT1_STR contiene la sentencia INSERT con la cláusula VALUES */
EXEC SQL PREPARE STMT1_NAME FROM :STMT1_STR;
... /* código para establecer SQLN en 5 y asignar la SQLDA        */
EXEC SQL DESCRIBE INPUT STMT1_NAME INTO :SQLDA;
.
.
.
```

Este ejemplo utiliza la primera técnica descrita en “Asignación de la SQLDA” en “DESCRIBE OUTPUT”.

DESCRIBE OUTPUT

La sentencia DESCRIBE OUTPUT obtiene información acerca de una sentencia preparada.

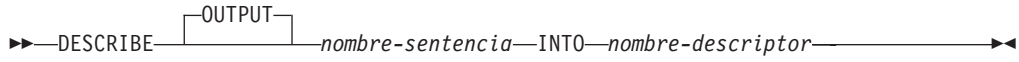
Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-sentencia
Identifica la sentencia preparada. Cuando se ejecuta la sentencia DESCRIBE OUTPUT, el nombre debe identificar una sentencia que el proceso de aplicación haya preparado en el servidor actual.

Si la sentencia preparada es una sentencia SELECT o VALUES INTO, la información devuelta describe las columnas en la tabla de resultados. Si la sentencia preparada es una sentencia CALL, la información devuelta describe los parámetros de salida, definidos como OUT o INOUT, del procedimiento.

INTO *nombre-descriptor*
Identifica un área de descriptor de SQL (SQLDA). Antes de ejecutar la sentencia DESCRIBE OUTPUT, se debe establecer la variable siguiente en la SQLDA:

SQLN Especifica el número de apariciones de SQLVAR proporcionadas en la SQLDA. SQLN debe establecerse en un valor mayor que o igual a cero antes de que se ejecute la sentencia DESCRIBE OUTPUT.

Cuando se ejecuta la sentencia DESCRIBE OUTPUT, el gestor de bases de datos asigna valores a las variables de la SQLDA de la siguiente manera:

SQLDAID
Los 6 primeros bytes se establecen en 'SQLDA ' (es decir, 5 letras seguidas del carácter de espacio).

El séptimo byte, definido como SQLDOUBLED, se establece basándose en las columnas de resultados o los marcadores de parámetro descritos:

- Si la SQLDA contiene dos entradas SQLVAR para cada columna o parámetro de salida, el séptimo byte se establece en '2'. Esta técnica se utiliza para acomodar columnas o parámetros de salida de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.
- De lo contrario, el séptimo byte se establece en el carácter de espacio.

El séptimo byte se establece en el carácter de espacio si no hay suficiente espacio en la SQLDA para incluir la descripción de todos los marcadores de parámetro de salida o las columnas de resultado.

El octavo byte se establece en el carácter de espacio.

SQLDABC
Longitud de la SQLDA en bytes.

SQLD Si la sentencia preparada es SELECT, SQLD se establece en el número de columnas de la tabla de resultados. Si la sentencia preparada es una sentencia CALL, SQLD se establece en el número de parámetros OUT e INOUT del procedimiento. De lo contrario, SQLD se establece en 0.

SQLVAR
Si el valor de SQLD es 0 o mayor que el valor de SQLN, no se asigna ningún valor a las apariciones de SQLVAR.

Si el valor de SQLD es *n*, donde *n* es mayor que 0 pero menor que o igual al valor de SQLN, se asigna valores a SQLTYPE, SQLLEN, SQLNAME, SQLLONGLEN y SQLDATATYPE_NAME para las primeras *n* apariciones de SQLVAR. Estos valores describen columnas de la tabla de resultados o marcadores de parámetro para los parámetros de salida del procedimiento. La primera aparición de SQLVAR describe la primera columna o el primer marcador de parámetro de salida, la segunda aparición de SQLVAR describe la segunda columna o el segundo marcador de parámetro de salida y así sucesivamente.

SQLVAR base

SQLTYPE
Código que muestra el tipo de datos de la columna o del parámetro e indica si puede o no puede contener valores nulos.

SQLLEN
Valor de longitud que depende del tipo de datos de la columna o del parámetro. SQLLEN es 0 para tipos de datos LOB.

SQLNAME
El valor de la variable sqlname se obtiene como se indica a continuación:

- Si SQLVAR corresponde a una columna que se ha obtenido para una referencia a una columna simple en la lista de selección de la sentencia-select, sqlname es el nombre de la columna.
- Si SQLVAR corresponde a un marcador de parámetro que no forma parte de una expresión en la lista de parámetros de un procedimiento, 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 o parámetros de tipo LOB, de tipo diferenciado, de tipo estructurado o de tipo de referencia.

SQLLONGLEN
El atributo de longitud de una columna o un parámetro BLOB, CLOB o DBCLOB.

SQLDATATYPE_NAME
Para cualquier columna o parámetro de tipo definido por el usuario (diferenciado o estructurado), el gestor de bases de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado. Para una columna o un parámetro de tipo de referencia, el gestor de bases de datos establece esta opción en el nombre de tipo definido por el usuario totalmente calificado del tipo de destino de la referencia. De lo contrario, el nombre de esquema es SYSIBM y el nombre de tipo es el nombre que aparece en la columna TYPENAME de la vista de catálogo SYSCAT.DATATYPES.

Notas

- Antes de ejecutar la sentencia DESCRIBE OUTPUT, el valor de SQLN debe establecerse de manera que indique cuántas apariciones de SQLVAR se proporcionan en la SQLDA y debe asignarse suficiente almacenamiento para incluir las apariciones de SQLN. Por ejemplo, para obtener la descripción de las columnas de la tabla de resultados de una sentencia SELECT preparada, el número de apariciones de SQLVAR no debe ser menor que el número de columnas.
- Si se espera un LOB de gran tamaño, tenga en cuenta que el manejo de este LOB afectará a la memoria de la aplicación. Si se da esta condición, considere la posibilidad de utilizar localizadores o variables de referencia a archivos. Modifique la SQLDA después de que se haya ejecutado la sentencia DESCRIBE OUTPUT pero antes de asignar almacenamiento para que un SQLTYPE de SQL_TYP_xLOB cambie a SQL_TYP_xLOB_LOCATOR o SQL_TYP_xLOB_FILE con los cambios correspondientes en otros campos, por ejemplo SQLLEN. Después asigne el almacenamiento basado en SQLTYPE y continúe.
- Las conversiones de páginas de códigos entre páginas de códigos UNIX ampliado (EUC) y páginas de códigos DBCS, o entre páginas de códigos Unicode y no Unicode, puede producir la expansión y la contracción de las longitudes de caracteres.
- Si se está seleccionando un tipo estructurado, pero no se define ninguna transformación FROM SQL (porque no se ha especificado ningún TRANSFORM GROUP utilizando el registro especial CURRENT DEFAULT TRANSFORM GROUP (SQLSTATE 428EM) o porque el grupo mencionado no tiene ninguna función de transformación FROM SQL definida (SQLSTATE 42744)), se devuelve 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 apariciones de SQLVAR para acomodar cualquier lista de selección que la aplicación vaya a tener que procesar. Si la tabla contiene cualquier columna de tipo LOB, tipo diferenciado, tipo estructurado o de tipo de referencia, el número de las SQLVAR debe ser el doble que el número máximo de columnas; de lo contrario, el número debe ser igual al número máximo de columnas. Después de realizar la asignación, la aplicación puede utilizar esta SQLDA repetidas veces.

  Esta técnica utiliza una gran cantidad de almacenamiento que nunca se desasigna, incluso cuando la mayor parte de su almacenamiento no se utilice para una lista de selección en particular.

  Segunda técnica: Repita los dos siguientes pasos para cada lista de selección procesada:

  1. Ejecute una sentencia DESCRIBE OUTPUT con una SQLDA que no tenga apariciones de SQLVAR; es decir, una SQLDA para la que SQLN es cero. El

valor devuelto para SQLD es el número de columnas de la tabla de resultados. Este es el número necesario de apariciones de SQLVAR o la mitad del número necesario. Puesto que no había ninguna entrada SQLVAR, se emitirá un aviso con SQLSTATE 01005. Si el SQLCODE que se asocia a ese aviso es +237, +238 o +239, el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD. (En la devolución de estos SQLCODE positivos se da por supuesto que el valor de la opción de enlace SQLWARN era YES (devolución de SQLCODE positivos). Si SQLWARN se había establecido en NO, sigue devolviéndose +238 para indicar que el número de entradas de SQLVAR debe ser el doble del valor que se devuelve en SQLD.)

2. Asigne una SQLDA con suficientes apariciones de SQLVAR. A continuación, ejecute la sentencia DESCRIBE OUTPUT de nuevo, utilizando esta SQLDA nueva.

Esta técnica permite una mejora gestión de almacenamiento que la primera técnica, pero doble el número de sentencias DESCRIBE OUTPUT.

Tercera técnica: Asigne una SQLDA que sea lo suficientemente grande para manejar la mayoría de, y quizá todas, las listas de selección pero que también sea razonablemente pequeña. Ejecute DESCRIBE y compruebe el valor SQLD. Utilice el valor SQLD para el número de apariciones de SQLVAR para asignar una SQLDA mayor, si es necesario.

Esta técnica está comprendida entre las dos primeras técnicas. Su eficacia depende de una buena elección del tamaño de la SQLDA original.

- ***Consideraciones para columnas ocultas implícitamente:*** una sentencia DESCRIBE OUTPUT devuelve información de una columna implícitamente oculta sólo si la columna está especificada explícitamente como parte de la lista SELECT de la tabla resultante final de la consulta que se describe. Si las columnas implícitamente ocultas no forman parte de la tabla resultante de una consulta, la sentencia DESCRIBE OUTPUT que devuelve información sobre dicha consulta no contendrá información sobre columnas implícitamente ocultas.

Ejemplo

En un programa C, ejecute una sentencia DESCRIBE OUTPUT con una SQLDA que no tenga apariciones de SQLVAR. Si SQLD es mayor que cero, utilice el valor para asignar una SQLDA con el número necesario de apariciones de SQLVAR y después ejecute una sentencia DESCRIBE que utilice dicha SQLDA.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char  stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

... /* código para solicitar una consulta al usuario, después generar */
    /* una sentencia select en stmt1_str            */
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

... /* código para establecer SQLN en cero y asignar la SQLDA */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* código para comprobar que SQLD se mayor que cero, para establecer */
    /* SQLN en SQLD, después para volver a asignar SQLDA          */
EXEC SQL  DESCRIBE STMT1_NAME INTO :sqlda;

... /* código para preparar la utilización de SQLDA             */
    /* y asignar almacenamientos intermedios para recibir datos   */
EXEC SQL  OPEN DYN_CURSOR;
```

```
   ... /* bucle para leer filas de la tabla de resultados
*/
   EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :sqlda;
   .
   .
   .
```

DISCONNECT

La sentencia DISCONNECT finaliza una o más conexiones cuando no existe ninguna unidad de trabajo activa (es decir, tras una operación de confirmación o de retrotracción). Si el destino de la sentencia DISCONNECT es una única conexión, la conexión finaliza sólo si la base de datos ha participado en una unidad de trabajo existente, con independencia de si existe o no una unidad de trabajo activa. Por ejemplo, si otras bases de datos han realizado tareas, pero el destino en cuestión no, la desconexión puede tener lugar sin finalizarse la conexión.

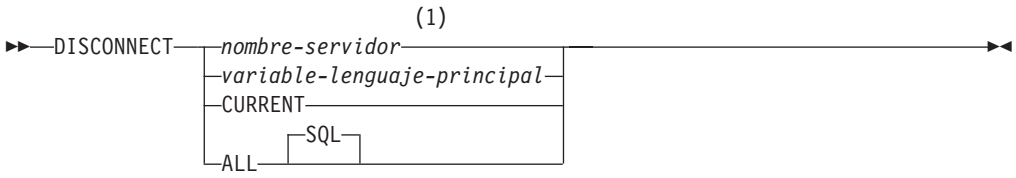
Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Notas:

1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal.

Descripción

nombre-servidor **o** *variable-lenguaje-principal*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lenguaje-principal* 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). Sólo es válido cuando se utiliza DB2 Transaction Manager.

Notas

- Si la sentencia DISCONNECT se realiza satisfactoriamente, se destruye cada conexión.

  Si la sentencia DISCONNECT no es satisfactoria, el estado de la conexión del proceso de aplicación y los estados de sus conexiones no se cambian.
- Si se utiliza DISCONNECT para destruir la conexión actual, la siguiente sentencia de SQL ejecutada debe ser CONNECT o SET CONNECTION.
- La semántica de CONNECT Tipo 1 no excluye la utilización de DISCONNECT. Sin embargo, aunque se puedan utilizar DISCONNECT CURRENT y DISCONNECT ALL, no dan como resultado una operación de confirmación como lo haría la sentencia CONNECT RESET.

  Si se especifica *nombre-servidor* o *variable-lenguaje-principal* en la sentencia DISCONNECT, debe identificar la conexión actual porque CONNECT Tipo 1 sólo da soporte a una conexión cada vez. Generalmente, DISCONNECT 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;
```

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 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

Cuando se descartan objetos que permiten nombres de dos partes, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los elementos siguientes:

- Privilegio DROPIN para el esquema del objeto
- Propietario del objeto, tal como está registrado en la columna OWNER de la vista de catálogo del objeto
- Privilegio CONTROL para el objeto (aplicable sólo a índices, especificaciones de índice, apodos, paquetes, tablas y vistas)
- Propietario del tipo definido por el usuario, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.DATATYPES (aplicable sólo al descartar un método asociado a un tipo definido por el usuario)
- Autorización SYSADM o DBADM

Al descartar una jerarquía de tablas o vistas, los privilegios del ID de autorización de la sentencia deben incluir uno de los privilegios anteriores para cada tabla o vista de la jerarquía.

Al descartar un esquema, el ID de autorización de la sentencia debe mantener la autorización SYSADM o DBADM o ser el propietario del esquema, tal como está registrado en la columna OWNER de la vista de catálogo SYSCAT.SCHEMATA.

Al descartar una agrupación de almacenamientos intermedios, un grupo de particiones de base de datos, una plantilla de histograma o un espacio de tablas, el ID de autorización de la sentencia debe mantener la autoridad SYSADM o SYSCTRL.

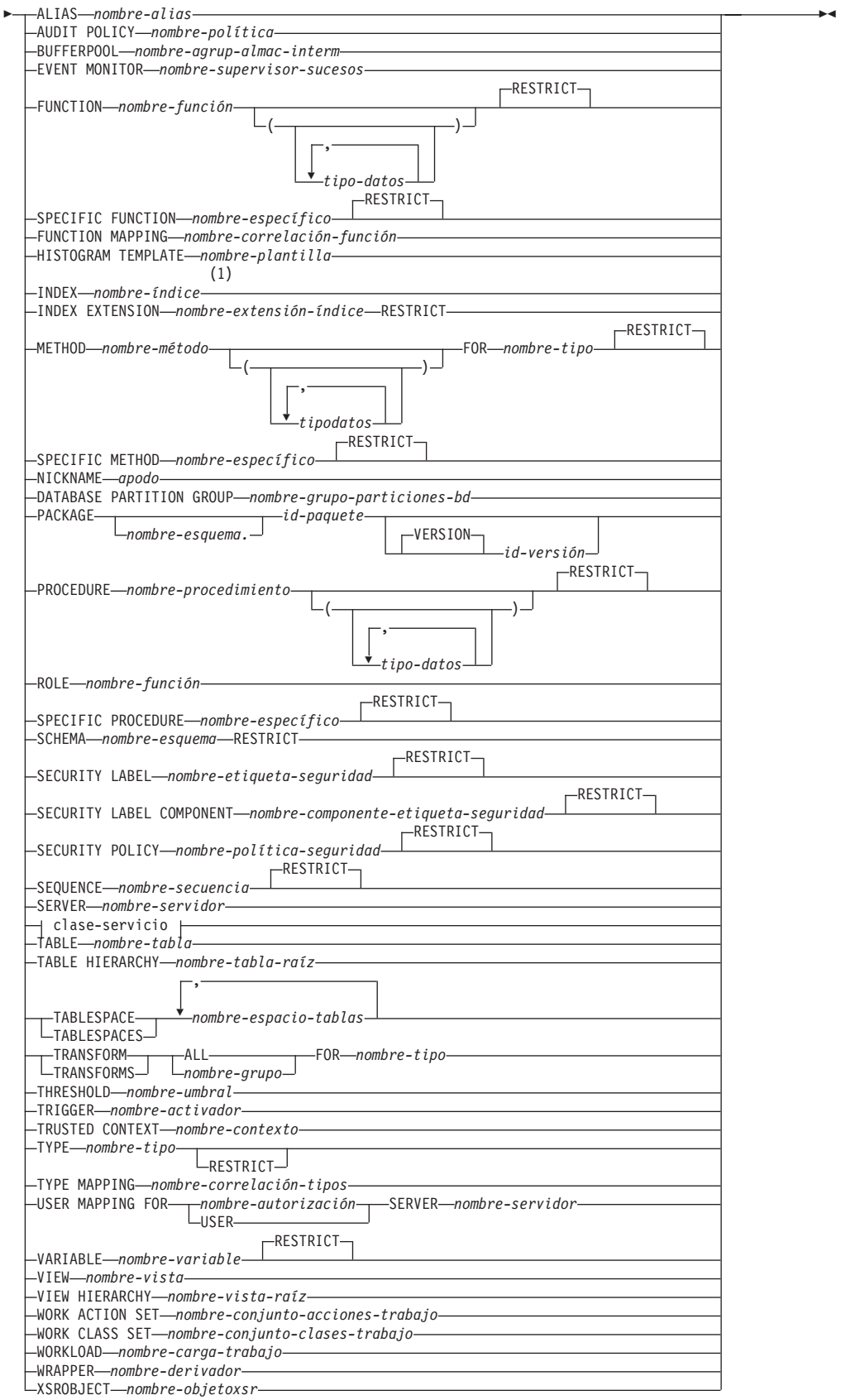
Al descartar una correlación de tipos de datos, un supervisor de sucesos, una correlación de funciones, una definición de servidor, una clase de servicio, un conjunto de acciones de trabajo, un conjunto de clases de trabajo, un umbral, un histograma, o un derivador, el ID de autorización de la sentencia debe mantener la autoridad SYSADM o DBADM.

Al descartar una correlación de usuarios, el ID de autorización de la sentencia debe mantener la autorización SYSADM o DBADM 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.

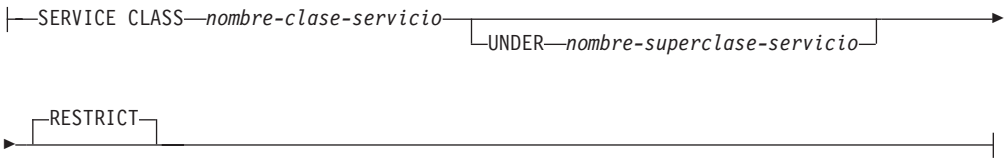
Al descartar una transformación, el ID de autorización de la sentencia debe mantener la autorización SYSADM o DBADM o debe ser el propietario de *nombre-tipo*.

Al descartar una política de comprobación, una función, una etiqueta de seguridad, un componente de etiqueta de seguridad, una política de seguridad o un contexto fiable, el ID de autorización de la sentencia debe mantener la autoridad SECADM.

Sintaxis


ALIAS nombre-alias
AUDIT POLICY nombre-política
BUFFERPOOL nombre-agrup-almac-interm
EVENT MONITOR nombre-supervisor-sucesos
FUNCTION nombre-función (, tipo-datos) RESTRICT
SPECIFIC FUNCTION nombre-específico RESTRICT
FUNCTION MAPPING nombre-correlación-función
HISTOGRAM TEMPLATE nombre-plantilla
(1)
INDEX nombre-índice
INDEX EXTENSION nombre-extensión-índice RESTRICT
METHOD nombre-método (, tipodatos) FOR nombre-tipo RESTRICT
SPECIFIC METHOD nombre-específico RESTRICT
NICKNAME apodo
DATABASE PARTITION GROUP nombre-grupo-particiones-bd
PACKAGE nombre-esquema. id-paquete VERSION id-versión
PROCEDURE nombre-procedimiento (, tipo-datos) RESTRICT
ROLE nombre-función
SPECIFIC PROCEDURE nombre-específico RESTRICT
SCHEMA nombre-esquema RESTRICT
SECURITY LABEL nombre-etiqueta-seguridad RESTRICT
SECURITY LABEL COMPONENT nombre-componente-etiqueta-seguridad RESTRICT
SECURITY POLICY nombre-política-seguridad RESTRICT
SEQUENCE nombre-secuencia RESTRICT
SERVER nombre-servidor
clase-servicio
TABLE nombre-tabla
TABLE HIERARCHY nombre-tabla-raíz
TABLESPACE
TABLESPACES
, nombre-espacio-tablas
TRANSFORM
TRANSFORMS
ALL
nombre-grupo
FOR nombre-tipo
THRESHOLD nombre-umbral
TRIGGER nombre-activador
TRUSTED CONTEXT nombre-contexto
TYPE nombre-tipo RESTRICT
TYPE MAPPING nombre-correlación-tipos
USER MAPPING FOR nombre-autorización
USER
SERVER nombre-servidor
VARIABLE nombre-variable RESTRICT
VIEW nombre-vista
VIEW HIERARCHY nombre-vista-raíz
WORK ACTION SET nombre-conjunto-acciones-trabajo
WORK CLASS SET nombre-conjunto-clases-trabajo
WORKLOAD nombre-carga-trabajo
WRAPPER nombre-derivador
XSROBJECT nombre-objetoxsr

clase-servicio:



Notas:

1 *Nombre-índice* puede ser el nombre de un índice o una especificación de índice.

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.)

AUDIT POLICY *nombre-política*
Identifica la política de comprobación que se va a descartar. El *nombre-política* debe identificar una política de comprobación que exista en el servidor actual (SQLSTATE 42704). La política de comprobación no debe asociarse con ningún objeto de base de datos (SQLSTATE 42893). La política de comprobación especificada se suprime del catálogo.

BUFFERPOOL *nombre-agrup-almac-interm*
Identifica la agrupación de almacenamientos intermedios que se debe eliminar. El *nombre-agrup-almac-interm* debe identificar una agrupación de almacenamientos intermedios que se haya descrito en el catálogo (SQLSTATE 42704). Puede que no haya ningún espacio de tablas asignado a la agrupación de almacenamientos intermedios (SQLSTATE 42893). La agrupación de almacenamientos intermedios IBMDEFAULTBP no se puede 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 está activo, se devuelve un error (SQLSTATE 55034); de lo contrario, el supervisor de sucesos se suprime. Tenga en cuenta que si un supervisor de sucesos se ha activado previamente mediante la sentencia SET EVENT MONITOR STATE y la base de datos se ha desactivado y se ha vuelto a activar, debe utilizar la sentencia SET EVENT MONITOR STATE para desactivar el supervisor de sucesos antes de emitir la sentencia DROP.

Si hay archivos de sucesos en la vía de acceso de destino de un supervisor de sucesos WRITE TO FILE que se está descartando, los archivos de sucesos no se

suprimirán. Sin embargo, si se crea un nuevo supervisor de sucesos que especifique la misma vía de acceso de destino, se suprimirán los archivos de sucesos.

Cuando se 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. No se pueden descartar las funciones implícitamente generadas por la sentencia CREATE TYPE (Diferenciado).

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 sentencias de SQL estático, la opción precompilación/vinculación 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 implícito o especificado, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia específica de la función en el esquema especificado o implícito, se devuelve 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/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto 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 su lugar, puede haber un

conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse cuando se busque 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).

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 fuente.
- 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 existe ninguna función con esta signatura en el esquema implícito o especificado, se devuelve 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 sentencias de SQL estático, la opción precompilación/vinculación 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 especificado o implícito; de lo contrario, se devuelve 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 fuente.
- 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 “Normas” en la página 743 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.

No se pueden descartar las correlaciones de funciones por omisión, pero se pueden inhabilitar mediante la sentencia CREATE FUNCTION MAPPING. Si se descartar una correlación de funciones definida por el usuario que se había creado para alterar temporalmente una correlación de funciones por omisión, se restituye la correlación de funciones por omisión.

Los paquetes que tengan una dependencia de la correlación de funciones eliminada se invalidarán.

HISTOGRAM TEMPLATE *nombre-plantilla*
Identifica la plantilla de histograma que se debe descartar. El *nombre-plantilla* debe identificar una plantilla de histograma que exista en el servidor actual (SQLSTATE 42704). El *nombre-plantilla* no puede ser SYSDEFAULTHISTOGRAM (SQLSTATE 42832). La plantilla de histograma no puede eliminarse si de ella depende una clase de servicio o una acción de trabajo (SQLSTATE 42893). La plantilla de histograma especificada se suprime del catálogo.

INDEX *nombre-índice*
Identifica el índice o especificación de índice que se debe 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 necesite para una clave primaria o restricción de unicidad, para una tabla de consultas materializadas duplicada o una columna XML (SQLSTATE 42917). El índice especificado o la especificación de índice se suprimen.

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-tipos*, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia específica del método para el tipo de datos especificado, se devuelve 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 haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse cuando se busque 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 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 con esta signatura para el tipo de datos especificado, se devuelve 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 fuente.
- 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 devuelve 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 fuente.
- 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-bd*
Identifica el grupo de particiones de base de datos que va a eliminarse. El

parámetro *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos que se haya descrito en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

La eliminación de un grupo de particiones de base de datos elimina todos los espacios de tablas que se han definido en el grupo de particiones de base de datos. Todos los objetos de base de datos que existan con dependencias en las tablas del espacio de tablas (por ejemplo, paquetes, restricciones de referencia, etcétera) se descartan o invalidan (según sea adecuado), y las vistas y los activadores dependientes no estarán operativos.

Los grupos de particiones de base de datos 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 que debe descartarse. La instancia de procedimiento especificada debe ser un procedimiento 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 implícito o especificado, se devuelve 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 sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. Si hay más de una instancia específica del procedimiento en el esquema mencionado o implícito, se devuelve un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento* **(***tipo-datos***,...)**
Proporciona la signatura del método, que identifica de manera exclusiva el procedimiento que se debe eliminar. El algoritmo de selección de procedimientos no se utiliza. Para procedimientos federados, la información de la signatura no está especificada en la sentencia CREATE PROCEDURE, sino que la información está disponible en el catálogo del sistema.

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/vinculación QUALIFIER especifica implícitamente el calificador para los nombres de objeto no calificados.

(tipo-datos,...)
Debe coincidir con los tipos de datos que se han especificado en la sentencia CREATE PROCEDURE en la posición correspondiente, excepto los procedimientos federados, donde el tipo de datos debe coincidir con el almacenado en el catálogo local del parámetro 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 haber un conjunto vacío de paréntesis codificados para indicar que esos atributos deben ignorarse cuando se busque una coincidencia del tipo de datos.

No puede utilizarse FLOAT() (SQLSTATE 42601) puesto que el valor del parámetro indica tipos de datos diferentes (REAL o DOUBLE).

Sin embargo, si la longitud, la precisión o la escala está codificada, el valor debe coincidir exactamente con el especificado en la sentencia CREATE PROCEDURE o, para procedimientos federados, debe coincidir exactamente con el almacenado en el catálogo local para el parámetro correspondiente.

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 evita 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.

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

ROLE *nombre-función*
Identifica la función que debe descartarse. El *nombre-función* debe identificar una función que ya exista en el servidor actual (SQLSTATE 42704). El *nombre-función* no debe identificar una función o una función que contenga *nombre-función*, si la función tiene privilegio EXECUTE sobre una rutina o privilegio USAGE sobre una secuencia y un objeto SQL que no sea un paquete depende de la rutina o secuencia (SQLSTATE 42893). El propietario del objeto SQL es *nombre-autorización* o cualquier usuario que sea miembro de *nombre-autorización*, donde *nombre-autorización* es una función.

Una sentencia A DROP ROLE falla (SQLSTATE 42893) si se da alguna de las condiciones siguientes para la función que debe eliminarse:

- Existe una carga de trabajo de modo que uno de los valores del atributo de conexión SESSION_USER ROLE es *nombre-función*
- Existe un contesto fiable que utiliza *nombre-función*

La función especificada se suprime del catálogo.

SPECIFIC PROCEDURE *nombre-específico*
Identifica el procedimiento en particular que debe descartarse, utilizando el nombre específico que se ha indicado o que se ha tomado por omisión al crear el procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de procedimiento específica en el esquema especificado o implícito; de lo contrario, se devuelve un error (SQLSTATE 42704).

RESTRICT
La palabra clave RESTRICT evita 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 se describe en el catálogo (SQLSTATE 42704). La palabra clave RESTRICT impone la norma de que no puede haber ningún objeto definido en el esquema especificado para que se suprima de la base de datos (SQLSTATE 42893).

SECURITY LABEL *nombre-etiqueta-seguridad*
Identifica la etiqueta de seguridad que se descartará. Se debe calificar el nombre con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704).

RESTRICT
Esta opción, que es el valor por omisión, impide que se descarte la etiqueta de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

- Uno o varios ID de autorización tienen la etiqueta de seguridad para acceso de lectura
- Uno o varios Id de autorización tienen la etiqueta de seguridad para acceso de grabación
- La etiqueta de seguridad se utiliza para proteger una o varias columnas

SECURITY LABEL COMPONENT *nombre-componente-etiqueta-seguridad*
Identifica el componente de la etiqueta de seguridad que se descartará. El *nombre-comp-etiqueta-seg* debe identificar un componente de la etiqueta de seguridad que se describe en el catálogo (SQLSTATE 42704).

RESTRICT
Esta opción, que es el valor por omisión, impide que se descarte el componente de la etiqueta de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

- Una o varias políticas de seguridad que incluyen el componente de la etiqueta de seguridad están definidos

SECURITY POLICY *nombre-política-seguridad*
Identifica la política de seguridad que se descartará. El *nombre-política-seguridad* debe identificar una política de seguridad que exista en el servidor actual (SQLSTATE 42704).

RESTRICT
Esta opción, que es el valor por omisión, impide que se descarte la política de seguridad si existe alguna de las dependencias siguientes (SQLSTATE 42893):

- Una o varias tablas están asociadas con esta política de seguridad
- Uno o varios ID de autorización tienen una exención con una de las normas de esta política de seguridad
- Una o varias etiquetas de seguridad están definidas para esta política de seguridad

SEQUENCE *nombre-secuencia*
Identifica la secuencia en particular que se debe 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 implícita o explícitamente, se devuelve un error (SQLSTATE 42704).

RESTRICT
Esta opción, que es el valor por omisión, impide que se descarte la secuencia de seguridad 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. Todos los procedimientos federados que dependen de la definición del servidor también se descartan.

SERVICE CLASS *nombre-clase-servicio*
Identifica la clase de servicio que debe descartarse. El *nombre-clase-servicio* debe identificar una clase de servicio que esté descrita en el catálogo (SQLSTATE 42704). Para descartar una subclase de servicio, el *nombre-superclase-servicio* debe especificarse utilizando la cláusula UNDER.

UNDER *nombre-superclase-servicio*
Especifica la superclase de servicio de la subclase de servicio al descartar una subclase de servicio. El *nombre-superclase-servicio* debe identificar una superclase de servicio que esté descrita en el catálogo (SQLSTATE 42704).

RESTRICT
Esta palabra clave aplica la norma que establece que la clase de servicio no se descartará si existe alguna de las dependencias siguientes:

- La clase de servicio es una superclase de servicio y existe una subclase de servicio definida por el usuario bajo la clase de servicio (SQLSTATE 5U031). Primero debe descartarse la subclase de servicio.
- La clase de servicio es una superclase de servicio y existe una correlación de conjuntos de acciones de trabajo con la clase de servicio (SQLSTATE 5U031). Primero debe descartarse el conjunto de acciones de trabajo.
- La clase de servicio es una subclase de servicio y existe una correlación de acciones de trabajo con la clase de servicio (SQLSTATE 5U031). Primero debe descartarse la acción de trabajo.
- La clase de servicio tiene una correlación de cargas de trabajo (SQLSTATE 5U031). Primero debe eliminarse la correlación. Elimine la correlación de cargas de trabajo descartando la carga de trabajo o modificando la carga de trabajo para que no se correlacione con la clase de servicio.
- La clase de servicio tiene un umbral asociado (SQLSTATE 5U031). Primero debe descartarse el umbral.
- La clase de servicio debe inhabilitarse (SQLSTATE 5U031).

RESTRICT es el comportamiento por omisión.

TABLE *nombre-tabla*
Identifica la tabla base o la tabla temporal declarada que se debe descartar. El *nombre-tabla* debe identificar una tabla que esté descrita en el catálogo, si se trata de una tabla temporal declarada, el *nombre-tabla* debe estar calificado por

el nombre de esquema SESSION y existir en la aplicación (SQLSTATE 42704). Las subtablas de una tabla con tipo dependen de sus supertablas. Deben 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.

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.

Una tabla desenlazada recientemente inicialmente no es accesible. Esto impide que se lea, se modifique o se descarte la tabla hasta que la sentencia SET INTEGRITY se pueda ejecutar para actualizar de forma incremental las MQT o para completar el proceso de restricciones de clave foránea. Una vez que se ejecute la sentencia SET INTEGRITY en todas las tablas dependientes, la tabla es completamente accesible, su atributo desenlazado se restaura y se puede descartar.

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.

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-tablas*
Identifica los espacios de tablas que se descartarán; el *nombre-espacio-tablas* debe identificar un espacio de tablas que se describe en el catálogo (SQLSTATE 42704). Este nombre consta de una sola parte.

Los espacios de tablas no se descartarán (SQLSTATE 55024) si existe alguna tabla que almacene al menos una de sus partes en un espacio de tablas que se está descartando y tiene una o varias partes en otro espacio de tablas que no se está descartando (estas tablas deben descartarse primero) o si alguna tabla que reside en el espacio de tablas tiene el atributo RESTRICT ON DROP.

Los objetos cuyos nombres contienen el prefijo 'SYS' son objetos definidos por el sistema y, salvo en el caso de los espacios de tablas SYSTOOLSPACE y SYSTOOLSTMPSPACE, no se pueden descartar (SQLSTATE 42832).

Un espacio de tablas temporal del sistema (SYSTEM TEMPORARY) no se puede 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 descartado una tabla temporal declarada, se considerará al espacio de tablas USER TEMPORARY en uso hasta que se haya confirmado la unidad de trabajo que contiene la sentencia DROP TABLE.

Al descartar un espacio de tablas se descartan todos los objetos definidos en el espacio de tablas. Todos los objetos de base de datos existentes con dependencias en el espacio de tablas como, por ejemplo, paquetes, restricciones de referencia, etc. se eliminan o invalidan (lo que sea adecuado) y las vistas y activadores dependientes pasan a estar no operativos.

Los contenedores creados por un usuario no se suprimen. Los directorios de la vía de acceso del nombre del contenedor creados por el gestor de bases de datos durante la ejecución de CREATE TABLESPACE se suprimen. Se suprimen todos los contenedores que están por debajo del directorio de bases de datos. Una vez confirmada la sentencia DROP TABLESPACE, los contenedores del archivo DMS o los contenedores SMS del espacio de tablas especificado se suprimen, si es posible. Si no es posible suprimirlos (por ejemplo, porque otro agente los tiene abiertos), los archivos se truncan a una longitud de cero. Una vez terminados todos los contenedores, o cuando se emite el mandato DEACTIVATE DATABASE, estos archivos de longitud cero se suprimen.

THRESHOLD *nombre-umbral*
Identifica el umbral que debe descartarse. El *nombre-umbral* debe identificar un umbral que exista en el servidor actual (SQLSTATE 42704). Este nombre consta de una sola parte. Los umbrales con una cola, por ejemplo

TOTALSCPARTITIONCONNECTIONS y
CONCURRENTDBCOORDACTIVITIES, deben inhabilitarse antes de que se puedan descartar (SQLSTATE 5U025). El umbral especificado se suprime del catálogo.

TRIGGER *nombre-activador*
Identifica el activador que se debe 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.

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 existen transformaciones definidas para *nombre-tipo*, se devuelve un error (SQLSTATE 42740).

DROP TRANSFORM es lo opuesto de CREATE TRANSFORM. Hace que las funciones de transformación asociadas a determinados grupos, para un tipo de datos 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 sistema 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.

TRUSTED CONTEXT *nombre-contexto*
Identifica el contexto fiable que debe descartarse. El *nombre-contexto* debe identificar un contexto fiable que exista en el servidor actual (SQLSTATE 42704). Si el contexto fiable se descarta mientras están activas las conexiones fiables para este contexto, dichas conexiones seguirán siendo fiables hasta que terminen o hasta que se produzca el siguiente intento de volver a utilizarlas. Si se intenta cambiar el usuario de estas conexiones fiables, se devuelve un error (SQLSTATE 42517). El contexto fiable especificado se suprime del catálogo.

TYPE *nombre-tipo*
Identifica el tipo definido por el usuario que se debe eliminar. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción precompilación/vinculación QUALIFIER especifica de forma implícita el calificador para los nombres de objeto 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.

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 es un tipo de parámetro o se utiliza en el cuerpo de un procedimiento de SQL.
- El tipo, o una referencia al tipo, se utiliza en el cuerpo de una función o método SQL, pero no es un tipo de parámetro ni un tipo de valor de retorno.
- El tipo se utiliza en una restricción de comprobación, un activador, una definición de vista o en una extensión de índice.

Si no se especifica RESTRICT, el comportamiento es el mismo que RESTRICT, excepto por las funciones y los métodos que utilizan el tipo.

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 descartan 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 descartan objetos adicionales.

VARIABLE *nombre-variable*
Identifica la variable global que se debe descartar. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

RESTRICT
Esta palabra clave aplica la norma que establece que no puede descartarse

la variable global si se le hace referencia en una función, método, activador o vista (SQLSTATE 42893). RESTRICT es el comportamiento por omisión.

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 vistas 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.

WORK ACTION SET *nombre-conjunto-acciones-trabajo*
Identifica el conjunto de acciones de trabajo que se debe descartar. El *nombre-conjunto-acciones-trabajo* debe identificar un conjunto de acciones de trabajo que exista en el servidor actual (SQLSTATE 42704). También se descartan todas las acciones de trabajo contenidas en *nombre-conjunto-acciones-trabajo*.

WORK CLASS SET *nombre-conjunto-clases-trabajo*
Identifica la clase de trabajo que se debe descartar. El *nombre-conjunto-clases-trabajo* debe identificar un conjunto de clases de trabajo que exista en el servidor actual (SQLSTATE 42704). También se descartan todas las clases de trabajo contenidas en *nombre-conjunto-clases-trabajo*.

WORKLOAD *nombre-carga-trabajo*
Identifica la carga de trabajo que se debe descartar. Este nombre consta de una sola parte. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). No se puede descartar SYSDEFAULTUSERWORKLOAD o SYSDEFAULTADMWORKLOAD (SQLSTATE 42832). Para poder descartarse, una carga de trabajo debe estar deshabilitada y no debe tener apariciones de carga de trabajo activas asociadas a ella (SQLSTATE 5U023). La carga de trabajo especificada se suprime del catálogo.

WRAPPER *nombre-derivador*
Identifica el derivador que se debe eliminar. El *nombre-derivador* debe identificar un derivador que esté descrito en el catálogo (SQLSTATE 42704). Se suprime el derivador.

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 derivador. 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. Todos los procedimientos federados que dependen de la definición del servidor también se descartan.

XSROBJECT *nombre-objetoxsr*
Identifica el objeto XSR que se descartará. El *nombre-objetoxsr* debe identificar un objeto XSR que se describe en el catálogo (SQLSTATE 42704).

Se descartan las restricciones de comprobación que hacen referencia al objeto XSR. Se marcan como inoperativos todos los activadores y vistas que hacen referencia al objeto XSR. Los paquetes dependen de un objeto XSR descartado se invalidan.

Normas

Dependencias: La Tabla 28 en la página 744 muestra las dependencias que los objetos tienen 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 la Tabla 28 en la página 744 porque darían como resultado columnas o filas en blanco:

- Las sentencias EVENT MONITOR, PACKAGE, PROCEDURE, SCHEMA, TYPE MAPPING y USER MAPPING DROP no tienen dependencias de objeto.
- Los tipos de objeto de alias, agrupación de almacenamientos intermedios, clave de distribución, privilegio y procedimiento no dependen de la sentencia DROP.

- Una sentencia DROP SERVER, DROP FUNCTION MAPPING, o DROP TYPE MAPPING en una unidad de trabajo dada (UOW) no puede procesarse bajo ninguna de las condiciones siguientes:
  - La sentencia hace referencia a una sola fuente de datos y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o vista dentro de esta fuente de datos (SQLSTATE 55006).
  - La sentencia hace referencia a una categoría de una fuente de datos (por ejemplo, todas las fuentes de datos para un tipo específico y versión) y la UOW ya incluye una sentencia SELECT que hace referencia a un apodo para una tabla o vista dentro de estas fuentes de datos (SQLSTATE 55006).

Tabla 28. Dependencias

	Tipo de objeto																								
Sentencia	**CONSTRAINT**	**FUNCTION**	**FUNCTION MAPPING**	**GLOBAL VARIABLE**	**INDEX**	**INDEX EXTENSION**	**METHOD**	**NICKNAME**	**DB PARTITION GROUP**	**PACKAGE**[31]	**SERVER**	**SERVICE CLASS**	**TABLE**	**TABLESPACE**	**THRESHOLD**	**TRIGGER**	**TYPE**	**TYPE MAPPING**	**USER MAPPING**	**VIEW**	**WORK ACTION**	**WORK ACTION SET**	**WORKLOAD**	**XSROBJECT**
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 ALTER COLUMN	-	A	-	A	-	-	-	-	-	A	-	-	-	-	-	A	-	-	-	A	-	-	-	X[34]
ALTER TABLE DROP COLUMN	C	C	-	C	C	-	-	-	-	-	-	-	-	-	-	C	-	-	-	C	-	-	-	X[34]

Tabla 28. Dependencias (continuación)

	Tipo de objeto																							
Sentencia	**CONSTRAINT**	**FUNCTION**	**FUNCTION MAPPING**	**GLOBAL VARIABLE**	**INDEX**	**INDEX EXTENSION**	**METHOD**	**NICKNAME**	**DBPARTITIONGROUP**	**PACKAGE**[31]	**SERVER**	**SERVICE CLASS**	**TABLE**	**TABLESPACE**	**THRESHOLD**	**TRIGGER**	**TYPE**	**TYPE MAPPING**	**USER MAPPING**	**VIEW**	**WORK ACTION**	**WORK ACTION SET**	**WORKLOAD**	**XSROBJECT**
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	-	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	R[7]	-	-	X	-	-	R	-	-	R	-	-	-	R	-	-	-	-
DROP FUNCTION MAPPING	-	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP INDEX	R	-	-	-	-	-	-	-	-	A	-	-	-	-	-	-	-	-	-	R[17]	-	-	-	-
DROP INDEX EXTENSION	-	R	-	R	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DROP METHOD	R	R[7]	R	R	-	R	R	-	-X/A[30]	-	-	R	-	-	R	-	-	-	R	-	-	-	-	-

Tabla 28. Dependencias (continuación)

	Tipo de objeto																								
Sentencia	**CONSTRAINT**	**FUNCTION**	**FUNCTION MAPPING**	**GLOBAL VARIABLE**	**INDEX**	**INDEX EXTENSION**	**METHOD**	**NICKNAME**	**DB PARTITION GROUP**	**PACKAGE**[31]	**SERVER**	**SERVICE CLASS**	**TABLE**	**TABLESPACE**	**THRESHOLD**	**TRIGGER**	**TYPE**	**TYPE MAPPING**	**USER MAPPING**	**VIEW**	**WORK ACTION**	**WORK ACTION SET**	**WORKLOAD**	**XSROBJECT**
DROP NICKNAME	-	R	-	R	C	-	R	-	-	A	-	-	C[11]	-	-	-	-	-	-	X[16]	-	-	-	-
DROP PROCEDURE	-	R[7]	-	R	-	-	R[7]	-	-	A	-	-	-	-	-	R	-	-	-	-	-	-	-	-
DROP SEQUENCE	-	R	-	-	-	-	R	-	-	A	-	-	-	-	-	R	-	-	-	-	-	-	-	-
DROP SERVER	-	C[21]	C[19]	-	-	-	-	C	-	A	-	-	-	-	-	-	-	C[19]	C	-	-	-	-	-
DROP SERVICE CLASS	-	-	-	-	-	-	-	-	-	-	-	R[35]	-	-	R[35]	-	-	-	-	-	R[35]	-	R[35]	-
DROP TABLE[32]	C	R	-	R	C	-	-	-	-	A[9]	-	-	RC[11]	-	-	X[16]	-	-	-	X[16]	-	-	-	X[34]
DROP TABLE HIERARCHY	C	R	-	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	-	R	-	-	-	A[12]	-	-	R[18]	-	-	R[13]	R[4]	-	-	R[14]	-	-	-	-
DROP VARIABLE	-	-	R	R	-	-	R	-	-	A	-	-	-	-	-	R	-	-	-	R	-	-	-	-
DROP VIEW	-	R	-	R	-	-	-	-	-	A[2]	-	-	-	-	-	X[16]	-	-	-	X[15]	-	-	-	-
DROP VIEW HIERARCHY	-	R	-	R	-	-	-	-	-	A[2]	-	-	-	-	-	X[16]	-	-	-	X[16]	-	-	-	-
DROP WORK CLASS SET	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R[36]	-	-
DROP WRAPPER	-	-	C	-	-	-	-	-	-	-	C	-	-	-	-	-	-	C	-	-	-	-	-	-
DROP XSROBJECT	C	-	-	-	-	-	-	-	-	A	-	-	-	-	-	X	-	-	-	X	-	-	-	-
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 distribución.

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] Descartar un tipo definido por el usuario hace que se descarten en cascada las funciones y los métodos que utilizan el tipo como parámetro, como tipo resultante o en el cuerpo del método o de la función. Si el tipo definido por el usuario es un tipo estructurado, también se descartan todos los métodos asociados con el tipo. El descarte de estas funciones y métodos no se ve impedido por el hecho de que el tipo y la función o método sean dependientes entre sí.

[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 fragmenta espacios de tablas (índices, columnas largas o particiones de datos en distintos espacios de tablas) dichos espacios de tablas no se encuentran en la lista que se está descartando, los espacios de tablas no se pueden descartar mientras la tabla siga existiendo.

[7] Una función puede depender de otra función específica si la función dependiente nombra la función de base en una cláusula SOURCE. Una función o método puede también depender de otra función o método determinados si la rutina dependiente está escrita en SQL y utiliza la rutina base en su cuerpo. Un método externo o una función externa con un parámetro de tipo estructurado o tipo de retorno dependerá también de una o más funciones de transformación.

[8] Sólo la pérdida del privilegio SELECT dará lugar a que se 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 especificado.

20 Si la clave de distribución se define en una tabla en un grupo de particiones de base de datos de varias particiones, la clave de distribución es necesaria.

21 Si una función de tabla dependiente OLE DB tiene "R" objetos dependientes (véase DROP FUNCTION), el servidor no se puede 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.

[34] Un XSROBJECT puede ser inoperativo para la descomposición como consecuencia de los cambios realizados en una tabla asociada con el esquema XML para descomposición. Los cambios que pueden afectar a la descomposición son los siguientes: descartar la tabla o una columna de la tabla, o cambiar una columna de la tabla. El estado de descomposición del esquema XML se puede restaurar mediante la emisión de una sentencia ALTER XSROBJECT para habilitar o inhabilitar la descomposición del esquema XML.

[35]

- No se puede descartar una clase de servicio si está correlacionada con un umbral (SQLSTATE 5U031).
- No se puede descartar una clase de servicio si está correlacionada con una carga de trabajo (SQLSTATE 5U031).
- No se puede descartar una superclase de servicio hasta que se hayan descartado todas sus subclases de servicio definidas por el usuario (SQLSTATE 5U031).
- No se puede descartar una superclase de servicio si está correlacionada con un conjunto de acciones de trabajo (SQLSTATE 5U031).
- No se puede descartar una subclase de servicio si está correlacionada con una acción de trabajo (SQLSTATE 5U031).

[36] No se puede descartar un conjunto de clases de trabajo hasta que se haya descartado el conjunto de acciones de trabajo definido en él.

Notas

- 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.

- 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, no es posible descartar la propiedad de transformación de la función, de lo contrario, los paquetes se marcarán como inoperativos, por el mero hecho que contienen invocaciones explícitas en una expresión SQL.
- Las secuencias creadas por el sistema para las columnas IDENTITY no pueden eliminarse utilizando la sentencia DROP SEQUENCE.
- Cuando se descarta una secuencia, también se descartan todos los privilegios de ésta y los paquetes que hacen referencia a la secuencia se invalidan.
- Para apodos relacionales, la sentencia DROP NICKNAME de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las siguientes condiciones (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Para apodos no relacionales, la sentencia DROP NICKNAME dentro de una unidad de trabajo (UOW) determinada no se puede procesar bajo ninguna de las condiciones siguientes (SQLSTATE 55007):
  - Un apodo al que se hace referencia en esta sentencia tiene un cursor abierto en la misma UOW
  - Una sentencia SELECT de la misma UOW ya hace referencia a un apodo al que se hace referencia en esta sentencia
  - Ya se ha emitido una sentencia INSERT, DELETE o UPDATE en la misma UOW para el apodo al que se hace referencia en esta sentencia
- Una sentencia DROP SERVER (SQLSTATE 55006), o las sentencias DROP FUNCTION MAPPING o DROP TYPE MAPPING (SQLSTATE 55007) de una unidad de trabajo (UOW) determinada no pueden procesarse en bajo ninguna de las condiciones siguientes:

  - La sentencia hace referencia a una única fuente de datos y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de esa fuente de datos
    - Un cursor abierto en un apodo para una tabla o vista de esa fuente de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de esta fuente de datos
  - La sentencia hace referencia a una categoría de fuentes de datos (por ejemplo, todas las fuentes de datos de un tipo y versión específicos) y la UOW ya incluye uno de los elementos siguientes:
    - Una sentencia SELECT que hace referencia a un apodo para una tabla o vista de una de esas fuentes de datos
    - Un cursor abierto en un apodo para una tabla o vista de una de esas fuentes de datos
    - Una sentencia INSERT, DELETE o UPDATE que se ha emitido para un apodo de una tabla o vista de una de esas fuentes de datos
- La sentencia DROP WORKLOAD no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.
- Una aplicación sólo puede emitir una de estas sentencias cada vez y sólo se permite una de estas sentencias dentro de una unidad de trabajo. Cada sentencia debe ir seguida de una sentencia COMMIT o ROLLBACK antes de que pueda emitirse otra de estas sentencias (SQLSTATE 5U021).
  - CREATE HISTOGRAM TEMPLATE, ALTER HISTOGRAM TEMPLATE o DROP (HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS, ALTER SERVICE CLASS o DROP (SERVICE CLASS)
  - CREATE THRESHOLD, ALTER THRESHOLD o DROP (THRESHOLD)
  - CREATE WORK ACTION, ALTER WORK ACTION o DROP (WORK ACTION)
  - CREATE WORK CLASS, ALTER WORK CLASS o DROP (WORK CLASS)
  - CREATE WORKLOAD, ALTER WORKLOAD o DROP (WORKLOAD)
  - GRANT (privilegios de carga de trabajo) o REVOKE (privilegios de carga de trabajo)
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP
    - DISTINCT TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
    - DATA TYPE *nombre-tipo* puede especificarse en vez de TYPE *nombre-tipo*
  - Para mantener la compatibilidad con DB2 para z/OS:
    - SYNONYM puede especificarse en lugar de ALIAS
    - PROGRAM puede especificarse en lugar de PACKAGE

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 tablas (ACCOUNTING y PLANS) del grupo de particiones de base de datos.

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP DATABASE PARTITION GROUP BUSINESS_OPS
```

Ejemplo 6: Pellow desea 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 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. Descartar 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
```

Ejemplo 22: Descarte el componente de la etiqueta de seguridad LEVEL.

```
DROP SECURITY LABEL COMPONENT LEVEL
```

Ejemplo 23: Descarte la etiqueta de seguridad EMPLOYEESECLABEL de la política de seguridad DATA_ACCESS.

```
DROP SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
```

Ejemplo 24: Descarte la política de seguridad DATA_ACCESS.

```
DROP SECURITY POLICY DATA_ACCESS
```

Ejemplo 25: Descarte el componente de la etiqueta de seguridad GROUPS.

```
DROP SECURITY LABEL COMPONENT GROUPS
```

Ejemplo 26: Descarte el esquema XML EMPLOYEE que se encuentra en el esquema SQL HR.

```
DROP XSROBJECT HR.EMPLOYEE
```

Ejemplo 27: descarte la subclase de servicio DOGSALES de la superclase de servicio PETSALES.

```
DROP SERVICE CLASS DOGSALES UNDER PETSALES
```

Ejemplo 28: descarte la superclase de servicio PETSALES, que no tiene subclases de servicio definidas por el usuario. La subclase por omisión para la clase de servicio PETSALES se descarta automáticamente.

```
DROP SERVICE CLASS PETSALES
```

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 se debe especificar 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 sistema 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 de lenguaje principal a las que se hace referencia en las sentencias de SQL se deben declarar en una sección de declaración de variable de lenguaje principal en todos los lenguajes de sistema principal, distintos de REXX. Además, la declaración de cada variable debe preceder a la primera referencia a la variable.

Las variables declaradas fuera de una sección de declaración no deben tener el mismo nombre que las variables declaradas dentro de una sección de declaración.

EXECUTE 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 parámetro.

Invocación

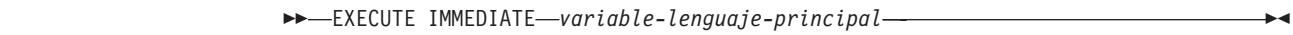
Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

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-lenguaje-principal
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 2.097.152 bytes. Tenga en cuenta que un CLOB(2097152) puede contener una sentencia de tamaño máximo, pero un VARCHAR no puede. El valor de la variable del lenguaje principal identificada se denomina serie de sentencia.

La serie de 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 parámetro ni referencias a variables del lenguaje principal, y no debe empezar por EXEC SQL. No debe contener un terminador de sentencia, a excepción de las sentencias 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 que se especifica en una sentencia CALL no debe tener ningún parámetro OUT ni 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;
```

EXECUTE

La sentencia EXECUTE ejecuta una sentencia de SQL preparada.

Invocación

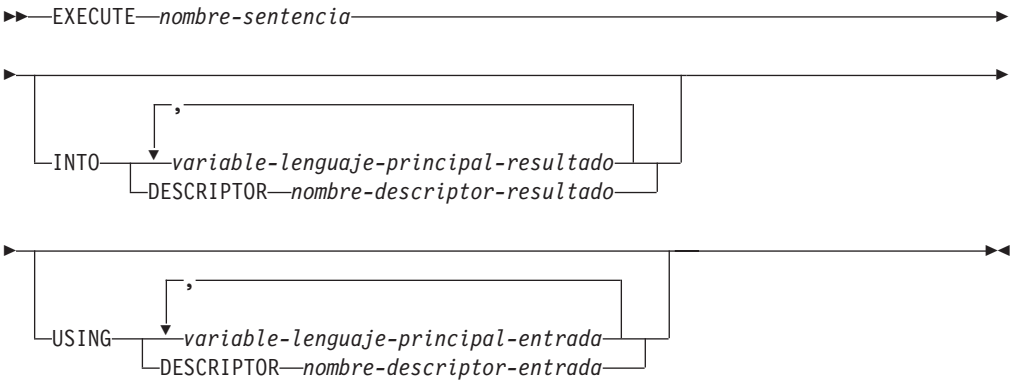
Esta sentencia sólo puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

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 podría verse afectado por la opción de enlace DYNAMICRULES.

Para las sentencias en las que la comprobación de la autorización se lleva a cabo en el momento de preparación de la sentencia (DML), no se requiere autorización para utilizar la 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.

En el caso de una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos OUT e INOUT para el procedimiento son marcadores de parámetro de salida. Si en la sentencia aparece algún marcador de parámetro de salida, debe especificarse la cláusula INTO (SQLSTATE 07007).

variable-lenguaje-principal-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 apariciones 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.

En el caso de una sentencia CALL dinámica, los marcadores de parámetro que aparecen en los argumentos IN e INOUT para el procedimiento son marcadores de parámetro de entrada. Para todas las demás sentencias dinámicas, todos los marcadores de parámetro son marcadores de parámetro de entrada. Si en la sentencia aparece algún marcador de parámetro de salida, debe especificarse la cláusula USING (SQLSTATE 07004).

variable-lenguaje-principal-entrada, **...**
Identifica una variable del lenguaje principal que se declara en el programa de acuerdo a las normas para la declaración de variables del lenguaje principal. El número de variables debe ser el mismo que el número de marcadores de parámetro de entrada de la sentencia preparada. La variable número *n* corresponde al marcador de parámetro número *n* de la sentencia preparada. Las variables de localizador y las variables de referencia a archivo, cuando deben utilizarse, pueden proporcionarse como la fuente de los valores de los marcadores de parámetro.

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 apariciones 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 aparición SQLVAR.

Si deben incluirse datos de entrada de tipo de datos estructurados o LOB, deberán existir dos entradas de SQLVAR para cada marcador de parámetro.

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

Notas

- Antes de ejecutarse la sentencia preparada, cada marcador de parámetro de entrada se sustituye eficazmente por el valor de su 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 indica una variable de lenguaje principal entrante que se corresponde con el marcador de parámetro P. El valor de V se asigna la variable de destino para P de acuerdo con las normas para asignar un valor a una columna. Por lo tanto:

  – V debe ser compatible con el destino.
  – Si V es una serie, su longitud no debe ser mayor que el atributo de longitud del destino.
  – Si V es un número, el valor absoluto de su parte correspondiente al entero no debe ser mayor que el valor absoluto máximo de la parte correspondiente a los enteros del destino.
  – Si los atributos de V no son idénticos a los atributos del destino, el valor se convierte para ajustarse a los atributos del destino.

  Cuando se ejecuta la sentencia preparada, el valor que se utiliza el 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 V con 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.

  Supongamos que V es una variable del lenguaje principal de salida que corresponde al marcador de parámetro P, el cual se utiliza para el argumento A

de un procedimiento. El valor de A se asigna a V de acuerdo con las normas para la recuperación de un valor de una columna. Por lo tanto:

- V debe ser compatible con A.
- Si V es una serie de caracteres, su longitud no debe ser menor que la longitud de A o el valor de A se truncará.
- Si V es un número, el valor absoluto máximo de su parte integrante no debe ser menor que el valor absoluto de la parte integrante de A.
- Si los atributos de V no son idénticos a los atributos de A, el valor de A se convierte para ajustarse a los atributos de V.

- ***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 parámetro. Las variables del lenguaje principal `h1 - h4` corresponden al formato de TDEPT.

```
strcpy (s,"INSERT INTO TDEPT VALUES(?,?,?,?)");
EXEC SQL PREPARE DEPT_INSERT FROM :s;
  .
  .
(Compruebe la ejecución satisfactoria y ponga valores en :h1, :h2, :h3, :h4)
  .
  .
EXEC SQL EXECUTE DEPT_INSERT USING :h1, :h2,
:h3, :h4;
```

Ejemplo 2: La sentencia EXECUTE utiliza una SQLDA.

```
EXECUTE S3 USING DESCRIPTOR :sqlda3
```

Ejemplo 3: Dado un procedimiento para otorgar una bonificación a un empleado:

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
                             IN DEPTNO INTEGER,
                             OUT CHEQUE INTEGER,
                             INOUT BONUS DEC(6,0))
...
```

Llame dinámicamente al procedimiento desde una aplicación en C. El procedimiento toma las siguientes variables del lenguaje principal como entrada:
- *employee*, el número de ID del empleado
- *dept*, el número del departamento
- *bonus*, la bonificación deseada para el empleado

El procedimiento devuelve los valores siguientes a las variables del lenguaje principal:
- *cheque_no*, el número de ID del cheque
- *bonus*, el importe real de la bonificación (tras realizarse cualquier ajuste necesario)

```
strcpy (s, "CALL GIVE_BONUS(?, ?, ?, ?)");
EXEC SQL PREPARE DO_BONUS FROM :s;
  .
  .
/* Comprobar la ejecución satisfactoria y colocar valores en
```

```
   :employee, :dept y :bonus */
  .
  .
EXEC SQL EXECUTE DO_BONUS INTO :cheque_no, :bonus
                          USING :employee, :dept, :bonus;
  .
  .
/* Comprobar la ejecución satisfactoria y procesar los
   valores devueltos en :cheque_no y :bonus */
```

EXPLAIN

La sentencia EXPLAIN captura información sobre el plan de acceso elegido para la sentencia explicable proporcionada y coloca esta información en las tablas de explicación.

Una *sentencia explicable* puede ser una sentencia XQuery válida o una de las siguientes sentencias de SQL: CALL, SQL compuesto (dinámico), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES o VALUES INTO.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

La sentencia que se debe explicar no se ejecuta.

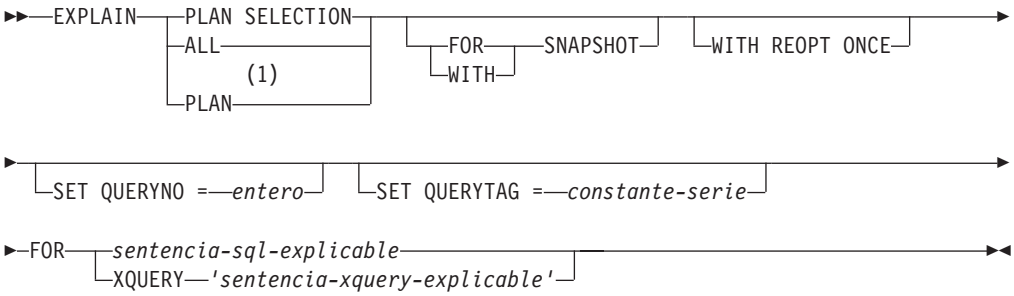
Autorización

Las normas de autorización son las definidas en la sentencia de consulta especificada en la sentencia EXPLAIN. Por ejemplo, si se utiliza una sentencia DELETE como la sentencia explicable, las normas de autorización para la sentencia DELETE se aplican cuando se explica la sentencia DELETE.

Las normas de autorización para sentencias EXPLAIN estáticas son las que se aplican a versiones estáticas de la sentencia pasada como la sentencia explicable. Las sentencias EXPLAIN preparadas dinámicamente utilizan las normas de autorización de la preparación dinámica de la sentencia pasada como la sentencia explicable.

El ID de autorización actual debe tener el privilegio INSERT en las tablas de explicación.

Sintaxis



Notas:

1 Sólo se da soporte a la opción PLAN para la tolerancia de la sintaxis de sentencias EXPLAIN existentes de DB2 para z/OS. No hay ninguna tabla PLAN. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

Descripción

PLAN SELECTION
Indica que la información de la fase de selección del plan de la compilación de consultas debe insertarse en las tablas de explicación.

ALL
La especificación de ALL es equivalente a especificar PLAN SELECTION.

PLAN
La opción PLAN proporciona la tolerancia de la sintaxis para las aplicaciones de bases de datos existentes de otros sistemas. La especificación de PLAN es equivalente a especificar PLAN SELECTION.

FOR SNAPSHOT
Esta cláusula indica que sólo debe tomarse una instantánea de explicación y colocarla en la columna SNAPSHOT de la tabla EXPLAIN_STATEMENT. No se captura ninguna otra información distinta de la que se encuentra en las tablas EXPLAIN_INSTANCE y EXPLAIN_STATEMENT.

La información de instantánea de explicación está pensada para ser utilizada con Visual Explain.

WITH SNAPSHOT
Esta cláusula indica que, además de la información de explicación normal, se debe tomar una instantánea de explicación.

El comportamiento por omisión de la sentencia EXPLAIN es reunir únicamente información de explicación normal y no la instantánea de explicación.

La información de instantánea de explicación está pensada para ser utilizada con Visual Explain.

por omisión (no se especifican FOR SNAPSHOT ni WITH SNAPSHOT)
Coloca la información de explicación en las tablas de explicación. No se toma ninguna instantánea para utilizarla con Visual Explain.

WITH REOPT ONCE
Esta cláusula indica que la *sentencia explicable* especificada se debe volver a optimizar utilizando los valores para variables de lenguaje principal, marcadores de parámetro, registros especiales o variables globales utilizados anteriormente para volver a optimizar esta sentencia con REOPT ONCE. Las tablas de explicación se llenarán con los datos del nuevo plan de acceso. Si el usuario tiene autorización DBADM, o la variable de registro de base de datos DB2_VIEW_REOPT_VALUES se ha establecido en YES, la tabla EXPLAIN_PREDICATE también se rellenará con los valores si se han utilizado para reoptimizar la sentencia.

SET QUERYNO = *entero*
Asocia el *entero*, a través de la columna QUERYNO de la tabla EXPLAIN_STATEMENT, con la *sentencia explicable*. El valor entero suministrado debe ser un valor positivo.

Si no se especifica esta cláusula para una sentencia EXPLAIN dinámica, se asigna un valor por omisión de uno (1). Para una sentencia EXPLAIN estática, el valor por omisión asignado es el número de sentencia asignado por el precompilador.

SET QUERYTAG = *constante-serie*
Asocia la *constante-serie*, a través de la columna QUERYTAG de la tabla EXPLAIN_STATEMENT, con la *sentencia explicable*. La *constante-serie* puede ser cualquier serie de caracteres de hasta 20 bytes de longitud. Si el valor

suministrado es inferior a 20 bytes de longitud, el valor se rellena por la derecha con blancos hasta la longitud necesaria.

Si no se especifica esta cláusula para una sentencia EXPLAIN, se utilizan blancos como el valor por omisión.

FOR *sentencia-sql-explicable*
Especifica la sentencia de SQL que se debe explicar. Esta sentencia puede ser cualquier sentencia CALL, SQL compuesto (dinámico), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES o VALUES INTO SQL válida. Si la sentencia EXPLAIN está incorporada en un programa, la *sentencia-sql-explicable* puede contener referencias a las variables 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 parámetro.

La *sentencia-sql-explicable* debe ser una sentencia de SQL válida que podría prepararse y ejecutarse independientemente de la sentencia EXPLAIN. No puede ser un nombre de sentencia ni una variable del lenguaje principal. Las sentencias de SQL que hacen referencia a los cursores definidos a través de CLP no son válidas para utilizarlas con esta sentencia.

Para explicar el SQL dinámico dentro de una aplicación, toda la sentencia EXPLAIN debe estar preparada dinámicamente.

FOR XQUERY *'sentencia-xquery-explicable'*
Especifica la sentencia de XQUERY que se debe explicar. Esta sentencia puede ser cualquier sentencia XQUERY válida.

Si la sentencia EXPLAIN está incorporada en un programa, la *'sentencia-xquery-explicable'* puede contener referencias a variables del lenguaje principal, siempre y cuando las variables del lenguaje principal no se utilicen en el nivel superior de la sentencia XQUERY, sino que se pasen a través de una función XMLQUERY, mediante un predicado XMLEXISTS, o una función XMLTABLE. Las variables del lenguaje principal deben definirse en el programa.

De manera similar, si EXPLAIN se prepara de forma dinámica, la *'sentencia-xquery-explicable'* puede contener marcadores de parámetro, siempre y cuando se siguen las mismas restricciones que para pasar variables del lenguaje principal.

De forma alternativa, se puede utilizar la función de DB2 XQUERY db2-fn:sqlquery para incorporar sentencias de SQL en referencias a variables del lenguaje principal y marcadores de parámetro.

La *sentencia-xquery-explicable* debe ser una sentencia de XQUERY válida que podría prepararse y ejecutarse independientemente de la sentencia EXPLAIN. Las sentencias de consulta que hacen referencia a los cursores definidos a través de CLP no se pueden utilizar con esta sentencia.

Notas

El recurso Explain utiliza los ID siguientes como el esquema al calificar tablas de explicación que se están llenando de datos:

- El ID de autorización de sesión SQL dinámico
- El ID de autorización de sentencia para SQL estático

El esquema se puede asociar con un conjunto de tablas de explicación o alias que apuntan a un conjunto de tablas de explicación en un esquema distinto. Si no se encuentran tablas de explicación en el esquema, el recurso Explain comprueba la existencia de tablas de explicación en el esquema SYSTOOLS e intenta utilizar dichas tablas.

La tabla siguiente muestra la interacción de las palabras clave de instantánea y la información de explicación.

Palabra clave especificada	¿Capturar información de Explain?	¿Tomar una instantánea para Visual Explain?
ninguna	Sí	No
FOR SNAPSHOT	No	Sí
WITH SNAPSHOT	Sí	Sí

Si no están especificadas las cláusulas FOR SNAPSHOT ni WITH SNAPSHOT, no se toma una instantánea de explicación.

Antes de realizar la invocación de la sentencia EXPLAIN, el usuario deberá haber creado las tablas de explicación. La información que genera esta sentencia se almacena en las tablas de explicación, en el esquema que se ha designado en el momento de compilarse la sentencia.

Si se producen errores durante la compilación de la *sentencia explicable* proporcionada, entonces no se almacena información en las tablas de explicación.

El plan de acceso para la *sentencia explicable* no se guarda y, por consiguiente, no se podrá invocar posteriormente. La información de explicación para la *sentencia explicable* se inserta cuando se compila la sentencia EXPLAIN misma.

Para una sentencia de consulta EXPLAIN estática, la información se inserta en las tablas de explicación 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 o PREP. Las sentencias EXPLAIN estáticas se pueden utilizar para hacer que las tablas de explicación se llenen para una sentencia de consulta estática específica entre muchas; simplemente, ponga un prefijo en la sentencia de destino con la sintaxis de sentencia EXPLAIN adecuada y vincule la aplicación sin utilizar las opciones BIND ni PREP de explicación. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de explicación real.

Las sentencias EXPLAIN estáticas de un procedimiento de SQL se evalúan cuando se compila el procedimiento.

Para las sentencias EXPLAIN de consulta con vinculación incremental, las tablas de explicación 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 de explicación se

llenan con datos, el calificador de la tabla de explicación y el ID de autorización utilizado durante el llenado de datos serán los pertenecientes al propietario del paquete. La sentencia EXPLAIN también se puede utilizar cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de explicación real.

Para sentencias EXPLAIN dinámicas, las tablas de explicación 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 consulta dinámica. La sentencia EXPLAIN debe utilizarse cuando es ventajoso establecer el campo QUERYNO o QUERYTAG en el momento de la invocación de EXPLAIN real.

Si la opción de vinculación REOPT se establece en ONCE y el registro especial CURRENT EXPLAIN MODE o CURRENT EXPLAIN SNAPSHOT se establece en REOPT, la ejecución de sentencias de consulta dinámica y estática que contienen variables del lenguaje principal, registros especiales, marcadores de parámetro o variables globales hará que sólo se capture la información de explicación para la sentencia cuando se vuelva a optimizar la sentencia. De forma alternativa, si la opción de vinculación REOPT se establece en ALWAYS, la información de explicación se capturará cada vez que se ejecuten estas sentencias.

Ejemplos

Ejemplo 1: Explique una sentencia SELECT simple y póngale el distintivo QUERYNO = 13.

```
EXPLAIN PLAN SET QUERYNO = 13
  FOR SELECT C1
  FROM T1
```

Ejemplo 2: Explique una sentencia SELECT simple y póngale el distintivo QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
  FOR SELECT C1
  FROM T1
```

Ejemplo 3: Explique una sentencia SELECT simple y póngale los distintivos QUERYNO = 13 y QUERYTAG = 'TEST13'.

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
  FOR SELECT C1
  FROM T1
```

Ejemplo 4: Intente obtener información de explicación cuando no existan tablas de explicación.

```
EXPLAIN ALL FOR SELECT C1
  FROM T1
```

Esta sentencia fallará porque no se han definido las tablas de explicación (SQLSTATE 42704).

Ejemplo 5: La siguiente sentencia será satisfactoria si se encuentra en la antememoria del paquete y ya se ha compilado utilizando REOPT ONCE.

```
EXPLAIN ALL WITH REOPT ONCE FOR SELECT C1
  FROM T1
  WHERE C1 = :<variable del lenguaje principal>
```

Ejemplo 6: el ejemplo siguiente utiliza la función db2-fn:xmlcolumn, que toma el nombre que distingue entre mayúsculas y minúsculas de una columna XML como un argumento y devuelve una secuencia XML que es la concatenación de valores de columna XML.

Imaginemos una tabla denominada BUSINESS.CUSTOMER con una columna XML denominada INFO. Una XQuery simple que devuelve todos los documentos de la columna INFO es:

```
EXPLAIN PLAN SELECTION
   FOR XQUERY 'db2-fn:xmlcolumn ("BUSINESS.CUSTOMER.INFO")'
```

Si un valor de la columna es nulo, la secuencia que se obtendrá para esa fila estará vacía.

FETCH

La sentencia FETCH 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.

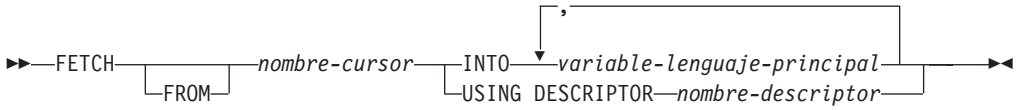
Invocación

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-lenguaje-principal*, **...**
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 apariciones 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 aparición SQLVAR.

Si las columnas resultantes de LOB o de tipo estructurado necesitan acomodarse, debe haber dos entradas SQLVAR para cada elemento de la lista de selección (o columna de la tabla de resultados).

SQLD debe establecerse en un valor mayor o igual que cero y menor o igual que SQLN.

La variable *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 capacidades de búsqueda adicionales. Por ejemplo cuando la tabla de resultados de un cursor es de sólo lectura, se puede utilizar la función SQLFetchScroll() para situar el cursor en cualquier punto dentro de dicha tabla de resultados.
- Para un cursor actualizable, se obtiene un bloqueo en una fila cuando se capta.
- Si la definición del cursor contiene una sentencia de cambio de datos de SQL o invoca una rutina que modifica datos de SQL, un error durante la operación de recuperación no hace que las filas modificadas se retrotraigan, aunque el error haga que se cierre el cursor.

Ejemplos

Ejemplo 1: En este ejemplo C, la sentencia FETCH coloca los resultados de la sentencia SELECT en las variables de programa `dnum`, `dname` y `mnum`. Cuando ya no quedan más filas para leer, se devuelve la condición de no encontrado.

```
EXEC SQL  DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

EXEC SQL  OPEN C1;

mientras (SQLCODE==0) {
    EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}

EXEC SQL  CLOSE C1;
```

Ejemplo 2: La sentencia FETCH utiliza una SQLDA.

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

FLUSH 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.

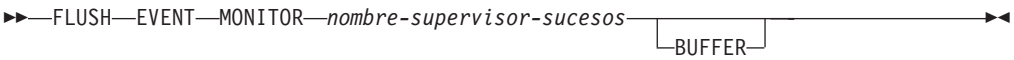
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

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

Sintaxis



Descripción

nombre-supervisor-sucesos
Nombre del supervisor de sucesos. Este nombre consta de una sola parte. Se trata de un identificador ordinario.

BUFFER
Indica que se deben grabar los almacenamientos intermedios del supervisor de sucesos. Si se especifica BUFFER, no se generan los registros parciales. Sólo se graban los datos que ya están presentes en los almacenamientos intermedios del supervisor de sucesos.

Notas

- Cuando se desecha el supervisor de sucesos no se restauran los valores del supervisor de sucesos. Esto significa que el registro del supervisor de sucesos que se habría generado si no se hubiese desechado, se seguirá generando cuando se active el suceso del supervisor normal.

FLUSH OPTIMIZATION PROFILE CACHE

Se pueden compilar diversas sentencias utilizando el mismo perfil de optimización. Para conseguir que el proceso de perfiles de optimización sea más eficaz, el perfil de optimización se procesa la primera vez que se utiliza para optimizar una sentencia y la salida se almacena en la antememoria de perfiles de optimización. Las posteriores referencias al perfil de optimización utilizan la versión procesada de la antememoria de perfiles de optimización.

Un perfil de optimización se debe eliminar de la antememoria de perfiles de optimización cuando la versión almacenada en SYSTOOLS.OPT_PROFILE se haya actualizado. Cuando la versión antigua se elimine de la antememoria, al optimizar las posteriores sentencias que utilicen el perfil de optimización se utilizará la versión nueva.

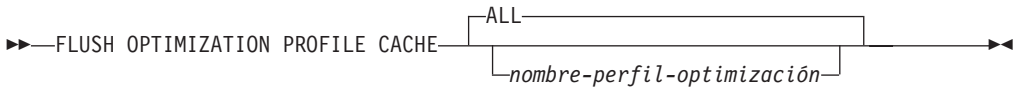
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir la autorización SYSADM o DBADM (SQLSTATE 42502).

Sintaxis



Descripción

nombre-perfil-optimización
Especifica el nombre del perfil de optimización que se debe desechar de la antememoria de perfiles de optimización. Si el nombre especificado no está calificado, se utilizará el valor del registro CURRENT DEFAULT SCHEMA como calificador implícito.

ALL
Especifica que todos los perfiles de todas las particiones de base de datos activas se deben desechar de la antememoria de perfiles de optimización.

Notas

- La sentencia FLUSH OPTIMIZATION PROFILE CACHE elimina todos o un solo perfil de optimización de la antememoria de perfiles de optimización. También causa la invalidación lógica de cualquier sentencia de SQL dinámico en antememoria que se haya preparado con el perfil de optimización en cuestión.
- Cuando se realiza la siguiente petición para la misma sentencia de SQL, se vuelven a generar planes de acceso nuevos para los planes dinámicos invalidados.
- Los paquetes que hacen referencia a un perfil de optimización eliminado de la antememoria de perfiles de optimización por esta sentencia se deben volver a enlazar explícitamente para permitir que se generen nuevos planes de acceso.

Ejemplos

Ejemplo 1: El perfil de optimización "Rick"."Foo" se desecha de la antememoria de perfiles de optimización.

```
SET CURRENT SCHEMA = '"Rick"'
FLUSH OPTIMIZATION PROFILE CACHE "Foo"
```

Ejemplo 2: El perfil de optimización JOHN.ALL se elimina de la antememoria de perfiles de optimización.

```
SET CURRENT SCHEMA = '"Rick"'
FLUSH OPTIMIZATION PROFILE CACHE JOHN.ALL
```

Mensajes

- No se emitirá ningún error si la antememoria de perfiles de optimización está vacía o si los perfiles de optimización especificados (ya sea explícita o implícitamente) no existen en la antememoria de perfiles de optimización.

FLUSH PACKAGE CACHE

La sentencia FLUSH PACKAGE CACHE elimina todas las sentencias de SQL dinámico colocadas en antememoria que actualmente están en la antememoria de paquetes. Esta sentencia da lugar a la invalidación lógica de cualquier sentencia de SQL dinámico colocada en antememoria y hace que DB2 compile implícitamente la siguiente petición para la misma sentencia de SQL.

Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

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

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 dinámico colocada en antememoria que actualmente esté utilizándose podrá seguir existiendo en la antememoria del paquete hasta que el usuario actual ya no la necesite; el nuevo usuario siguiente de la misma sentencia hará que DB2 realice una preparación implícita de la sentencia y el nuevo usuario ejecutará la nueva versión de la sentencia de SQL dinámico colocada en antememoria.

FOR

La sentencia FOR ejecuta una sentencia o grupo de sentencias para cada fila de una tabla.

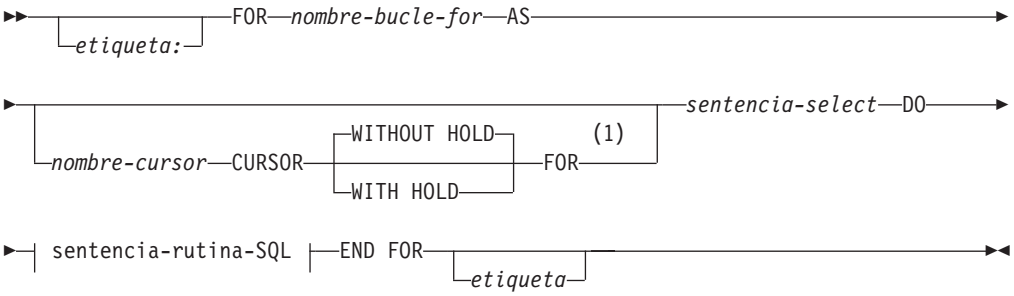
Invocación

Esta sentencia 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.

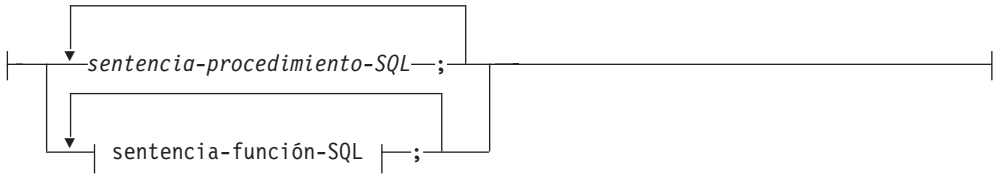
Autorización

No se requieren privilegios para invocar la sentencia FOR. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia FOR. Para obtener información acerca de la autorización que se necesita para utilizar un cursor, consulte "DECLARE CURSOR".

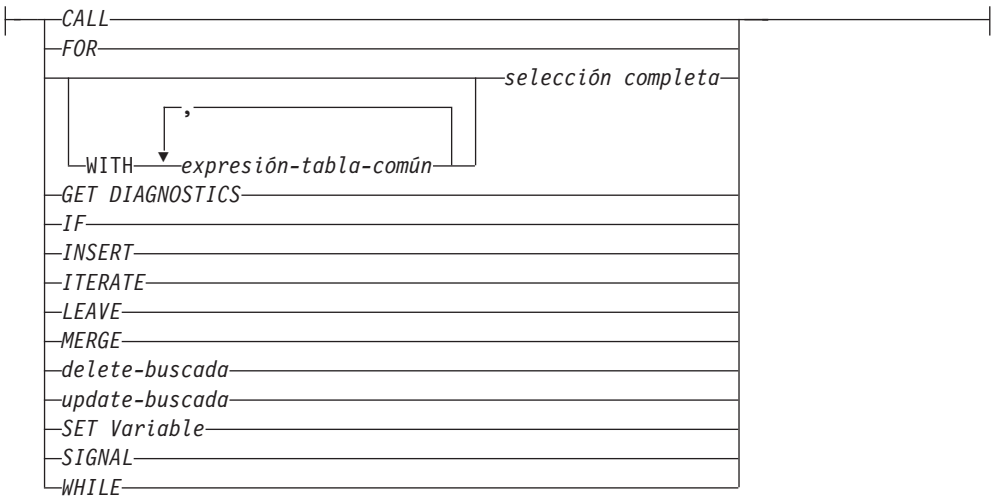
Sintaxis



sentencia-rutina-SQL:



sentencia-función-SQL:



Notas:

1 Esta opción sólo se puede utilizar en el contexto de un procedimiento de 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 una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

nombre-bucle-for
Especifica una etiqueta para la sentencia compuesta implícita que se genera para implementar la sentencia FOR. Sigue las normas para la etiqueta de una sentencia compuesta excepto en que no se puede utilizar con una sentencia ITERATE o LEAVE en la sentencia FOR. El *nombre-bucle-for* se utiliza para calificar los nombres de columna devueltos por la *sentencia-select* especificada.

nombre-cursor
Designa el cursor utilizado para seleccionar filas de la tabla de resultados de la sentencia SELECT. Si no se especifica, DB2 genera un nombre de cursor exclusivo. Para obtener una descripción de WITHOUT HOLD o WITH HOLD, consulte “DECLARE CURSOR”.

sentencia-select
Especifica la sentencia SELECT del cursor. Todas las columnas de la lista de selección deben tener un nombre y no pueden existir dos columnas con el mismo nombre.

En un activador, una función, un método o una sentencia dinámica compuesta, la *sentencia-select* sólo debe constar de una *selección completa* con expresiones de tabla comunes opcionales.

sentencia-procedimiento-SQL
Especifica una o varias sentencias que se deben invocar para cada fila de la tabla. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte *sentencia-procedimiento-SQL* en “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 de 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 CONCAT ','
        CONCAT firstnme CONCAT ' ' CONCAT midinit;
      INSERT INTO tnames VALUES (fullname);
  END FOR;
END
```

FREE LOCATOR

La sentencia FREE LOCATOR elimina la asociación entre una variable localizadora y su valor.

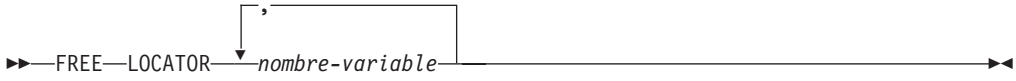
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

LOCATOR *nombre-variable,* **...**
Identifica una o varias variables localizadoras que deben declararse de acuerdo con las normas para la declaración de variables localizadoras.

La variable-localizadora debe tener actualmente un localizador asignado a ella. Es decir, deberá haberse asignado un localizador durante esta unidad de trabajo (por medio de una sentencia CALL, FETCH, SELECT INTO o VALUES INTO) y no deberá haberse liberado posteriormente (por medio de una sentencia FREE LOCATOR); de lo contrario, se devolverá un error (SQLSTATE 0F001).

Si se especifica más de un localizador, se liberan todos los localizadores que se pueden liberar, sin tener en cuenta los errores detectados en otros localizadores de la lista.

Ejemplo

En un programa COBOL, libere las variables localizadoras de BLOB, TKN-VIDEO y TKN-BUF, y la variable localizadora de CLOB, LIFE-STORY-LOCATOR.

```
EXEC SQL
FREE LOCATOR :TKN-VIDEO, :TKN-BUF, :LIFE-STORY-LOCATOR
END-EXEC.
```

GET DIAGNOSTICS

La sentencia GET DIAGNOSTICS se utiliza para obtener información acerca de la sentencia de SQL ejecutada anteriormente.

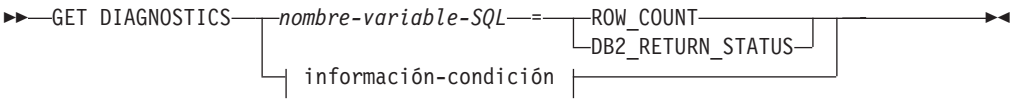
Invocación

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.

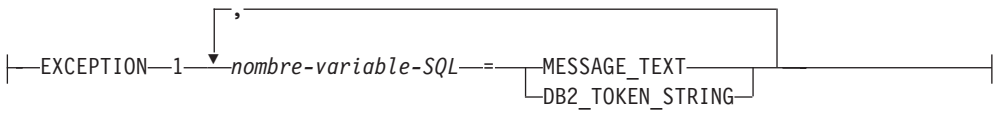
Autorización

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 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

- La sentencia GET DIAGNOSTICS no cambia el contenido del área de diagnósticos (SQLCA). Las variables especiales SQLSTATE o SQLCODE declaradas en un procedimiento de SQL se establecen en el SQLSTATE o SQLCODE devuelto por la ejecución de la sentencia GET DIAGNOSTICS.
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - RETURN_STATUS puede especificarse en lugar de DB2_RETURN_STATUS.

Ejemplos

En un procedimiento de SQL, ejecute una sentencia GET DIAGNOSTICS para determinar cuántas filas se actualizaron.

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
   LANGUAGE SQL
   BEGIN
      DECLARE SQLSTATE CHAR(5);
      DECLARE rcount INTEGER;
      UPDATE CORPDATA.PROJECT
          SET PRSTAFF = PRSTAFF + 1.5
        WHERE DEPTNO = deptnbr;
      GET DIAGNOSTICS rcount = ROW_COUNT;
-- En este momento, rcount contiene el número de filas que se actualizaron.
...
   END
```

Dentro de un procedimiento de SQL, procese el valor de estado devuelto por la invocación de un procedimiento denominado TRYIT, el cual puede devolver explícitamente un valor positivo, lo que indica un error de usuario, o encontrar errores de SQL que producirían un valor de estado de retorno negativo. Si el procedimiento se ejecuta satisfactoriamente, devuelve un valor igual a cero.

```
CREATE PROCEDURE TESTIT ()
   LANGUAGE SQL
   A1:BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
```

```
   CALL TRYIT;
   GET DIAGNOSTICS RETVAL = DB2_RETURN_STATUS;
   IF RETVAL <> 0 THEN
      ...
      LEAVE A1;
   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.

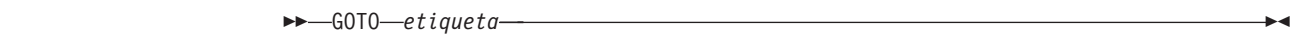
Invocación

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 bases de datos)

Este formato de la sentencia GRANT otorga las autorizaciones que se aplican a toda la base de datos (en lugar de privilegios que se aplican a objetos específicos de la base de datos).

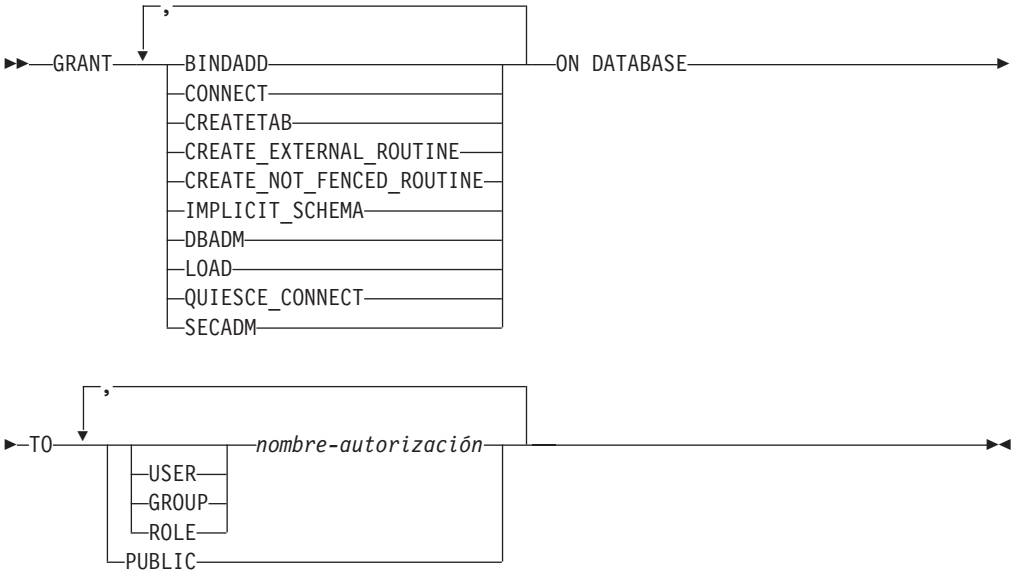
Invocación

Esta sentencia 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 o SECADM, se precisa 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 las 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, salvo la autorización del administrador de seguridad (SECADM). Un administrador de bases de datos tiene casi todos los privilegios para casi todos los objetos de la base de datos. Las únicas excepciones son los privilegios que forman parte de la autorización del administrador de seguridad.

Un administrador de bases de datos puede otorgar cualquier privilegio que forma parte de una autorización de la base de datos a otros usuarios.

Todas las autorizaciones de bases de datos, salvo SECADM, 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.

SECADM
Otorga la autorización de administrador de seguridad. La autorización SECADM sólo se puede otorgar a un usuario. No se puede otorgar a un grupo ni a una función (SQLSTATE 42521) ni a PUBLIC (SQLSTATE 42508). La autorización permite realizar las operaciones siguientes a quien la posee:

- Crear y descartar objetos de seguridad como, por ejemplo, políticas de comprobación, funciones, etiquetas de seguridad, componentes de etiqueta de seguridad, políticas de seguridad y contextos fiables
- Otorgar y revocar funciones, etiquetas de seguridad y exenciones
- Otorgar y revocar el privilegio SETSESSIONUSER
- Ejecutar TRANSFER OWNERSHIP en objetos propiedad de otros

A diferencia de otras autorizaciones de bases de datos, la autorización SYSADM no permite realizar las acciones anteriores a quien la posee.

TO
Especifica a quién se otorgan las autorizaciones.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga las autorizaciones a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”. DBADM no se puede otorgar a PUBLIC.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.

  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.

Notas

- No se puede otorgar la autorización DBADM al grupo especial PUBLIC. Por lo tanto, otorgar autorización DBADM a una función *nombre-función* falla si *nombre-función* se otorga a PUBLIC ya sea de forma directa o indirecta (SQLSTATE 42508).
  - La función *nombre-función* se otorga directamente a PUBLIC si se ha emitido la siguiente sentencia:

    ```
    GRANT ROLE nombre-función TO PUBLIC
    ```

  - La función *nombre-función* se otorga indirectamente a PUBLIC si se han emitido las siguientes sentencias:

    ```
    GRANT ROLE nombre-función TO ROLE nombre2-función
    GRANT ROLE nombre2-función TO PUBLIC
    ```

- ***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 la autorización BINDADD en la base de datos a un grupo denominado D024. Hay un grupo y un usuario llamados D024 en el sistema.

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

Observe que debe especificarse la palabra clave GROUP; de lo contrario, se producirá un error ya que existen tanto un usuario como un grupo llamados D024. Cualquier miembro del grupo D024 podrá enlazar paquetes en la base de datos, pero el usuario D024 no podrá (a menos que también sea miembro del grupo D024, se le haya otorgado la autorización BINDADD previamente o se haya otorgado la autorización BINDADD a otro grupo del que D024 sea miembro).

Ejemplo 3: Proporcione la autorización de administrador de seguridad al usuario Walid.

```
GRANT SECADM ON DATABASE TO USER Walid
```

GRANT (Exención)

Este formato de sentencia GRANT otorga a un usuario, grupo o función una exención en una norma de acceso para la política de seguridad de control de acceso basado en etiquetas (LBAC). Cuando el usuario que mantiene la exención accede a los datos de una tabla protegida por esa política de seguridad, la norma indicada no se impondrá al decidir si se puede acceder a los datos.

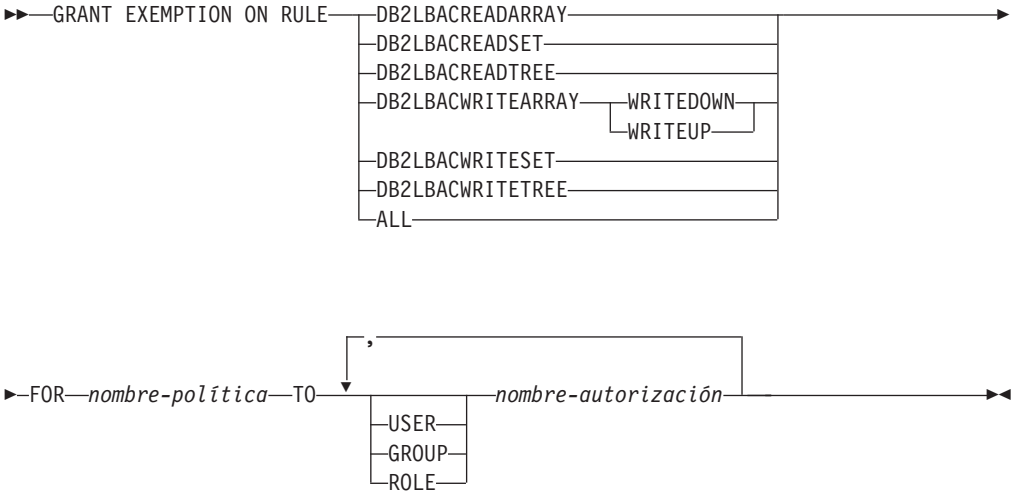
Invocación

Esta sentencia 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

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

Sintaxis



Descripción

EXEMPTION ON RULE
Otorga una exención en una norma de acceso.

DB2LBACREADARRAY
Otorga una en la norma DB2LBACREADARRAY predefinida.

DB2LBACREADSET
Otorga una exención en la norma DB2LBACREADSET predefinida.

DB2LBACREADTREE
Otorga una en la norma DB2LBACREADTREE predefinida.

DB2LBACWRITEARRAY
Otorga una exención en la norma DB2LBACWRITEARRAY predefinida.

WRITEDOWN
Especifica que la exención sólo se aplica a la escritura hacia abajo.

WRITEUP
Especifica que la exención sólo se aplica a la escritura hacia arriba.

DB2LBACWRITESET
Otorga una exención en la norma DB2LBACWRITESET predefinida.

DB2LBACWRITETREE
Otorga una exención en la norma DB2LBACWRITETREE predefinida.

ALL
Otorga una exención en todas las normas predefinidas.

FOR *nombre-política*
Identifica la política de seguridad para la que se otorga la exención. La exención sólo será efectiva para las tablas que estén protegidas por esta política de seguridad. El nombre debe identificar a una política de seguridad ya descrita en el catálogo (SQLSTATE 42704).

TO
Especifica a quién se le otorga la exención.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- Si la política de seguridad no está definida para que tenga en cuenta el acceso a través de grupos o funciones, se ignorará cualquier exención otorgada a un grupo o función cuando se intente el acceso.

Notas

- Por omisión, cuando se crea una política de seguridad, sólo se tienen en cuenta las exenciones otorgadas a un usuario individual. Para que se tengan en cuenta los grupos o las funciones para la política de seguridad, deberá emitir la

sentencia ALTER SECURITY POLICY y especificar USE GROUP AUTHORIZATION o USE ROLE AUTHORIZATION que sea de aplicación.

Ejemplos

Ejemplo 1: Otorgar una exención en la norma de acceso DB2LBACREADSET para la política de seguridad DATA_ACCESS del usuario WALID.

```
GRANT EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS TO USER WALID
```

Ejemplo 2: Otorga una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEDOWN para la política de seguridad DATA_ACCESS al usuario BOBBY.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
  FOR DATA_ACCESS TO USER BOBBY
```

Ejemplo 3: Otorga una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEUP para la política de seguridad DATA_ACCESS al usuario BOBBY.

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
  FOR DATA_ACCESS TO USER BOBBY
```

GRANT (privilegios de variable global)

Este formato de la sentencia GRANT otorga uno o más privilegios de una variable global creada.

Invocación

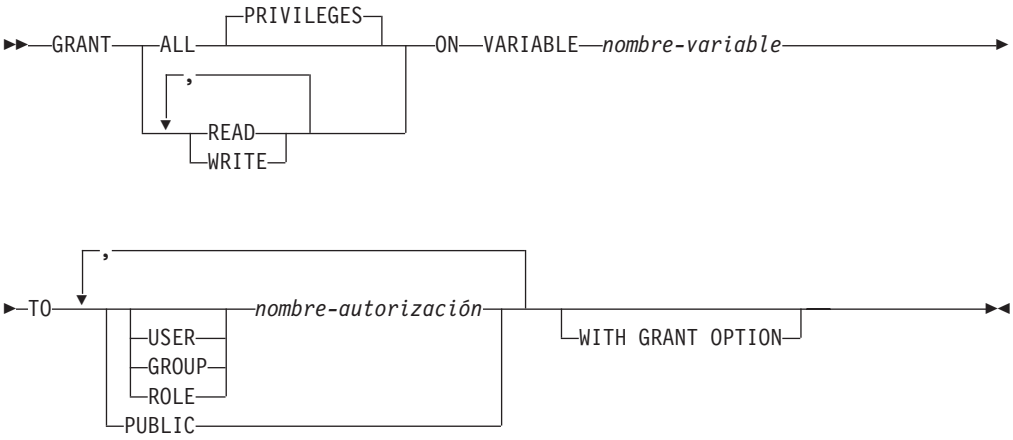
Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que se puede preparar dinámicamente.

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 la variable global
- Autorización SYSADM o DBADM

Sintaxis



Descripción

ALL PRIVILEGES
Otorga todos los privilegios sobre la variable global especificada.

READ
Otorga el privilegio para leer el valor de la variable global especificada.

WRITE
Otorga el privilegio para asignar un valor a la variable global especificada.

ON VARIABLE *nombre-variable*
Identifica la variable global sobre la que va a otorgarse uno o más privilegios. El *nombre-variable*, incluyendo un calificador implícito o explícito, debe identificar una variable global que ya exista en el servidor actual (SQLSTATE 42704).

TO
Especifica a quién se otorgan los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

ROLE
Especifica que el *nombre-autorización* identifique una función existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios especificados a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte "Autorizaciones, privilegios y propiedad de objetos".

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado pueda otorgar los privilegios a otros usuarios. Si se omite la cláusula WITH GRANT OPTION, los *nombre-autorización* especificados no pueden otorgar los privilegios a otros usuarios a menos que dicha autorización se haya recibido de alguna otra fuente.

Normas

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se asume GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.

Ejemplo

Otorgue el privilegio READ y WRITE en la variable global MYSCHEMA.MYJOB_PRINTER para el usuario ZUBIRI.

```
   GRANT READ, WRITE ON VARIABLE
MYSCHEMA.MYJOB_PRINTER TO ZUBIRI
```

GRANT (privilegios de índice)

Esta forma de la sentencia GRANT otorga el privilegio CONTROL en índices.

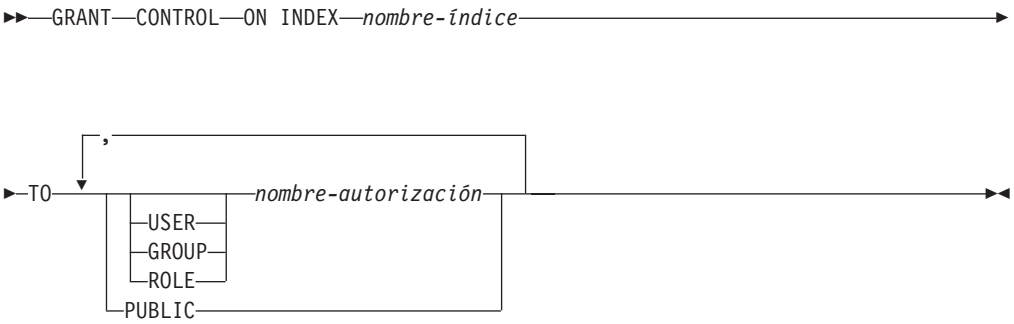
Invocación

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

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

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.

Ejemplo

```
GRANT CONTROL ON INDEX DEPTIDX TO USER KIESLER
```

GRANT (privilegios de paquete)

Esta forma de la sentencia GRANT otorga los privilegios en un paquete.

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).

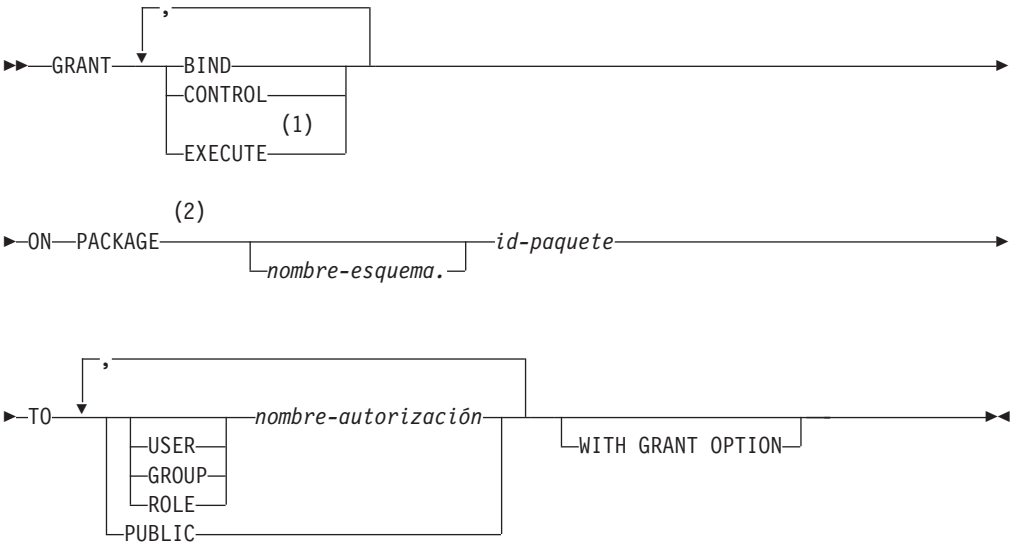
Autorización

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

- Privilegio CONTROL para el paquete referenciado
- WITH GRANT OPTION para cada privilegio identificado en *nombre-paquete*
- Autorización SYSADM o DBADM

Es necesaria la autorización SYSADM o DBADM para otorgar el privilegio CONTROL.

Sintaxis



Notas:

1 RUN se puede utilizar como sinónimo de EXECUTE.

2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

Descripción

BIND
Otorga el privilegio para enlazar un paquete. El privilegio BIND permite a un usuario volver a emitir el mandato BIND para ese paquete o bien emitir el mandato REBIND. También permite a un usuario crear una nueva versión de un paquete existente.

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado pueda otorgar (GRANT) los privilegios a otros usuarios.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios que pueden aplicarse, excepto CONTROL (SQLSTATE 01516).

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:

- Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
- Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
- Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
- Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
- Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
- Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.

Notas

- Los privilegios de paquete se aplican a todas las versiones de un paquete (es decir, a todos los paquetes que comparten el mismo ID de paquete y esquema de paquete). No es posible restringir el acceso sólo a una única versión. Puesto que el privilegio CONTROL se otorga implícitamente al usuario que enlaza el paquete, si dos usuarios distintos enlazan dos versiones de un paquete, a ambos usuarios se otorgará implícitamente acceso al paquete del otro.

Ejemplos

Ejemplo 1: Otorgue el privilegio EXECUTE en PACKAGE CORPDATA.PKGA a PUBLIC.

```
GRANT EXECUTE
  ON PACKAGE CORPDATA.PKGA
   TO PUBLIC
```

Ejemplo 2: Otorgue (GRANT) el privilegio EXECUTE en el paquete CORPDATA.PKGA a un usuario denominado EMPLOYEE. No hay ningún grupo ni usuario llamado EMPLOYEE.

```
GRANT EXECUTE ON PACKAGE
  CORPDATA.PKGA TO EMPLOYEE
```

o

```
GRANT EXECUTE ON PACKAGE
  CORPDATA.PKGA TO USER EMPLOYEE
```

GRANT (Función)

Este formato de la sentencia GRANT otorga funciones a usuarios, grupos u otras funciones.

Invocación

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

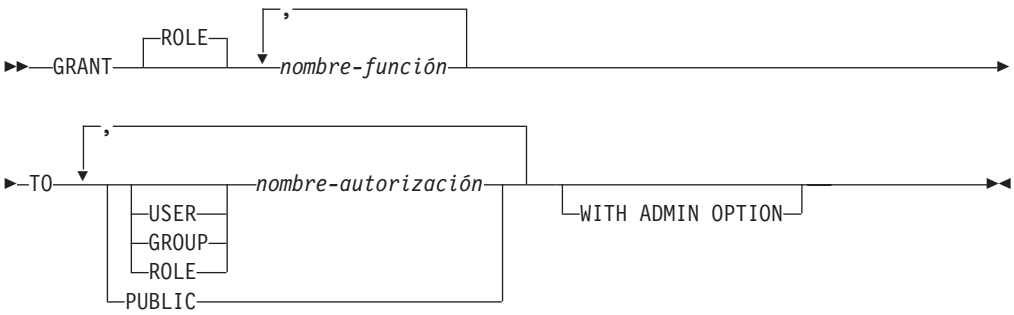
Autorización

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

- WITH ADMIN OPTION en la función
- Autorización SECADM

Es necesaria la autorización SECADM para otorgar el privilegio WITH ADMIN OPTION a un *nombre-autorización*.

Sintaxis



Descripción

ROLE *nombre-función*,...
Identifica una o más funciones a otorgar. Cada *nombre-función* debe identificar una función existente en el servidor actual (SQLSTATE 42704).

TO
Especifica a quién se otorga la función.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

ROLE
Especifica que el *nombre-autorización* identifica un rol existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga las funciones especificadas a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte "Autorizaciones, privilegios y propiedad de objetos".

WITH ADMIN OPTION
Permite al *nombre-autorización* especificado revocar el *nombre-función* a otros o asociar un comentario a la función. No permite que el *nombre-autorización* descarte la función.

Normas

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se asume GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.
- Las jerarquías de funciones pueden crearse otorgando una función a otra función. Sin embargo, los ciclos no se permiten (SQLSTATE 428GF). Por ejemplo, si la función R1 se otorga a otra función R2, la función R2 (u alguna otra función R*n* que contenga R2) no podría volverse a otorgar a R1 ya que esto ocasionaría un ciclo.

Notas

- Cuando la función R1 se otorga a otra función R2, R2 contiene R1.
- La autorización DBADM no puede otorgarse a PUBLIC. Por tanto:
  - La acción de otorgar la función R1 a PUBLIC falla (SQLSTATE 42508) si la función R1 retiene la autorización DBADM directa o indirectamente.
    - La función R1 retiene la autorización DBADM directamente si se ha emitido la siguiente sentencia:

      ```
      GRANT DBADM ON DATABASE TO ROLE R1
      ```

    - La función R1 retiene la autorización DBADM indirectamente si se han emitido las siguientes sentencias:

      ```
      GRANT DBADM ON DATABASE TO ROLE R2

      GRANT ROLE R2 TO ROLE R1
      ```

  - La acción de otorgar la función R1, que retiene la autorización DBADM, a la función R2 falla (SQLSTATE 42508) si la función R2 se otorga a PUBLIC directa o indirectamente.
    - La función R2 se otorga a PUBLIC directamente si se ha emitido la siguiente sentencia:

      ```
      GRANT ROLE R2 TO PUBLIC
      ```

- La función R2 se otorga a PUBLIC indirectamente si se han emitido las siguientes sentencias:

  ```
  GRANT ROLE R2 TO ROLE R3

  GRANT ROLE R3 TO PUBLIC
  ```

Ejemplos

Ejemplo 1: Otorgue la función INTERN a la función DOCTOR y la función DOCTOR a la función SPECIALIST.

```
GRANT ROLE INTERN TO ROLE DOCTOR

GRANT ROLE DOCTOR TO ROLE SPECIALIST
```

Ejemplo 2: Otorgue la función INTERN a PUBLIC.

```
GRANT ROLE INTERN TO PUBLIC
```

Ejemplo 3: Otorgue la función SPECIALIST al usuario BOB y al grupo TORONTO.

```
GRANT ROLE SPECIALIST TO USER BOB, GROUP TORONTO
```

GRANT (privilegios de rutina)

Esta forma de la sentencia GRANT otorga privilegios para una rutina (función, método o procedimiento).

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

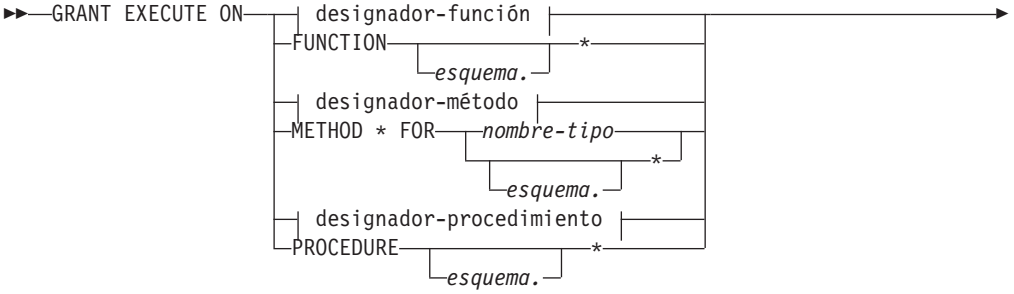
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

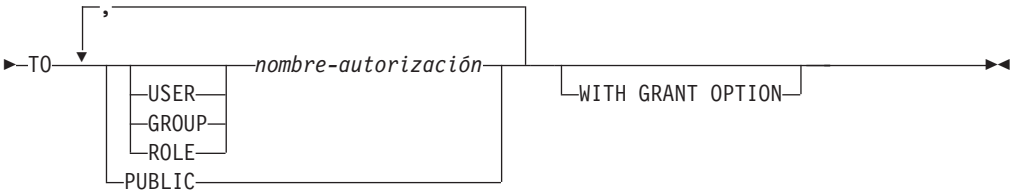
- WITH GRANT OPTION para EXECUTE en la rutina
- Autorización SYSADM o DBADM

Para otorgar todos los privilegios EXECUTE de rutina del esquema o tipo, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

- WITH GRANT OPTION para EXECUTE en todas las rutinas existentes y futuras (del tipo especificado) dentro del esquema especificado
- Autorización SYSADM o DBADM

Sintaxis





Descripción

EXECUTE
Otorga 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 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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

PUBLIC
Otorga el privilegio EXECUTE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados puedan otorgar (GRANT) el privilegio EXECUTE a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar el privilegio EXECUTE a otros usuarios si éstos:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar el privilegio EXECUTE desde alguna otra fuente.

Normas

- No es posible otorgar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA y no se han otorgado privilegios, se devolverá un mensaje de aviso (SQLSTATE 01007). Si el usuario que otorga privilegios no dispone de ningún privilegio sobre el objeto de la operación de concesión de privilegios, se devolverá un error (SQLSTATE 42501).

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
```

GRANT (privilegios de esquema)

Esta forma de la sentencia GRANT otorga privilegios en un esquema.

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).

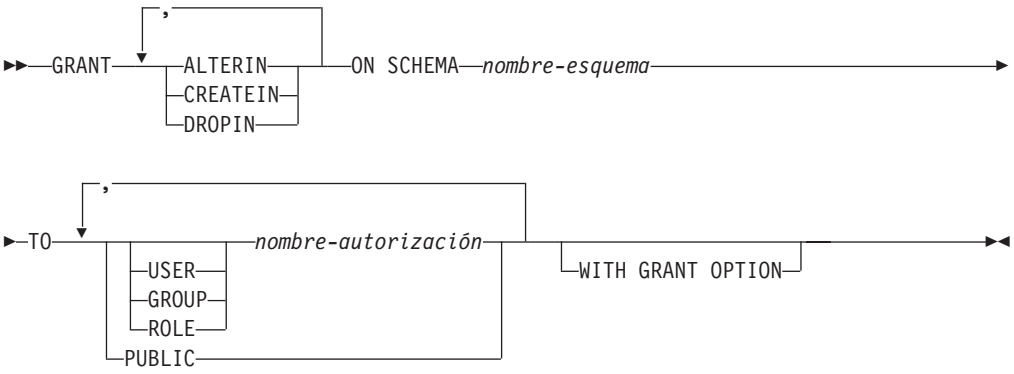
Autorización

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-esquema*
- Autorización SYSADM o DBADM

Los usuarios no pueden otorgar privilegios en ninguno de los nombres de esquema siguientes: SYSIBM, SYSCAT, SYSFUN o SYSSTAT (SQLSTATE 42501).

Sintaxis



Descripción

ALTERIN
Otorga el privilegio para modificar o comentar todos los objetos del esquema. El propietario de un esquema creado explícitamente recibe automáticamente el privilegio ALTERIN.

CREATEIN
Otorga el privilegio para crear objetos en el esquema. Siguen necesitándose las demás autorizaciones o privilegios necesarios para crear el objeto (como CREATETAB). El propietario de un esquema creado explícitamente recibe automáticamente el privilegio CREATEIN. En un esquema creado implícitamente se otorga automáticamente el privilegio CREATEIN a PUBLIC.

DROPIN
Otorga el privilegio para 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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga los privilegios a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

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

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un

mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

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
```

GRANT (Etiqueta de seguridad)

Este formato de sentencia GRANT otorga una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC) a un usuario, grupo o función para el acceso de lectura, para el acceso de grabación o para ambos.

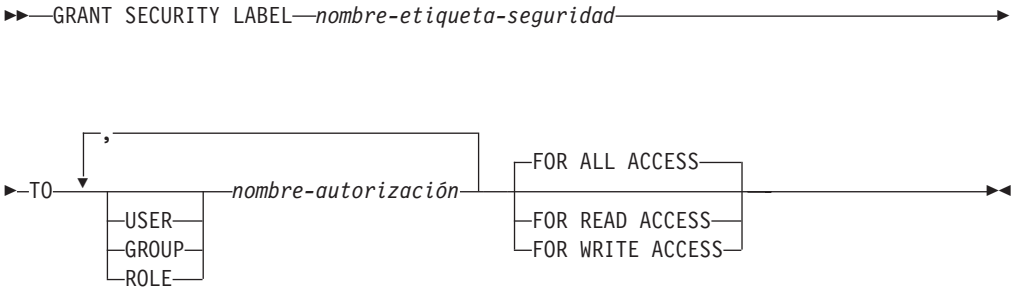
Invocación

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

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

Sintaxis



Descripción

SECURITY LABEL *nombre-etiqueta-seguridad*
Otorga la etiqueta de seguridad *nombre-etiqueta-seguridad*. Se debe calificar el nombre con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704).

TO
Especifica a quién se le otorga la etiqueta de seguridad especificada.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles.

FOR ALL ACCESS
Indica que la etiqueta de seguridad se otorgará a los accesos de lectura y grabación.

FOR READ ACCESS
Indica que la etiqueta de seguridad se otorgará solamente para el acceso de lectura.

FOR WRITE ACCESS
Indica que la etiqueta de seguridad se otorgará solamente para el acceso de grabación.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- Para cualquier política de seguridad, se puede otorgar como máximo a *nombre-autorización* una etiqueta de seguridad de esa política para el acceso de lectura y una para el acceso de grabación. Si el otorgado ya mantiene una etiqueta de seguridad para el tipo de acceso (lectura o grabación) indicado y que forma parte de la política de seguridad que califica a *nombre-etiqueta-seguridad*, se devuelve un error (SQLSTATE 428GR).
- Si la política de seguridad no está definida para que tenga en cuenta el acceso a través de grupos o funciones, se ignorará cualquier etiqueta de seguridad otorgada a un grupo o función cuando se intente el acceso.
- Si un *nombre-autorización* mantiene diferentes etiquetas de seguridad para los accesos de lectura y de grabación, las etiquetas de seguridad deben cumplir los criterios siguientes (SQLSTATE 428GQ):
  - Si algún componente de las etiquetas de seguridad es del tipo ARRAY, el valor de ese componente debe ser el mismo en ambas etiquetas de seguridad.
  - Si algún componente de las etiquetas de seguridad es del tipo SET, cada elemento del valor de ese componente en la etiqueta de seguridad de grabación debe formar parte también del valor de ese componente en la etiqueta de seguridad de lectura.
  - Si algún componente de las etiquetas de seguridad es del tipo TREE, cada elemento del valor de ese componente en la etiqueta de seguridad de grabación debe ser el mismo o un descendiente de uno de los elementos del valor de ese mismo componente en la etiqueta de seguridad de lectura.

Notas

- Por omisión, cuando se crea una política de seguridad, sólo se tienen en cuenta las etiquetas de seguridad otorgadas a un usuario individual. Para que se tengan en cuenta los grupos o las funciones para la política de seguridad, deberá emitir la sentencia ALTER SECURITY POLICY y especificar USE GROUP AUTHORIZATION o USE ROLE AUTHORIZATION que sea de aplicación.

Ejemplos

Ejemplo 1: La sentencia siguiente otorga dos etiquetas de seguridad al usuario GUYLAINE. Se otorga la etiqueta de seguridad EMPLOYEESECLABELREAD para el acceso de lectura y la etiqueta de seguridad EMPLOYEESECLABELWRITE para el acceso de grabación. Ambas etiquetas de seguridad forman parte de la política de seguridad DATA_ACCESS.

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
  TO USER GUYLAINE FOR READ ACCESS

GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
  TO USER GUYLAINE FOR WRITE ACCESS
```

Al mismo usuario se le otorga ahora la etiqueta de seguridad BEGINNER para los accesos de lectura y grabación. Esto no provoca un error, ya que BEGINNER forma parte de la política de seguridad CLASSPOLICY, y las etiquetas de seguridad ya mantenidas forman parte de la política de seguridad DATA_ACCESS.

```
GRANT SECURITY LABEL CLASSPOLICY.BEGINNER
  TO USER GUYLAINE FOR ALL ACCESS
```

GRANT (privilegios de secuencia)

Esta forma de la sentencia GRANT otorga privilegios para una secuencia.

Invocación

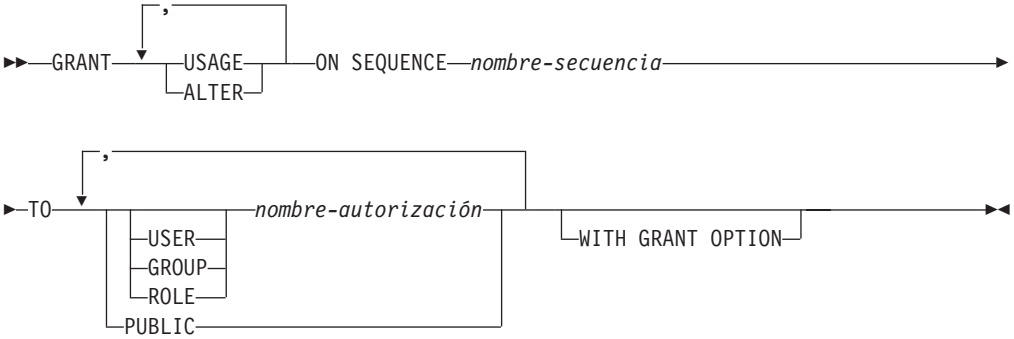
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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 un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

PUBLIC
Otorga los privilegios especificados a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado pueda otorgar los privilegios indicados a otros usuarios.

Si se omite WITH GRANT OPTION, el *nombre-autorización* especificado sólo puede otorgar los privilegios indicados a otros si:

- tienen autorización SYSADM o DBADM, o bien
- han recibido la capacidad de otorgar los privilegios especificados desde alguna otra fuente.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de concesión de privilegios.)

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
```

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.

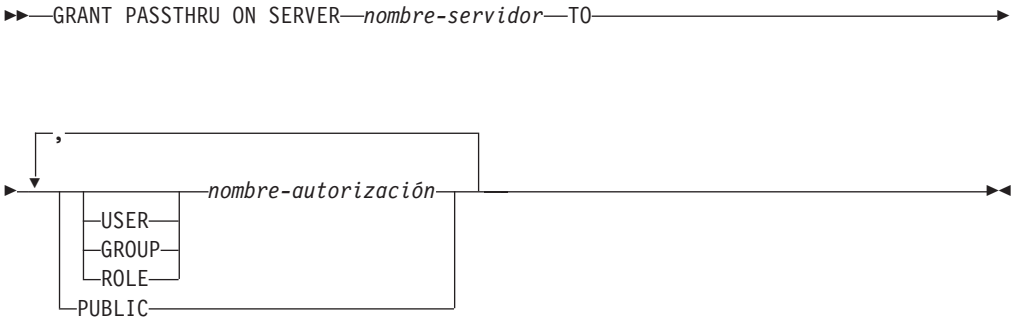
Invocación

Esta sentencia puede incorporarse 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

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

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga a un conjunto de usuarios (ID de autorización) el privilegio de

realizar un paso a través de *nombre-servidor*. Para obtener más información, consulte el apartado “Autorizaciones, privilegios y propiedad de objetos”.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.

Ejemplos

Ejemplo 1: Otorgue a R. Smith y a J. Jones el privilegio de paso a través para la fuente de datos SERVALL. Sus ID de autorización son RSMITH y JJONES.

```
   GRANT PASSTHRU ON SERVER SERVALL
TO USER RSMITH,
USER JJONES
```

Ejemplo 2: Otorgue el privilegio de realizar un paso a través para la fuente de datos EASTWING a un grupo cuyo ID de autorización es D024. Existe un usuario cuyo ID de autorización también es D024.

```
   GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

Debe especificarse la palabra clave GROUP; de lo contrario se producirá un error porque D024 es el ID de un usuario y el ID del grupo especificado (SQLSTATE 56092). Cualquier miembro del grupo D024 tendrá permitido realizar un paso a través para EASTWING. Por lo tanto, si el usuario D024 pertenece al grupo, este usuario podrá realizar un paso a través para EASTWING.

GRANT (privilegio SETSESSIONUSER)

Este formato de la sentencia GRANT otorga el privilegio SETSESSIONUSER a uno o más ID de autorización. El privilegio permite a quien lo mantiene utilizar la sentencia SET SESSION AUTHORIZATION para configurar la autorización de la sesión a uno de los ID de autorización de un conjunto especificado.

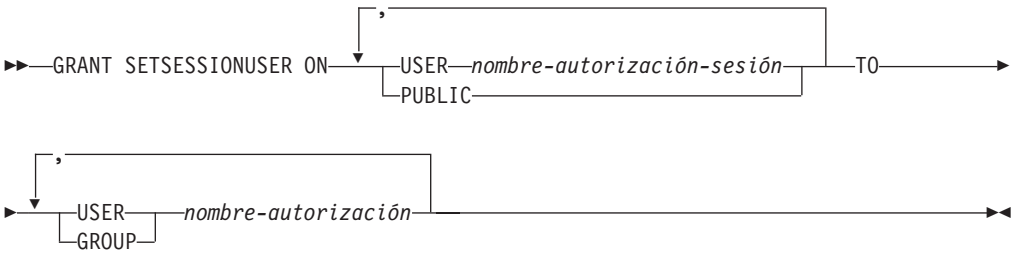
Invocación

Esta sentencia 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

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

Sintaxis



Descripción

SETSESSIONUSER ON
Otorga el privilegio para asumir la identidad de un nuevo ID de autorización.

USER *nombre-autorización-sesión*
Especifica el ID de autorización que el *nombre-autorización* será capaz de asumir, mediante la sentencia SET SESSION AUTHORIZATION. El *nombre-autorización-sesión* debe identificar a un usuario, no a un grupo.

PUBLIC
Especifica que el otorgado podrá asumir cualquier ID válido de autorización, mediante la sentencia SET SESSION AUTHORIZATION.

TO
Especifica a quién se otorga el privilegio.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER ni GROUP, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.

Ejemplos

Ejemplo 1: La sentencia siguiente otorga al usuario PAUL la posibilidad de configurar la autorización de la sesión al usuario WALID, y por lo tanto ejecutar sentencias como WALID.

```
GRANT SETSESSIONUSER ON USER WALID
  TO USER PAUL
```

Ejemplo 2: La sentencia siguiente otorga al usuario GUYLAINE la posibilidad de configurar la autorización de la sesión para el usuario BOBBY. También le otorga la posibilidad de configurar la autorización de la sesión para los usuarios RICK y KEVIN.

```
GRANT SETSESSIONUSER ON USER BOBBY, USER RICK, USER KEVIN
  TO USER GUYLAINE
```

Ejemplo 3: La sentencia siguiente otorga al usuario WALID y a todos los que pertenezcan a los grupos ADMINS y ACCTG la posibilidad de configurar la autorización de la sesión a cualquier usuario.

```
GRANT SETSESSIONUSER ON PUBLIC TO USER WALID, GROUP ADMINS, ACCTG
```

GRANT (privilegios de espacio de tablas)

Esta forma de la sentencia GRANT otorga privilegios para un espacio de tablas.

Invocación

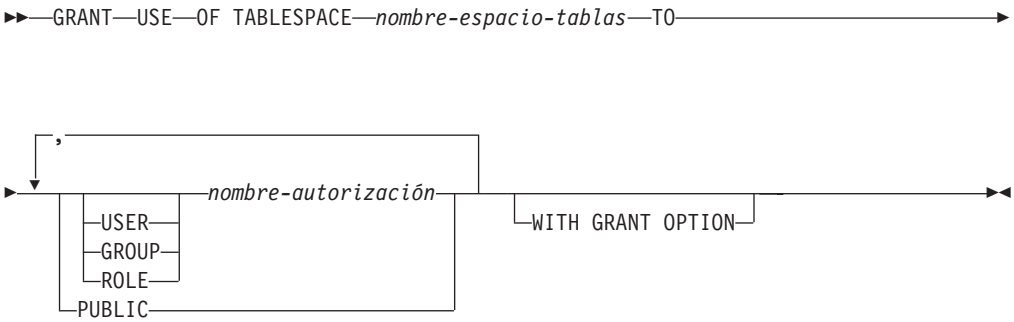
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga el privilegio USE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

WITH GRANT OPTION
Permite que el *nombre-autorización* especificado otorgue el privilegio USE a otros usuarios.

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.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.

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
```

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 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).

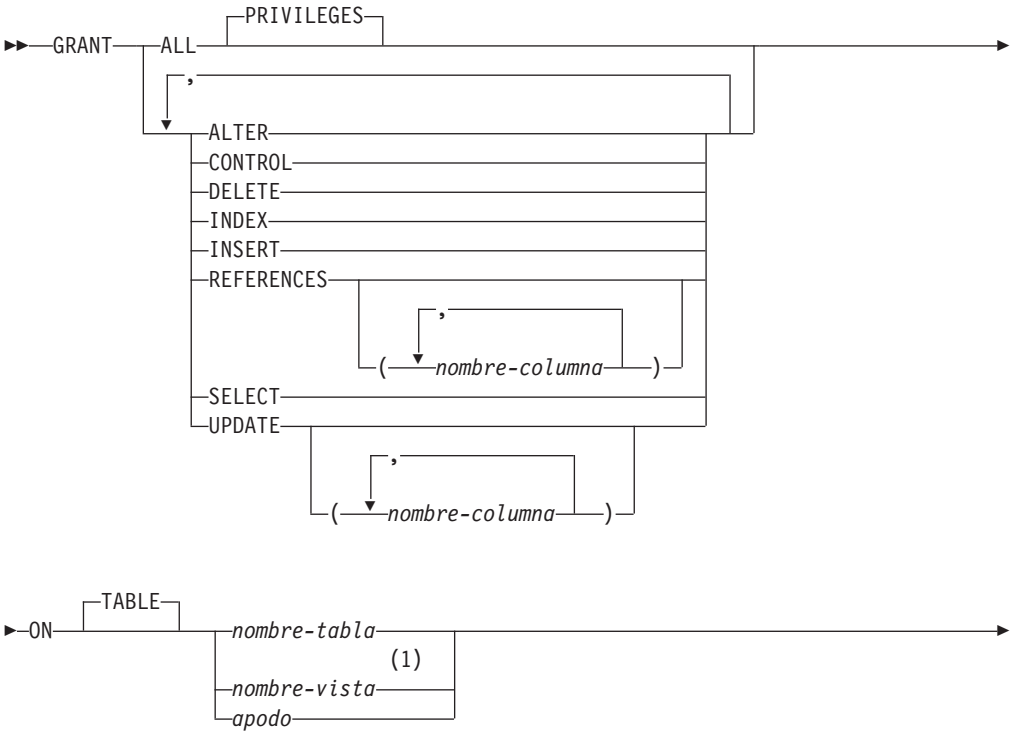
Autorización

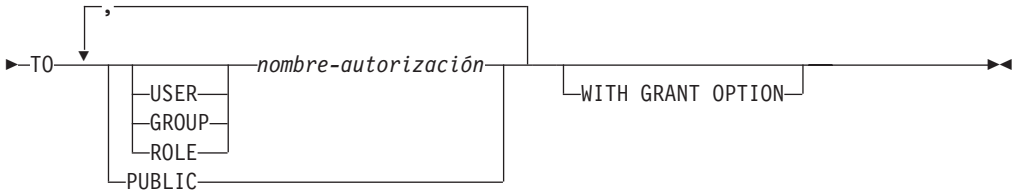
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL sobre la tabla, vista o apodo referenciado.
- WITH GRANT OPTION para cada privilegio identificado. Si se especifica ALL, el ID de autorización debe tener algún privilegio otorgable en la tabla, vista o apodo identificado.
- Autorización SYSADM o DBADM

Es necesaria la autorización DBADM o SYSADM para otorgar el privilegio CONTROL o para otorgar privilegios para tablas y vistas de catálogo.

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 sobre 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 sobre la tabla
- REFERENCES WITH GRANT OPTION para la tabla

entonces los usuarios autorizados pueden crear restricciones de referencia utilizando como clave padre todas las columnas de la tabla, incluso las que se han añadido después mediante la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son todos los privilegios REFERENCES de columna otorgables que el ID de autorización de la sentencia tiene en la tabla identificada.

El privilegio puede otorgarse para un apodo, aunque no pueden definirse claves foráneas para apodos de referencia.

REFERENCES (*nombre-columna***,...)**
Otorga el privilegio para crear y 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 sobre la tabla o vista
- UPDATE WITH GRANT OPTION en la tabla o vista

entonces la persona o personas a las que se otorga pueden actualizar todas las columnas actualizables de la tabla o vista en las que la persona que otorga tiene el privilegio así como aquellas columnas que se han añadido después utilizando la sentencia ALTER TABLE. De lo contrario, los privilegios otorgados son los privilegios UPDATE de columna otorgables que el ID de autorización de la sentencia tiene en la tabla o vista identificada.

UPDATE (*nombre-columna***,...)**
Otorga el privilegio de utilizar la sentencia UPDATE para actualizar solamente las columnas especificadas en la lista de columnas. Cada *nombre-columna* debe ser un nombre no calificado que identifique una columna de la tabla o vista identificada en la cláusula ON. El privilegio UPDATE de nivel de columna no puede otorgarse en las tablas con tipo, vistas con tipo o apodos (SQLSTATE 42997).

ON TABLE *nombre-tabla* **o** *nombre-vista* **o** *apodo*
Especifica la tabla, vista o apodo en la que se deben otorgar los privilegios.

No puede otorgarse ningún privilegio para una vista no operativa o para una tabla de 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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones. (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 Database para Linux, UNIX y Windows, los privilegios de tabla otorgados a grupos sólo se aplican a las sentencias que se preparan dinámicamente. Por ejemplo, si se ha otorgado el privilegio INSERT en la tabla PROJECT al grupo D204 pero no a UBIQUITY (un miembro de D204), UBIQUITY podría emitir la sentencia:

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

en la que el contenido de la serie es:

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

pero no podría precompilar ni enlazar un programa con la sentencia:

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

PUBLIC
Otorga los privilegios a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos". (Se han eliminado las restricciones anteriores que se aplicaban a la utilización de los privilegios que se otorgan a PUBLIC para las sentencias de SQL estático y la sentencia CREATE VIEW.)

WITH GRANT OPTION
Permite que los *nombres-autorización* especificados otorguen (GRANT) los privilegios a otros.

Si los privilegios especificados incluyen CONTROL, WITH GRANT OPTION se aplica a todos los privilegios aplicables excepto CONTROL (SQLSTATE 01516).

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente como USER y GROUP, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como USER, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* de acuerdo con el plug-in de seguridad vigente sólo como GROUP, se supone GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, se supone ROLE.
- En general, la sentencia GRANT procesará la operación de otorgar privilegios que el ID de autorización de la sentencia tenga permitido otorgar, devolviendo un aviso (SQLSTATE 01007) si no se han otorgado uno o varios privilegios. Si no se ha otorgado ningún privilegio, se devuelve un error (SQLSTATE 42501). (Si el paquete que se ha utilizado para procesar la sentencia se ha compilado previamente con LANGLEVEL establecido en SQL92E o MIA, se devolverá un mensaje de aviso (SQLSTATE 01007), a menos que el usuario que otorga privilegios no disponga de ningún privilegio para el objeto de la operación de

concesión de privilegios.) Si se 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

- 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.
- ***Compatibilidades***
  - Para mantener la compatibilidad con DB2 para z/OS:
    - Se tolera y se pasa por alto la sintaxis siguiente:
      - PUBLIC AT ALL LOCATIONS

Ejemplos

Ejemplo 1: Otorgue todos los privilegios de la tabla WESTERN_CR a PUBLIC.

```
GRANT ALL ON WESTERN_CR
   TO PUBLIC
```

Ejemplo 2: Otorgue los privilegios adecuados de la tabla CALENDAR para que los usuarios PHIL y CLAIRE puedan leerla e insertar nuevas entradas en ella. No les permita cambiar ni eliminar ninguna de las entradas existentes.

```
GRANT SELECT, INSERT ON CALENDAR
   TO USER  PHIL, USER CLAIRE
```

Ejemplo 3: Otorgue todos los privilegios de la tabla COUNCIL al usuario FRANK y la posibilidad de extender todos los privilegios a otros.

```
GRANT ALL ON COUNCIL
   TO USER FRANK WITH GRANT OPTION
```

Ejemplo 4: Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un usuario llamado JOHN. Hay un usuario llamado JOHN y no hay ningún grupo llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

o

```
GRANT SELECT
  ON CORPDATA.EMPLOYEE TO USER JOHN
```

Ejemplo 5: Otorgue (GRANT) el privilegio SELECT en la tabla CORPDATA.EMPLOYEE a un grupo llamado JOHN. Hay un grupo llamado JOHN y ningún usuario llamado JOHN.

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

o

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

Ejemplo 6: Otorgue (GRANT) los privilegios INSERT y SELECT en la tabla T1 a un grupo llamado D024 y a un usuario llamado D024.

```
GRANT INSERT, SELECT ON TABLE T1
  TO GROUP D024, USER D024
```

En este caso, tanto los miembros del grupo D024 como el usuario D024 tendrían permitido insertar (INSERT) y seleccionar (SELECT) en la tabla T1. También, se añadirían dos filas a la vista de catálogo SYSCAT.TABAUTH.

Ejemplo 7: Otorgue (GRANT) INSERT, SELECT y CONTROL en la tabla CALENDAR al usuario FRANK. FRANK debe poder pasar los privilegios a otros.

```
GRANT 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 mediante la creación de una especificación de índice para ORAREM1. Otorgue a SHAWN el privilegio para este apodo, de modo que podrá crear la especificación de índice.

```
GRANT INDEX ON NICKNAME ORAREM1
  TO USER SHAWN
```

GRANT (privilegios de carga de trabajo)

Este formato de la sentencia GRANT otorga el privilegio USAGE en una carga de trabajo.

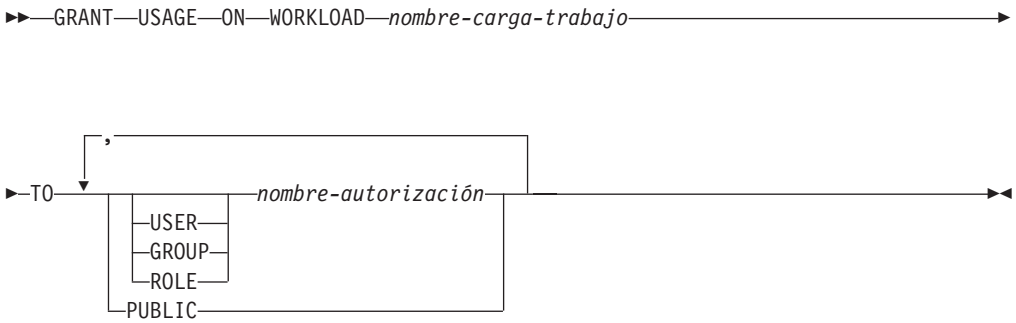
Invocación

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

Autorización

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

Sintaxis



Descripción

USAGE
Otorga el privilegio para utilizar una carga de trabajo. Las unidades de trabajo que somete un usuario sólo se correlacionarán con una carga de trabajo en la que el usuario tenga un privilegio USAGE. Un usuario con autorización SYSADM o DBADM tiene automáticamente un privilegio USAGE sobre cualquier carga de trabajo que exista en el servidor actual.

ON WORKLOAD *nombre-carga-trabajo*
Identifica la carga de trabajo a la que va a otorgarse el privilegio USAGE. Este nombre consta de una sola parte. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El nombre no puede ser 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

TO
Especifica a quién se otorga el privilegio USAGE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

ROLE
Especifica que el *nombre-autorización* identifica una función existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Otorga el privilegio USAGE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte "Autorizaciones, privilegios y propiedad de objetos".

Normas

- Por cada *nombre-autorización* especificado, si no se especifica ninguna de las palabras clave USER, GROUP o ROLE:
  - Si el plug-in de seguridad vigente para la instancia no puede determinar el estado del *nombre-autorización*, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como ROLE en la base de datos y como GROUP o USER en el sistema operativo, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* como USER y GROUP de acuerdo con el plug-in de seguridad vigente, se devuelve un error (SQLSTATE 56092).
  - Si se define el *nombre-autorización* sólo como USER de acuerdo con el plug-in de seguridad vigente, o si no está definido, se supone USER.
  - Si se define el *nombre-autorización* sólo como GROUP de acuerdo con el plug-in de seguridad vigente, se asume GROUP.
  - Si se define el *nombre-autorización* en la base de datos sólo como ROLE, entonces se supone ROLE.

Notas

- La sentencia GRANT no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.
- Si la base de datos se crea con la opción RESTRICT, el privilegio USAGE de la carga de trabajo de usuario por omisión, SYSDEFAULTUSERWORKLOAD, debe otorgarse de modo explícito por medio de un usuario que tenga autorización DBADM. Si la base de datos se crea sin la opción RESTRICT, el privilegio USAGE de SYSDEFAULTUSERWORKLOAD se otorgará a PUBLIC en el momento de creación de la base de datos.

Ejemplo

Otorgar al usuario LISA la capacidad de utilizar la carga de trabajo CAMPAIGN.

```
GRANT USAGE ON WORKLOAD CAMPAIGN TO USER LISA
```

GRANT (privilegios de objeto XSR)

Este formato de la sentencia GRANT otorga el privilegio USAGE sobre un objeto XSR.

Invocación

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

Autorización

Se necesita una de las autorizaciones siguientes:

- SYSADM o DBADM
- Propiedad del objeto XSR

Sintaxis

Descripción

ON XSROBJECT *nombre-objetoxsr*
Este nombre identifica el objeto XSR sobre el que se ha otorgado el privilegio USAGE. El *nombre-objetoxsr* (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este nombre, se devuelve un error (SQLSTATE 42704).

TO PUBLIC
Otorga el privilegio USAGE a un conjunto de usuarios (ID de autorización). Para obtener más información, consulte el apartado "Autorizaciones, privilegios y propiedad de objetos".

Ejemplo

Otorgar a todos los usuarios privilegio de uso sobre el esquema XML MYSCHEMA:

```
GRANT USAGE ON XSROBJECT MYSCHEMA TO PUBLIC
```

IF

La sentencia IF selecciona una vía de ejecución de acuerdo con la evaluación de una condición.

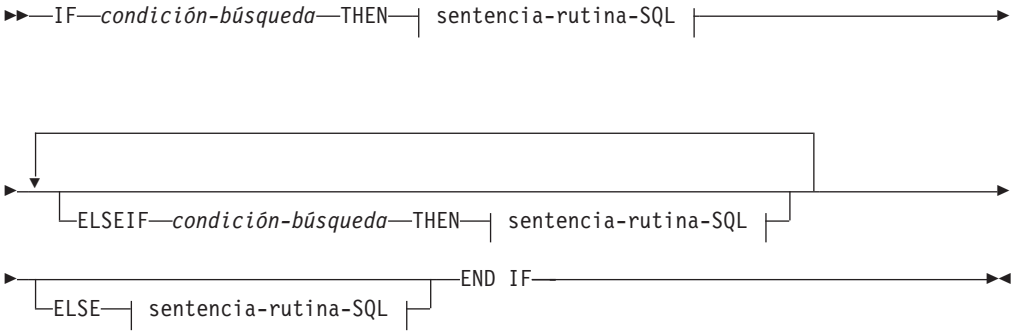
Invocación

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.

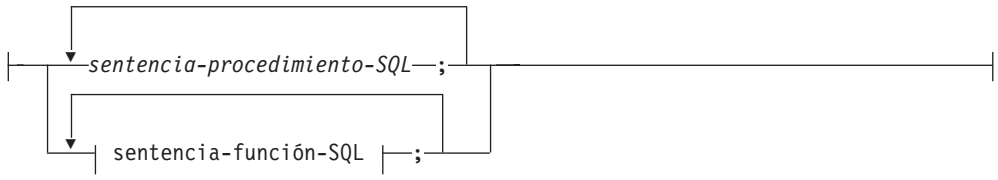
Autorización

No se requieren privilegios para invocar la sentencia IF. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL y 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 *sentencia-procedimiento-SQL* en "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 "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
```

INCLUDE

La sentencia INCLUDE inserta declaraciones en un programa fuente.

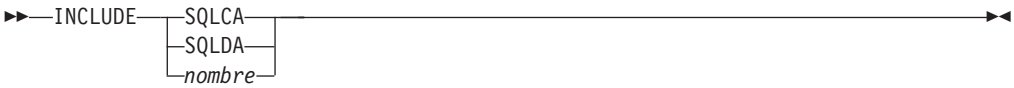
Invocación

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 del sistema principal especificado por *nombre*. Si es superior a 18 bytes o contiene caracteres que no están permitidos en un identificador SQL, debe ir entre comillas simples. Las sentencias INCLUDE *nombre* pueden anidarse, aunque no cíclicamente (por ejemplo, si A y B son módulos y A contiene una sentencia INCLUDE *nombre*, no es válido que A llame a B y que, a continuación, B llame a A).
- Si la opción de precompilación LANGLEVEL se establece en el valor SQL92E, no debe especificarse INCLUDE SQLCA. Las variables SQLSTATE y SQLCODE pueden estar definidas en la sección de declaraciones de variables de lenguaje principal.

Ejemplo

Incluya una SQLCA en un programa C.

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL  DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

EXEC SQL  OPEN C1;

mientras (SQLCODE==0) {
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

(Imprimir resultados)

}

EXEC SQL  CLOSE C1;
```

INSERT

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 a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

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

- Privilegio INSERT para la tabla, vista o apodo de destino
- Privilegio CONTROL sobre la tabla, vista o apodo de destino
- Autorización SYSADM o DBADM

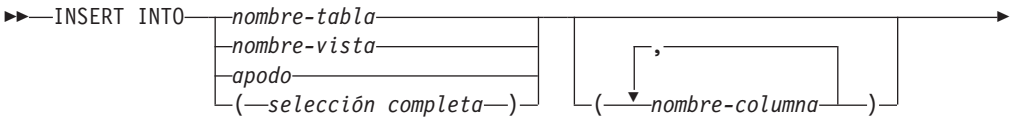
Además, para cada tabla, vista o apodo al que se haga referencia en cualquier selección completa utilizada en la sentencia INSERT, los privilegios del ID de autorización de la sentencia deben incluir, como mínimo, uno de los siguientes:

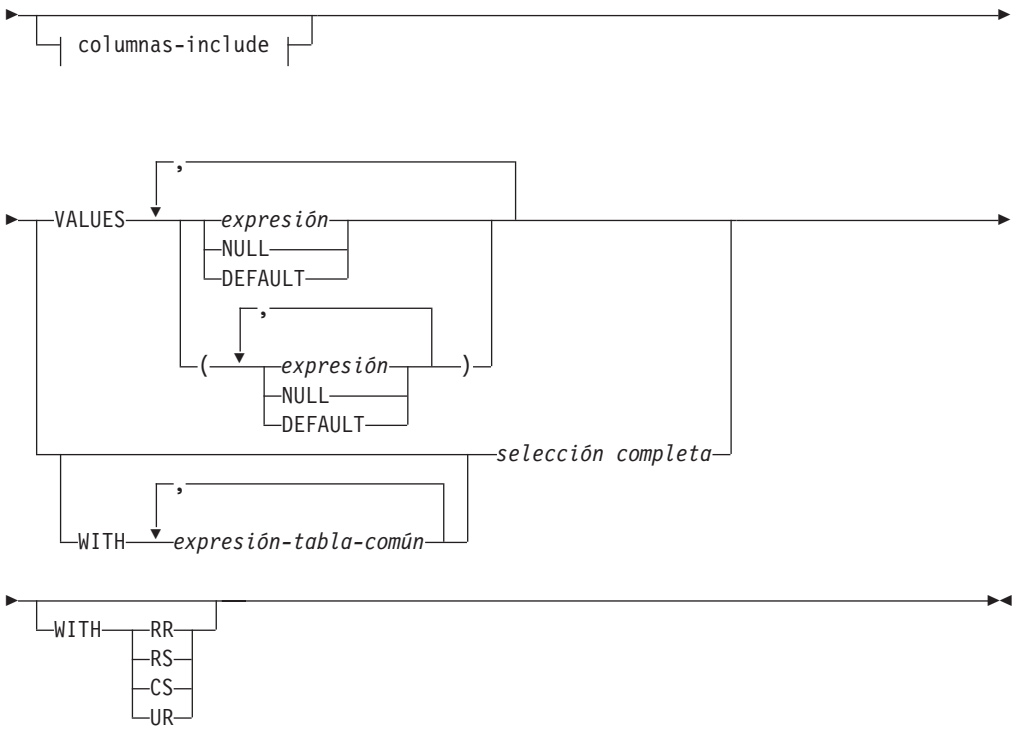
- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM

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 esté definido un activador INSTEAD OF para la operación de inserción de 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 identificar una columna de la tabla, vista o apodo especificado, o una columna de la selección completa. La misma columna no se puede identificar más de una vez. No puede identificarse una columna que no pueda aceptar la inserción de valores (por ejemplo, una columna basada en una expresión).

La omisión de la lista de columnas es una especificación implícita de una lista en la que cada columna de la tabla (que no está oculta implícitamente) o vista, o cada elemento de la lista de selección de la selección completa se identifica en orden de izquierda a derecha. Esta lista se establece cuando se prepara la sentencia y, por lo tanto, no incluye las columnas que se han añadido a la tabla después de preparar la sentencia.

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 ROW CHANGE TIMESTAMP, el gestor de bases de datos genera el valor para cada fila insertada como una indicación de fecha y hora que es exclusiva para cada partición de tabla en la partición de base de datos.
- Si se utiliza la cláusula WITH DEFAULT, el valor insertado será el valor tal como se ha definido para la columna (consulte *cláusula-por-omisión* en "CREATE TABLE").
- Si se utiliza la cláusula NOT NULL y no la cláusula GENERATED, o no se utiliza WITH DEFAULT o se utiliza DEFAULT NULL, no se puede especificar la palabra clave DEFAULT para esa columna (SQLSTATE 23502).
- Cuando la inserción se realice en un apodo, la palabra clave DEFAULT se pasará por medio de la sentencia INSERT a la fuente de datos sólo si la fuente de datos da soporte a la palabra clave DEFAULT en su sintaxis de lenguaje de consulta.

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. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

Normas

- ***Activadores:*** Las sentencias INSERT pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores insertados. Si una operación de inserción en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se han realizado en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.
- ***Valores por omisión:*** El valor insertado en cualquier columna que no esté en la lista de columnas es el valor por omisión de la columna o nulo. Las columnas que no permiten valores nulos y no están definidas con NOT NULL WITH DEFAULT deben incluirse en la lista de columnas. De manera similar, si se inserta en una vista, el valor insertado en cualquier columna de la tabla base que no esté en la vista es el valor por omisión de la columna o nulo. De ahí que todas las columnas de la tabla base que no estén en la vista deben ser un valor por omisión o permitir valores nulos. El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).
- ***Longitud:*** Si el valor de inserción de una columna es un número, la columna debe ser una columna numérica con la capacidad de representar la parte entera del número. Si el valor de inserción de una columna es una serie, la columna debe ser una columna de serie con un atributo de longitud que como mínimo sea tan grande como la longitud de la serie, o una columna de indicación de fecha y hora si la serie representa una fecha, hora o indicación de fecha y hora.
- ***Asignación:*** Los valores de inserción se asignan a las columnas de acuerdo con normas de asignación específicas.
- ***Validez:*** Si la tabla nombrada, o la tabla base de la vista nombrada, tiene uno o varios índices de unicidad, cada fila insertada en la tabla debe ajustarse a las restricciones impuestas por dichos índices. Si se nombra una vista cuya definición incluye WITH CHECK OPTION, cada fila insertada en la vista debe ajustarse a la definición de la vista. Para obtener información acerca de las normas que rigen esta situación, consulte “CREATE VIEW”.
- ***Integridad de referencia:*** Para cada restricción definida en una tabla, cada valor de inserción que no sea nulo de la clave foránea debe ser igual al valor de clave primaria de la tabla padre.
- ***Restricción de comprobación:*** Los valores de inserción deben cumplir las condiciones de control de las restricciones de comprobación definidas en la tabla. En una sentencia INSERT para una tabla con restricciones de comprobación definidas, se evalúan las condiciones de restricción una vez para cada fila que se inserta.
- ***Valores XML:*** Un valor que se inserta en una columna XML debe ser un documento XML con el formato correcto (SQLSTATE 2200M).

- ***Política de seguridad:*** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
  - Acceso de grabación a todas las columnas protegidas para las que se ha proporcionado explícitamente un valor de datos (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)

  El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:
  - No se proporciona explícitamente un valor para la columna DB2SECURITYLABEL
  - Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

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 insertar 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.
- Al realizar una inserción en una columna GENERATED ALWAYS ROW CHANGE TIMESTAMP, DB2 genera siempre un valor para la columna y los usuarios no deben especificar un valor en el momento de realizar la inserción (SQLSTATE 428C9). El valor generado por DB2 es exclusivo para cada fila insertada en la partición de base de datos.
- Cuando se inserta en una columna GENERATED BY DEFAULT, DB2 permite especificar un valor real para la columna dentro de la cláusula VALUES o en una subselección. Sin embargo, cuando se especifica un valor en la cláusula VALUES, DB2 no realiza ninguna verificación del valor. Para asegurar la unicidad de los valores de la columna IDENTITY, se debe crear un índice de unicidad en la columna de identidad.

  Cuando se inserta en una tabla que tiene una columna de identidad definida como GENERATED BY DEFAULT, y no se especifica una lista de columnas, la cláusula VALUES puede especificar la palabra clave DEFAULT para representar el valor de la columna de identidad. DB2 generará el valor para la columna de identidad.

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
  VALUES (DEFAULT, :hv_name, :hv_addr)
```

  En este ejemplo, EMPID se define como una columna de identidad y, por tanto, el valor insertado en esta columna es generado por DB2.
- Las normas para insertar en una columna de identidad utilizando una subselección son similares a las normas para una inserción mediante una cláusula VALUES. Sólo se puede especificar un valor para una columna de identidad si 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 la columna está definida como columna de identidad GENERATED ALWAYS o como una columna ROW CHANGE TIMESTAMP, es posible especificar una cláusula VALUES con la palabra clave DEFAULT. En este caso, la aplicación no proporciona ningún valor para la tabla y DB2 genera el valor para la columna de identidad o ROW CHANGE TIMESTAMP.

  ```
  INSERT INTO IDTABLE
    VALUES(DEFAULT)
  ```

  En la tabla del ejemplo anterior, formada por una sola columna que tiene el atributo de identidad, para insertar varias filas con una única sentencia INSERT puede utilizarse esta sentencia:

  ```
  INSERT INTO IDTABLE
    VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
  ```

- Cuando DB2 genera un valor para una columna de identidad, ese valor generado caduca; la próxima vez que sea necesario un valor, DB2 generará uno nuevo. Esto es válido aunque falle o se cancele una sentencia INSERT en la que interviene una columna de identidad.

  Por ejemplo, suponga que se ha creado un índice de unicidad para la columna de identidad. Si al generar un valor para una columna de identidad se detecta una violación de clave duplicada, se produce un error (SQLSTATE 23505) y se considera que el valor generado para la columna de identidad ha caducado. Esto puede ocurrir si la columna de identidad está definida como GENERATED BY DEFAULT y el sistema intenta generar un nuevo valor, pero el usuario ha especificado explícitamente valores para la columna de identidad en sentencias INSERT anteriores. En este caso, el volver a emitir la misma sentencia INSERT puede producir un resultado satisfactorio. DB2 generará el valor siguiente para la columna de identidad y es posible que este valor siguiente sea exclusivo, y que la sentencia INSERT tenga éxito.

- Si al generar un valor para una columna de identidad se excede el valor máximo de la columna (o el valor mínimo en el caso de una secuencia descendente), se

produce un error (SQLSTATE 23522). En este caso, el usuario debe 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;
```

Ejemplo 7: Utilice una sentencia del programa C para añadir un documento a la tabla DOCUMENTS. Obtenga valores para la columna ID del documento (DOCID) y la columna de datos del documento (XMLDOC) desde una variable del lenguaje principal que se vincula con un SQL TYPE IS XML AS BLOB_FILE.

```
EXEC SQL INSERT INTO DOCUMENTS
  (DOCID, XMLDOC) VALUES (:docid, :xmldoc)
```

Ejemplo 8: para las siguientes sentencias INSERT, suponga que la tabla SALARY_INFO está definida con tres columnas y que la última columna es una columna ROW CHANGE TIMESTAMP implícitamente oculta. En la sentencia siguiente, se hace referencia explícita a la columna implícitamente oculta en la lista de columnas y se proporciona un valor para dicha columna en la cláusula VALUES.

```
INSERT INTO SALARY_INFO (LEVEL, SALARY, UPDATE_TIME)
  VALUES (2, 30000, CURRENT TIMESTAMP)
```

La sentencia siguiente INSERT utiliza una lista de columnas implícitas. Una lista de columnas implícitas no incluye columnas implícitamente ocultas, de manera que la cláusula VALUES sólo contiene valores para las otras dos columnas.

```
INSERT INTO SALARY_INFO VALUES (2, 30000)
```

En este caso, la columna UPDATE_TIME debe definirse para que tenga un valor por omisión y el valor por omisión se utiliza para la fila que se inserta.

ITERATE

La sentencia ITERATE hace que el flujo de control vuelva al principio de un bucle con etiqueta.

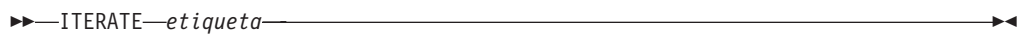
Invocación

Esta sentencia 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
```

LOCK TABLE

La sentencia LOCK TABLE impide que procesos de aplicación simultáneos utilicen o cambien una tabla.

Invocación

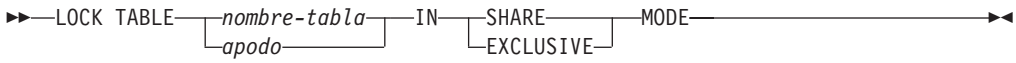
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

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

- Privilegio SELECT en la tabla
- Privilegio CONTROL sobre 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 o una tabla temporal declarada (SQLSTATE 42995). Si *nombre-tabla* es una tabla con tipo, debe ser la tabla raíz de la jerarquía de tablas (SQLSTATE 428DR). Cuando se ha especificado un apodo, DB2 bloquea el objeto subyacente (es decir, una tabla o una vista) de la fuente de datos a la que el apodo hace referencia.

IN SHARE MODE
Impide que procesos de aplicación simultáneos ejecuten alguna operación que no sea de sólo lectura en la tabla.

IN EXCLUSIVE MODE
Impide a los procesos de aplicación simultáneos ejecutar cualquier operación en la tabla. Tenga en cuenta que EXCLUSIVE MODE no impide que los procesos de aplicación simultáneos que estén ejecutando en el nivel de aislamiento Lectura no confirmada (UR) ejecuten operaciones de sólo lectura en la tabla.

Notas

- El bloqueo se utiliza para evitar operaciones simultáneas. No se adquiere necesariamente un bloqueo durante la ejecución de la sentencia LOCK TABLE si ya existe un bloqueo satisfactorio. El bloqueo que impide operaciones simultáneas se conserva como mínimo hasta la terminación de la unidad de trabajo.
- En una base de datos particionada, primero se adquiere un bloqueo de tabla en la primera partición de base de datos del grupo de particiones de base de datos (la partición de base de datos que tiene el número más bajo) y, después, en las

demás particiones de base de datos. Si se interrumpe la sentencia LOCK TABLE, la tabla puede estar bloqueada en algunas particiones de base de datos y en otras no. Si ocurre esto, emita otra sentencia LOCK TABLE para completar el bloqueo de todas las particiones de base de datos o emita una sentencia COMMIT o ROLLBACK para liberar los bloqueos actuales.

- Esta sentencia afecta a todas las particiones de base de datos del grupo de particiones de base de datos.
- Para tablas particionadas, el único bloqueo adquirido en la sentencia LOCK TABLE se encuentra en el nivel de la tabla; no se han adquirido bloqueos de particiones de datos.

Ejemplo

Obtenga un bloqueo en la tabla EMP. No permita que otros programas lean o actualicen la tabla.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

LOOP

La sentencia LOOP repite la ejecución de una sentencia o grupo de sentencias.

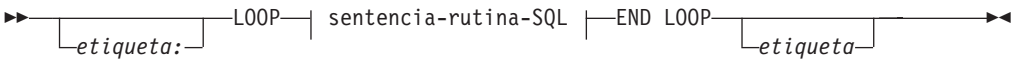
Invocación

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

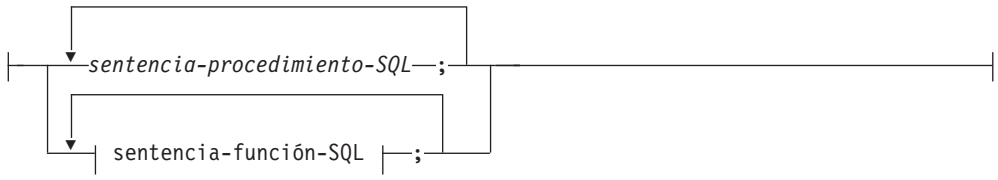
Autorización

No se requieren privilegios para invocar la sentencia LOOP. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL incorporadas en la sentencia LOOP.

Sintaxis



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 *sentencia-procedimiento-SQL* en "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 "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
```

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 a 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:

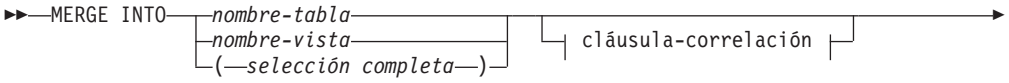
- Privilegio SELECT en cada tabla o vista identificada en la *referencia-tabla*
- Privilegio CONTROL en las tablas o vistas identificadas en la *referencia-tabla*
- Autorización SYSADM o DBADM

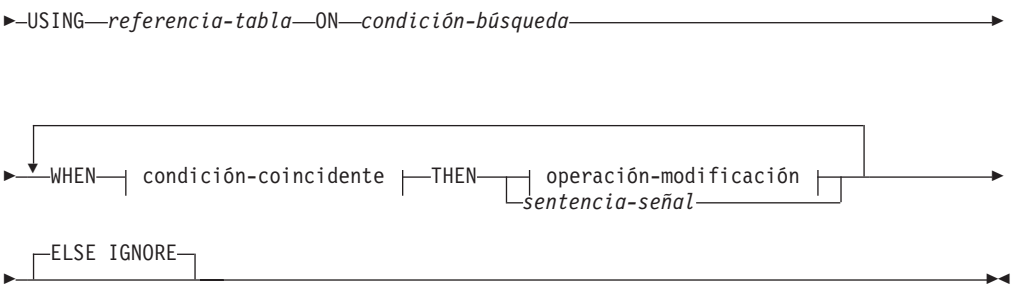
Si *condición-búsqueda*, *operación-inserción* o *cláusula-asignación* incluye una subconsulta, el ID de autorización de la sentencia también debe tener al menos uno de los privilegios siguientes:

- 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

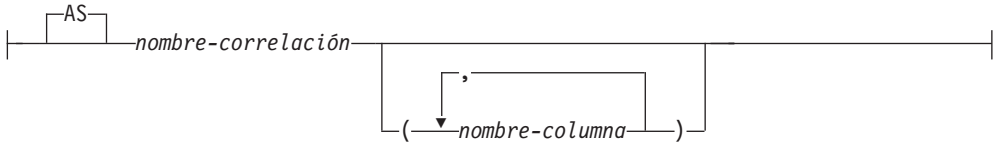
Si se especifica una expresión que se refiere a una función, el conjunto de privilegios debe incluir la autoridad que sea necesaria para ejecutar la función.

Sintaxis

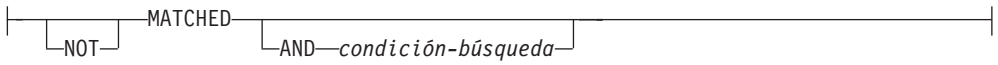




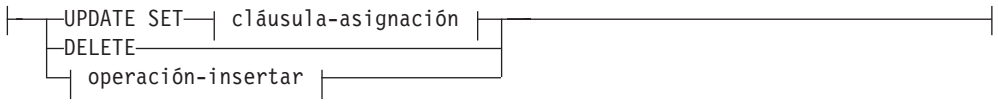
cláusula-correlación:



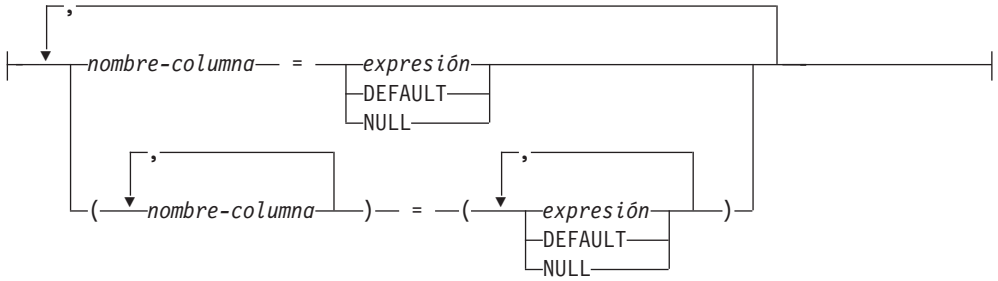
condición-coincidente:



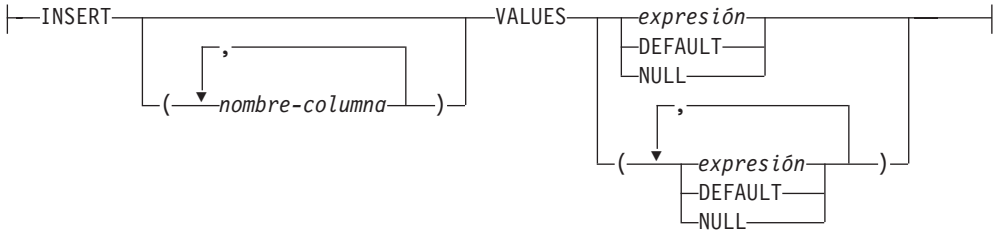
operación-modificación:



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.

No se puede utilizar un apodo (una referencia a una tabla federada remota) como tabla de destino.

cláusula-correlación
Se puede utilizar en *condición-búsqueda* o a la derecha de una *cláusula-asignación* para designar una tabla, vista o selección completa. Para ver una descripción de la *cláusula-correlación*, consulte "referencia-tabla" en la descripción de "Subselección".

USING *referencia-tabla*
Especifica un conjunto de filas como una tabla de resultados para fusionarse en el destino. Si la tabla de resultados está vacía, se devuelve un aviso (SQLSTATE 02000).

ON *condición-búsqueda*
Especifica las filas de *referencia-tabla* que se utilizarán en la operación de actualización y supresión de la fusión, y las filas que se utilizarán en la operación de inserción de la fusión. *condición-búsqueda* se aplica a cada fila de la tabla de destino y tabla de resultados de *referencia-tabla*. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* sea verdadero, se realiza la operación de actualización o supresión especificada. Para aquellas filas de la tabla de resultados de la *referencia-tabla* donde el resultado de la *condición-búsqueda* no sea verdadero, se realiza la operación de inserción especificada.

La *condición-búsqueda* tiene las restricciones siguientes (SQLSTATE 42972):

- No puede contener ninguna subconsulta, escalar ni de cualquier otra clase
- No puede incluir ninguna operación de desreferencia ni la función DEREF, donde el valor de referencia es distinto de la columna de identificadores de objeto
- No puede incluir una función SQL
- No puede incluir una expresión XMLQUERY ni XMLEXISTS
- Cualquier columna a la que se hace referencia en una expresión de la *condición-búsqueda* debe ser una columna de la tabla, vista o *referencia-tabla* de destino.
- Cualquier función a la que se haga referencia en una expresión de la *condición-unión* de una unión externa completa debe ser determinista y no debe tener ninguna acción externa

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 de una especificación implícita de una lista en la que cada columna de la tabla (que no está definida como implícitamente oculta) o vista se identifica en orden de izquierda a derecha. Esta lista se establece cuando se prepara la sentencia, y, por lo tanto, no incluye las columnas que se han añadido a la tabla después de preparar la sentencia.

VALUES
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).
- ***Política de seguridad*** si se protege la tabla de destino identificada o la tabla base de la vista de destino identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten los siguientes tipos de acceso.
  - Para la operación de actualización:
    - Acceso de grabación a todas las columnas protegidas que se están actualizando (SQLSTATE 42512)
    - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)
    - Acceso de grabación y de lectura a todas las filas que se están actualizando (SQLSTATE 42519)

    El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:
    - La columna DB2SECURITYLABEL no está incluida en la lista de columnas que se van a actualizar (y, por lo tanto, se actualizará implícitamente en la etiqueta de seguridad para el acceso de grabación del ID de autorización de la sesión)
    - Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
  - Para la operación de supresión:
    - Acceso de grabación a todas las columnas protegidas (SQLSTATE 42512)

- Acceso de grabación y de lectura a todas las filas que se han seleccionado para la supresión (SQLSTATE 42519)

– Para la operación de inserción:
  - Acceso de grabación a todas las columnas protegidas para las que se ha proporcionado explícitamente un valor de datos (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)

  El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:
  - No se proporciona explícitamente un valor para la columna DB2SECURITYLABEL
  - Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

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 las actividades (descripción y fecha) 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 ' no se puede modificar. Razón: No se conoce la fecha'
WHEN MATCHED AND ac.date < CURRENT DATE THEN
```

```
   DELETE
WHEN MATCHED AND ar.last_modified < ac.last_modified THEN
   UPDATE SET
      (description, date, last_modified) = (ac.description, ac.date, DEFAULT)
WHEN NOT MATCHED AND ac.date IS NULL THEN
   SIGNAL SQLSTATE '70002'
      SET MESSAGE_TEXT =
         ac.activity CONCAT ' no se puede insertar. Razón: No se conoce la fecha'
WHEN NOT MATCHED AND ac.date >= CURRENT DATE THEN
   INSERT
      (group, activity, description, date)
      VALUES ('A', ac.activity, ac.description, ac.date)
ELSE IGNORE
```

OPEN

La sentencia OPEN abre un cursor para que pueda utilizarse para leer filas de la tabla de resultados.

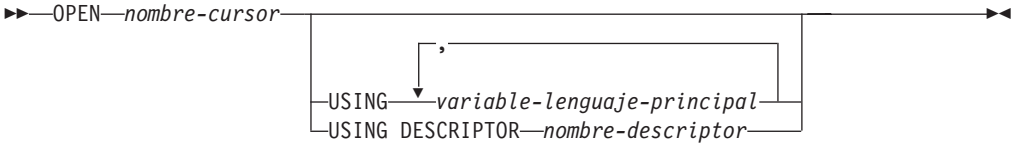
Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Se trata de 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, variables globales o expresiones PREVIOUS VALUE que se han especificado en la sentencia SELECT y los valores actuales de cualquier variable del lenguaje principal que se han especificado en la sentencia SELECT o en la cláusula USING de la sentencia OPEN. Las filas de la tabla de resultados pueden obtenerse durante la ejecución de la sentencia OPEN, y puede crearse una tabla temporal para contenerlas; o bien pueden obtenerse durante la ejecución de sentencias FETCH posteriores. En cualquier caso, el cursor se coloca en el estado abierto y se posiciona antes de la primera fila de su tabla de resultados. Si la tabla está vacía, el estado del cursor está, efectivamente, "después de la última fila".

USING
Introduce una lista de variables del lenguaje principal cuyos valores se sustituyen por los marcadores de parámetro (signos de interrogación) de una sentencia preparada. Si la sentencia DECLARE CURSOR nombra una sentencia

preparada que incluye marcadores de parámetro, debe utilizarse USING. Si la sentencia preparada no incluye ningún marcador de parámetros, se ignora USING.

variable-lenguaje-principal
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 parámetro de la sentencia preparada. La variable *n* corresponde al marcador de parámetro *n* de la sentencia preparada. Cuando sea adecuado, pueden proporcionarse variables localizadoras y variables de referencia a archivos como fuente de valores para marcadores de parámetro.

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 apariciones 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 aparición 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 parámetro. 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 parámetro 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 parámetro de la cláusula WHERE no se conocen.

```
EXEC SQL  BEGIN DECLARE SECTION;
  char    stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

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.

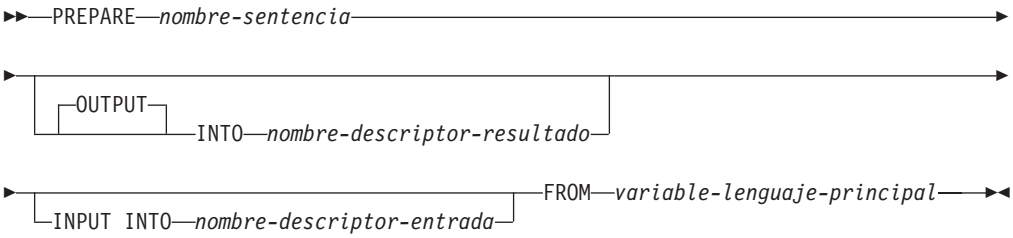
Autorización

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 podría verse afectado por la opción de enlace DYNAMICRULES.

En el caso de las sentencias en las que el control de autorizaciones se realiza durante la ejecución de la sentencia (sentencias DDL, GRANT y REVOKE), no es necesaria ninguna autorización para utilizar esta sentencia; no obstante, se comprueba la autorización cuando se ejecuta la sentencia preparada.

Para las sentencias que afectan a las tablas protegidas con una política de seguridad, las normas asociadas con la política de seguridad siempre se evaluarán durante el tiempo de ejecución de la sentencia.

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-lenguaje-principal
Especifica una variable del lenguaje principal que se ha descrito en el programa de acuerdo con las normas para la declaración de variables de serie de caracteres. Debe ser una variable de serie de caracteres de longitud que puede variar cuyo tamaño sea menor que el tamaño de sentencia máximo de 2.097.152 bytes. Tenga en cuenta que un CLOB(2097152) puede contener una sentencia de tamaño máximo, pero un VARCHAR no puede.

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
- SET variable
- UPDATE

Notas

- ***Marcadores de parámetro:*** Aunque una serie de sentencia no puede incluir referencias a variables de lenguaje principal, puede incluir *marcadores de parámetro*. Estos pueden sustituirse por los valores de las variables del lenguaje principal cuando se ejecuta la sentencia preparada. En el caso de una sentencia CALL, también puede utilizarse un marcador de parámetro para los argumentos OUT e INOUT para el procedimiento. Tras haberse ejecutado CALL, el valor que se ha devuelto para el argumento se asignará a la variable del lenguaje principal que corresponde al marcador de parámetro.

  Un marcador de parámetros es un signo de interrogación (?) 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 una explicación sobre cómo se sustituyen los marcadores de parámetro por valores, consulte “OPEN” y “EXECUTE”.

  Existen dos tipos de marcadores de parámetro:

  Marcador de parámetro con tipo
  Es un marcador de parámetros que se especifica junto con su tipo de datos de destino. Tiene el formato general:

  ```
  CAST(? AS tipo-datos)
  ```

  Esta notación no es una llamada a función, sino una “promesa” de que el tipo del parámetro en tiempo de ejecución será el tipo de datos especificado o algún otro tipo de datos que pueda convertirse 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 parámetro 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 parámetro con tipo pueden utilizarse en sentencias de SQL dinámico donde esté soportada una variable del lenguaje principal y el tipo de datos se base en la promesa realizada en la función CAST.

Los marcadores de parámetro sin tipo 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 las tablas siguientes. Las ubicaciones se agrupan 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 29. Uso de marcadores de parámetro sin tipo en expresiones (incluidas Select List, CASE y VALUES)

Ubicación del marcador de parámetros sin tipo	**Tipo de datos**
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 fecha y hora (por ejemplo, 'timecol + ?' o '? - datecol')	Error
Ambos operandos de un operador CONCAT	Error
Un operando de un operador CONCAT cuando el otro operando es un tipo de datos de caracteres que no es CLOB	Si un operando es CHAR(*n*) o VARCHAR(*n*), donde *n* es menor que 128, el otro es VARCHAR(254 - *n*); en el resto de casos, el tipo de datos es VARCHAR(254)

Tabla 29. Uso de marcadores de parámetro sin tipo en expresiones (incluidas Select List, CASE y VALUES) (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Un operando de un operador CONCAT cuando el otro operando es un tipo de datos gráfico que no es DBCLOB	Si un operando es GRAPHIC(*n*) o VARGRAPHIC(*n*), donde *n* es menor que 64, el otro es VARGRAPHIC(127 - *n*); en el resto de los casos, el tipo de datos es VARGRAPHIC(127)
Un operando de un operador CONCAT cuando el otro operando es una serie de gran objeto	Igual que el del otro operando
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 estructurado 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 de una expresión CASE (tanto simple como con búsqueda), con el resto de las expresiones-resultado que sean marcadores de parámetro sin tipo o NULL	Error
Cualquiera o todas las expresiones que siguen a WHEN en una expresión CASE simple	El resultado de la aplicación de las “Normas para los tipos de datos de resultado” 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 tipos de datos de resultados” a todas las expresiones-resultado 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 parámetro 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 tipos de datos de resultados” en todos los operandos que no son marcadores de parámetro sin tipo

Tabla 29. Uso de marcadores de parámetro sin tipo en expresiones (incluidas Select List, CASE y VALUES) (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Solo como una expresión-columna en una cláusula VALUES de una sola fila dentro de una sentencia INSERT	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Solo como una expresión-columna en una cláusula VALUES de múltiples filas dentro de una sentencia INSERT	El tipo de datos de la columna. Si la columna se define como un tipo diferenciado definido por el usuario, es el tipo de datos fuente del tipo diferenciado definido por el usuario. Si la columna está definida como tipo estructurado definido por el usuario, es el tipo de datos estructurado y también indica el tipo devuelto de la función de transformación.
Como un valor a la derecha de una sentencia SET de registro especial	El tipo de datos del registro especial

Tabla 30. Uso del marcador de parámetros sin tipo en predicados

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Ambos operandos de un operador de comparación	Error
Un operando de un operador de comparación cuando 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
El • primer y el segundo o el • primer y el tercer 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 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 de tipo
El primer operando de un predicado IN donde el lado derecho es una selección completa	El tipo de datos de la columna seleccionada

Tabla 30. Uso del marcador de parámetros sin tipo en predicados (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Cualquiera o todos los operandos de la lista IN del predicado IN	El resultado de aplicar las "normas para los tipos de datos de resultado" en todos los operandos en el predicado IN (los operandos que se encuentran a la izquierda y a la derecha del predicado IN) que no son marcadores de parámetro 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
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) 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(1)
Operando del predicado NULL	Error

Tabla 31. Uso del marcador de parámetros sin tipo en funciones incorporadas

Ubicación del marcador de parámetros sin tipo	Tipo de datos
Todos los operandos de COALESCE (también llamados VALUE) o NULLIF	Error
Cualquier operando de COALESCE o NULLIF cuando al menos el primer operando o es un marcador de parámetros sin tipo	El resultado de aplicar las "normas para 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 cuando el otro operando es de un tipo de datos de caracteres).	VARCHAR(32672)
POSSTR (un operando cuando 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 (primer operando)	VARCHAR(32672)
SUBSTR (segundo y tercer operandos)	INTEGER

Tabla 31. Uso del marcador de parámetros sin tipo en funciones incorporadas (continuación)

Ubicación del marcador de parámetros sin tipo	Tipo de datos
El primer operando de la función escalar TRANSLATE	Error
El segundo y el tercer operandos de la función escalar TRANSLATE	VARCHAR(32672) si el primer operando es un tipo de carácter; VARGRAPHIC(16336) si el primer operando es un tipo gráfico
El cuarto operando de la función escalar TRANSLATE	VoARCHAR(1) si el primer operando es un tipo de carácter; VARGRAPHIC(1) si el primer operando es un tipo gráfico
El segundo operando de la función escalar TIMESTAMP	TIME
Menos unitario	DOUBLE
Más unitario	DOUBLE
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)
Primer operando de la función XMLVALIDATE	XML
Todos los demás operandos de todas las demás funciones escalares	Error
Operando de una función de columna	Error

Tabla 32. Uso del marcador de parámetros sin tipo en rutinas definidas por el usuario

Ubicación del marcador de parámetros sin tipo	Tipo de datos
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

- 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 parámetro, 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 parámetro.

```
EXEC SQL  PREPARE STMT_NAME FROM :holder;
EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

Suponga que se debe preparar la sentencia siguiente:

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

Las columnas de la tabla DEPT se definen de la siguiente manera:

```
DEPT_NO   CHAR(3) NOT NULL, -- número de departamento
DEPTNAME  VARCHAR(29), -- nombre del departamento
MGRNO     CHAR(6), -- número del director
ADMRDEPT  CHAR(3)  -- número del departamento de administración
```

Para insertar el número de departamento G01 que se denomina COMPLAINTS, que no tiene ningún director y que informa al departamento A00, la cláusula INSERT_DA debe tener los valores de la Tabla 33 antes de emitirse la sentencia EXECUTE.

Tabla 33.

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	

Tabla 33. (continuación)

Campo de SQLDA	**Valor**
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	

Nota:

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.

REFRESH TABLE

La sentencia REFRESH TABLE renueva los datos de una tabla de consultas materializadas.

Invocación

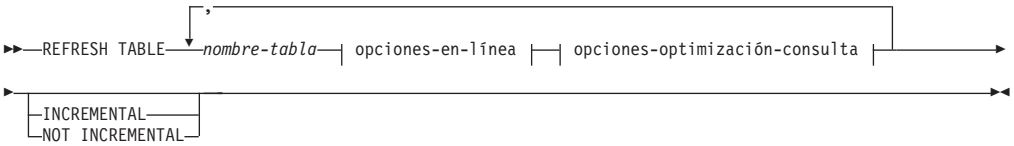
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

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

- Privilegio CONTROL sobre la tabla
- Autorización SYSADM o DBADM

Sintaxis



opciones-en-línea:

opciones-optimización-consulta:

Descripción

nombre-tabla
Identifica la tabla que se debe renovar.

El nombre, incluyendo el esquema implícito o explícito, debe identificar una tabla que ya exista en el servidor actual. La tabla debe permitir la sentencia REFRESH TABLE (SQLSTATE 42809). Ello incluye las tablas de consultas materializadas que se han definido con:

- REFRESH IMMEDIATE
- REFRESH DEFERRED

opciones-en-línea
Especifica la accesibilidad de la tabla mientras se está procesando.

ALLOW NO ACCESS
Especifica que ningún otro usuario puede acceder a la tabla mientras se está renovando, excepto si utiliza el nivel de aislamiento de lectura no confirmada.

ALLOW READ ACCESS
Especifica que los demás usuarios tienen acceso de sólo lectura a la tabla mientras se renueva.

ALLOW WRITE ACCESS
Especifica que los demás usuarios tienen acceso de lectura y escritura a la tabla mientras se renueva.

Para impedir una retrotracción de toda la sentencia a causa de un tiempo de espera excedido de bloqueo al utilizar la opción ALLOW READ ACCESS o ALLOW WRITE ACCESS, se recomienda la emisión de una sentencia SET CURRENT LOCK TIMEOUT (especificando la opción WAIT) antes de ejecutar la sentencia REFRESH TABLE y restaurar el registro especial a su valor anterior posteriormente. Tenga en cuenta, sin embargo, que el registro CURRENT LOCK TIMEOUT sólo afecta a un conjunto específico de tipos de bloqueo, no a todos los tipos.

opciones-optimización-consulta
Especifica las opciones de optimización de la consulta para la renovación de las tablas de consultas materializadas de REFRESH DEFERRED.

ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY
Especifica que, si el registro especial CURRENT REFRESH AGE se establece en 'ANY', la renovación de *nombre-tabla* permitirá el uso de las tablas de consultas materializadas REFRESH DEFERRED para optimizar la consulta que se utilice para renovar *nombre-tabla*. Si el *nombre-tabla* no es una tabla de consultas materializadas REFRESH DEFERRED, se devuelve un error (SQLSTATE 428FH). Las tablas de consultas materializadas REFRESH IMMEDIATE siempre se toman en consideración para la optimización de la consulta.

INCREMENTAL
Especifica una renovación incremental de la tabla donde sólo se considera la parte delta (si existe) de sus tablas subyacentes o el contenido de una tabla de etapas asociada (si existe una y si su contenido es coherente). Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que la definición de la tabla de 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.

Normas

- Si se emite REFRESH TABLE en una tabla de consultas materializadas que hace referencia a uno o más apodos, el ID de autorización de la sentencia deberá disponer de autorización para poder realizar selecciones desde las tablas de la fuente de datos (SQLSTATE 42501).

Notas

- Cuando la sentencia se utiliza para renovar una tabla de consultas materializadas REFRESH IMMEDIATE cuyas tablas subyacentes se han cargado, enlazado o desenlazado, puede que el sistema opte por renovar de manera incremental la tabla de consultas materializadas con las partes delta 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 delta de sus tablas subyacentes que se han capturado en la tabla de etapas. Sin embargo, existen algunas situaciones en las que no es posible esta optimización y es necesario realizar una renovación completa (es decir, volver a calcular la definición de la tabla de 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 (SQLSTATE 55019).
- Si se utiliza la opción ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY, asegúrese de que el orden de renovación de las tablas de consultas materializadas REFRESH DEFERRED sea correcto. Por ejemplo, imagine dos tablas de consultas materializadas, MQT1 y MQT2, cuyas consultas materializadas comparten las mismas tablas subyacentes. La consulta materializada de MQT2 se puede calcular utilizando MQT1, en lugar de las tablas subyacentes. Si se emplean distintas sentencias para renovar estas dos tablas de consultas materializadas, y se renueva MQT2 en primer lugar, el sistema puede elegir utilizar el contenido de MQT1, que todavía no se ha renovado, para renovar MQT2. En ese caso, MQT1 contendría los datos actuales pero MQT2 podría contener todavía datos obsoletos, aunque la renovación de ambas se realizara casi al mismo tiempo. El orden de renovación correcto, si se utilizan dos sentencias REFRESH en lugar de una, es renovar MQT1 primero.
- Si la tabla de consultas materializadas tiene una tabla de etapas asociada, la tabla de etapas se podará cuando la renovación se haya ejecutado satisfactoriamente.

RELEASE (conexión)

La sentencia RELEASE (conexión) establece una o más conexiones en estado pendiente de liberación.

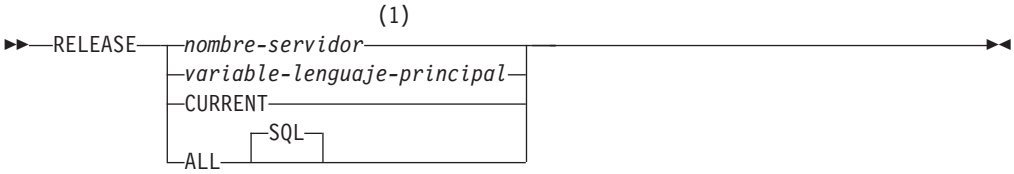
Invocación

Aunque un recurso SQL interactivo pueda proporcionar una interfaz que dé la apariencia de ejecución interactiva, esta sentencia sólo puede incorporarse dentro de un programa de aplicación. Es una sentencia ejecutable que no puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Notas:

1 Observe que un servidor de aplicaciones llamado CURRENT o ALL sólo puede identificarse mediante una variable del lenguaje principal o un identificador delimitado.

Descripción

nombre-servidor **o** *variable-lenguaje-principal*
Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lenguaje-principal* 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 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.

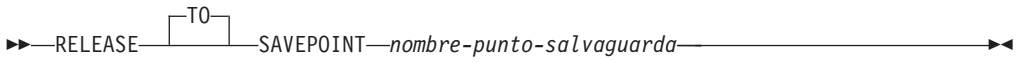
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-punto-salvaguarda
Especifica el punto de salvaguarda que se debe liberar. Cualquier punto de salvaguarda anidado en el punto de salvaguarda nombrado también se liberará. La retrotracción a ese punto de salvaguarda, o a cualquier punto de salvaguarda anidado en él, ya no será posible. Si el punto de salvaguarda no existe en el nivel de punto de salvaguarda actual (consulte la sección "Normas" en la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B001). El *nombre-puntosalvaguarda* no puede empezar por 'SYS' (SQLSTATE 42939).

Notas

- El nombre del punto de salvaguarda que se ha liberado se puede volver a utilizar ahora en otra sentencia SAVEPOINT, sin tener en cuenta si se ha especificado la palabra clave UNIQUE en una sentencia SAVEPOINT anterior especificando este mismo nombre de punto de salvaguarda.

Ejemplo

Ejemplo 1: Liberación de un punto de salvaguarda llamado SAVEPOINT1.

```
RELEASE SAVEPOINT SAVEPOINT1
```

RENAME TABLESPACE

La sentencia RENAME TABLESPACE cambia el nombre de un espacio de tablas existente.

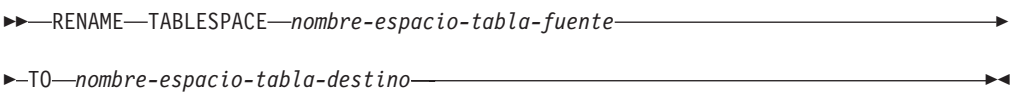
Invocación

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

Autorización

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

Sintaxis



Descripción

nombre-espacio-tabla-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-tabla-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
```

RENAME

La sentencia RENAME cambia el nombre de una tabla o de un índice existente.

Invocación

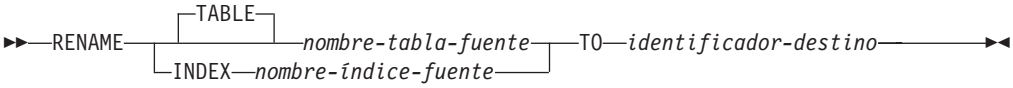
Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

- Privilegio CONTROL sobre la tabla o índice
- Privilegio ALTERIN para el esquema
- Autorización SYSADM o DBADM

Sintaxis



Descripción

TABLE *nombre-tabla-fuente*
Identifica la tabla existente que se debe renombrar. El nombre, incluyendo el nombre de esquema, debe identificar una tabla que ya exista en la base de datos (SQLSTATE 42704). No debe ser el nombre de una tabla de 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-fuente*
Especifica el nombre del índice existente cuyo nombre va a cambiarse. El nombre, incluido el nombre de esquema, debe identificar un índice que ya exista en la base de datos (SQLSTATE 42704). No 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
- No estar referenciada en un objeto XSR que se ha habilitado para su descomposición

Se devuelve un error (SQLSTATE 42986) si la tabla fuente viola una o más de esas condiciones.

Cuando se cambia el nombre de un índice:

- El índice fuente no debe ser un índice generado por el sistema para una tabla de implementación en la que se basa una tabla con tipo (SQLSTATE 42858).

Notas

- Las entradas del catálogo se actualizan para reflejar el nuevo nombre de la tabla o índice.
- *Todas* las autorizaciones que se asocian a la tabla fuente o al nombre del índice se *transfieren* al nuevo nombre de tabla o índice (las tablas del catálogo de autorización se actualizan de acuerdo con ello).
- Los índices definidos en la tabla fuente se *transfieren* a la nueva tabla (las tablas de catálogo de índice se actualizan del modo apropiado).
- RENAME TABLE invalida cualquier paquete que dependa de la tabla fuente. RENAME INDEX invalida cualquier paquete que dependa del índice fuente.
- Si se utiliza un alias para el *nombre-tabla-fuente*, éste deberá resolverse de modo que dé como resultado un nombre de tabla. La tabla se renombra dentro del esquema de la tabla. El alias no se cambia por la sentencia RENAME y continúa haciendo referencia al anterior nombre de tabla.
- El nombre de una tabla con una clave primaria o con restricciones de unicidad podrá cambiarse si ninguna clave foránea hace referencia a la clave primaria o a las restricciones de unicidad.

Ejemplos

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
```

REPEAT

La sentencia REPEAT ejecuta una sentencia o grupo de sentencias hasta que se cumpla una condición de búsqueda.

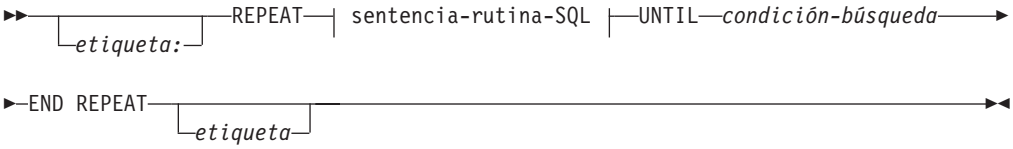
Invocación

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

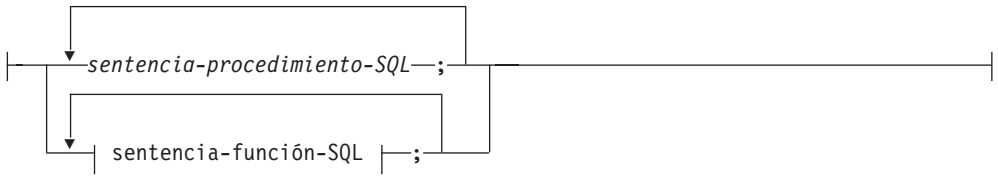
Autorización

No se requieren privilegios para invocar la sentencia REPEAT. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia REPEAT.

Sintaxis



sentencia-rutina-SQL:



Descripción

etiqueta
Especifica la etiqueta de la sentencia REPEAT. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica una etiqueta final, se debe especificar también la etiqueta inicial correspondiente.

sentencia-procedimiento-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte *sentencia-procedimiento-SQL* en "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 "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
```

RESIGNAL

La sentencia RESIGNAL se utilizan en un manejador de condiciones para retransmitir la condición que ha activado el manejador o para generar una condición alternativa a fin de poderse procesar a un nivel más alto. Produce una excepción, un aviso o una condición de no encontrado que se debe devolver, junto con el texto de mensaje opcional.

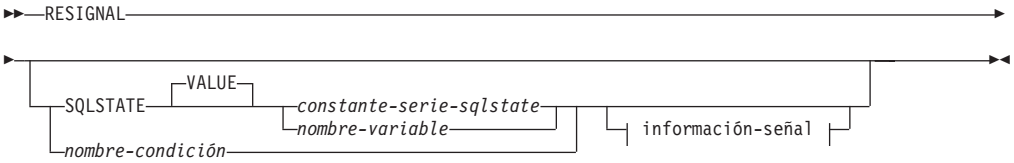
Invocación

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

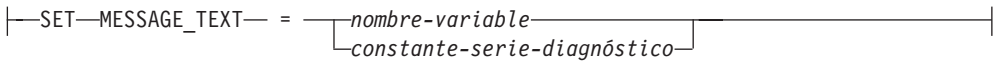
Autorización

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 exactamente 5 caracteres que siguen las normas para QLSTATE:

- 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
```

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 de SQL, devuelve opcionalmente un valor de estado de tipo entero.

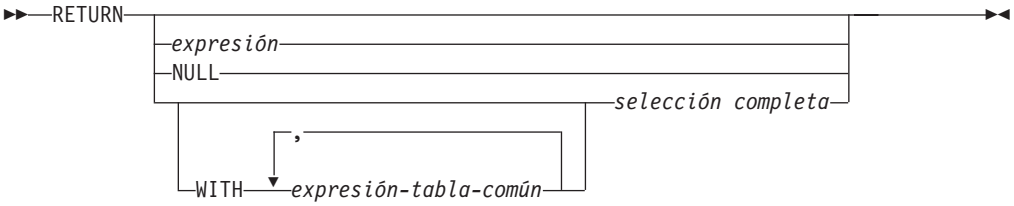
Invocación

Esta sentencia se puede incluir en una función de SQL, un método de SQL o un procedimiento de SQL. No se trata de una sentencia ejecutable y no puede prepararse de forma dinámica.

Autorización

No se requieren privilegios para invocar la sentencia RETURN. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar cualquier expresión o selección completa incorporada en la sentencia RETURN.

Sintaxis



Descripción

expresión
Especifica un valor devuelto procedente de la rutina:

- Si la rutina es una función o un método, se debe especificar *expresión*, NULL o *selección completa* (SQLSTATE 42631) 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 428F2). Un procedimiento no puede devolver NULL o una *selección completa*.

NULL
Especifica que la función o el método devuelve un valor nulo del tipo de datos definido en la cláusula RETURNS. No 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).
- En un procedimiento de SQL, no se permite una sentencia RETURN en el cuerpo de un manejador de condiciones (SQLSTATE 42601).

Notas

- Cuando se devuelve un valor desde un procedimiento, el llamador puede acceder al valor:
  - utilizando la sentencia GET DIAGNOSTICS para recuperar el valor de DB2_RETURN_STATUS cuando se ha llamado al procedimiento de SQL desde otro procedimiento de SQL
  - utilizando el parámetro encontrado para el marcador de parámetro de valor devuelto en la sintaxis de la cláusula de escape CALL (?=CALL...) en una aplicación CLI
  - directamente desde el campo `sqlerrd[0]` de la SQLCA, tras procesarse la cláusula CALL de un procedimiento de SQL. Este campo sólo es válido si el SQLCODE es cero o un valor positivo (de lo contrario, debe darse por supuesto un valor de -1).

Ejemplos

Utilice una sentencia RETURN para salir de un procedimiento de SQL con un valor de estado de cero si la ejecución es satisfactoria y de -200 si no lo es.

```
BEGIN
...
  GOTO FAIL
...
  SUCCESS: RETURN 0
  FAIL: RETURN -200
END
```

REVOKE (autorizaciones de bases de datos)

Esta forma de la sentencia REVOKE revoca las autorizaciones que se aplican a toda la base de datos.

Invocación

Esta sentencia 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).

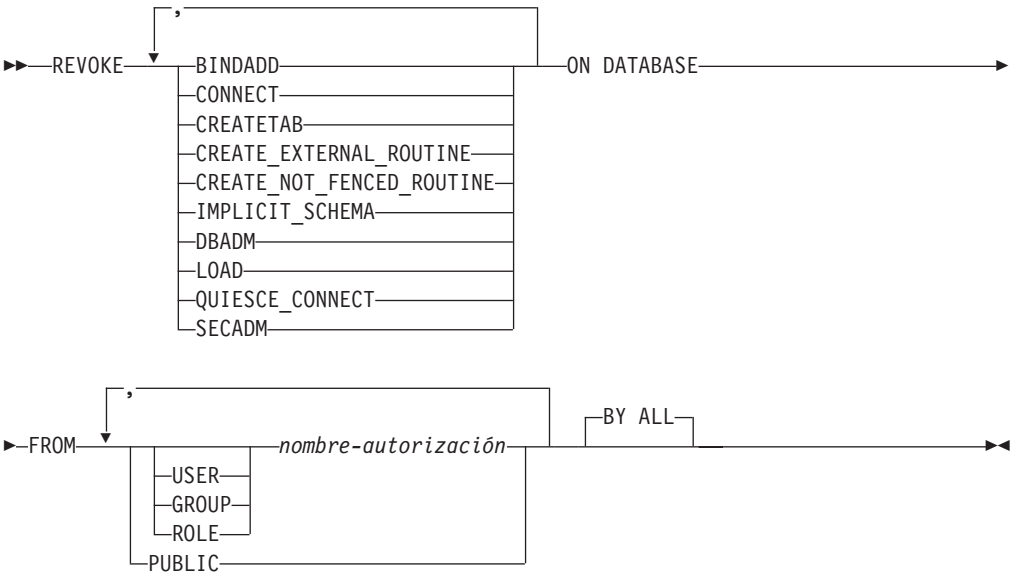
Autorización

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

Es necesaria la autorización SYSADM para revocar cualquiera de las autorizaciones siguientes:

- DBADM
- SECADM

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.

La autorización IMPLICIT_SCHEMA no puede revocarse de un *nombre-autorización* que contenga la autorización DBADM sin revocar también la autorización DBADM (SQLSTATE 42504).

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 bases 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.

SECADM
Revoca la autorización de administrador de seguridad.

FROM
Indica a quién se revocan las autorizaciones.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca las autorizaciones de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.DBAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- La revocación de un privilegio específico no revoca necesariamente la capacidad de realizar una acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o una función tienen otros privilegios o si el usuario tiene un grado de autorización más alto como, por ejemplo, DBADM.
- ***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.

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).

Ejemplo 3: Revoque la autorización del administrador de seguridad del usuario Walid.

```
REVOKE SECADM ON DATABASE FROM USER Walid
```

REVOKE (Exención)

Este formato de la sentencia REVOKE revoca una exención a una norma de acceso de control de acceso basado en etiquetas (LBAC).

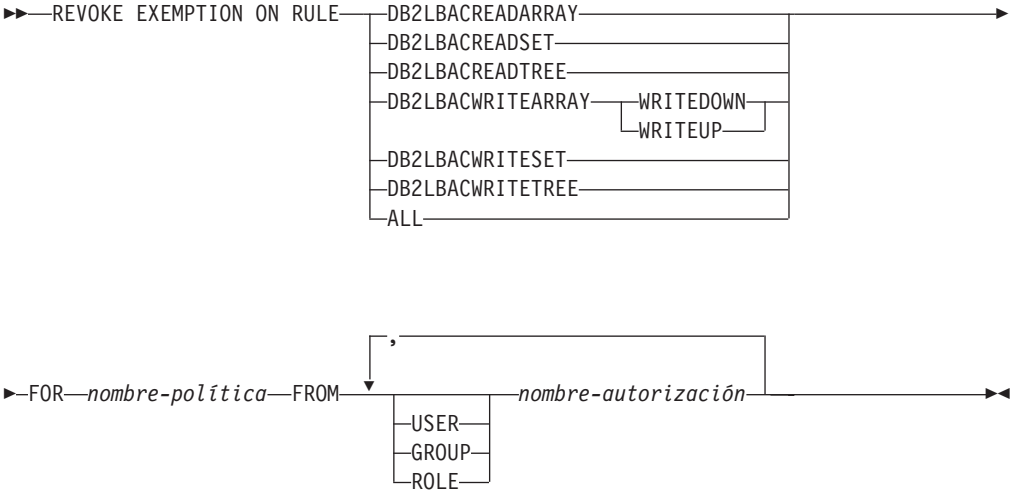
Invocación

Esta sentencia 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

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

Sintaxis



Descripción

EXEMPTION ON RULE
Revoca la exención en una norma de acceso.

DB2LBACREADARRAY
Revoca una exención en la norma DB2LBACREADARRAY predefinida.

DB2LBACREADSET
Revoca una exención en la norma DB2LBACREADSET predefinida.

DB2LBACREADTREE
Revoca una exención en la norma DB2LBACREADTREE predefinida.

DB2LBACWRITEARRAY
Revoca una exención en la norma DB2LBACWRITEARRAY predefinida.

WRITEDOWN
Especifica que la exención sólo se aplica a la escritura hacia abajo.

WRITEUP
Especifica que la exención sólo se aplica a la escritura hacia arriba.

DB2LBACWRITESET
Revoca una exención en la norma DB2LBACWRITESET predefinida.

DB2LBACWRITETREE
Revoca una exención en la norma DB2LBACWRITETREE predefinida.

ALL
Revoca las exenciones en todas las normas predefinidas.

FOR *nombre-política*
Especifica el nombre de la política de seguridad en la que se revocarán las exenciones.

FROM
Especifica a quién se le revoca la exención.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SECURITYPOLICYEXEMPTIONS en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Ejemplos

Ejemplo 1: Revocar la exención en la norma de acceso DB2LBACREADSET para la política de seguridad DATA_ACCESS del usuario WALID.

```
REVOKE EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS
  FROM USER WALID
```

Ejemplo 2: Revoca una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEDOWN para la política de seguridad DATA_ACCESS del usuario BOBBY.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
  FOR DATA_ACCESS FROM USER BOBBY
```

Ejemplo 3: Revoca una exención en la norma de acceso DB2LBACWRITEARRAY con la opción WRITEUP para la política de seguridad DATA_ACCESS del usuario BOBBY.

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
  FOR DATA_ACCESS FROM USER BOBBY
```

REVOKE (privilegios de variable global)

Este formato de la sentencia REVOKE revoca uno o más privilegios de una variable global creada.

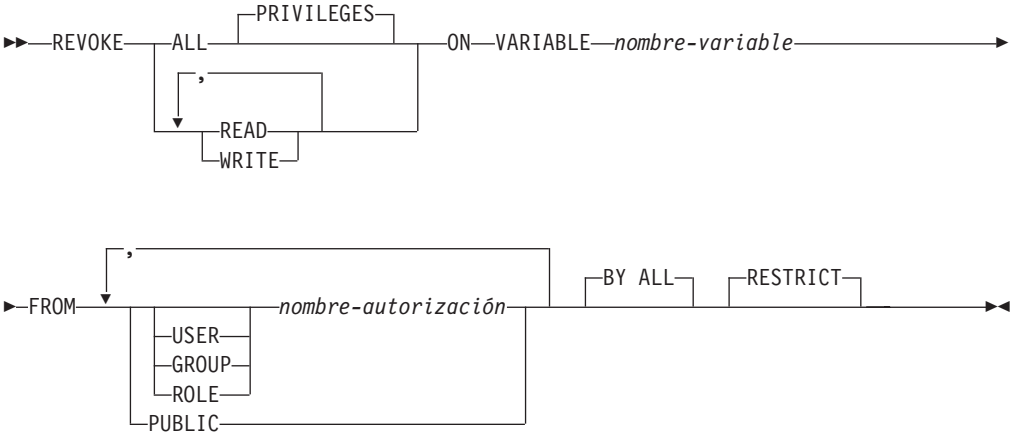
Invocación

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

Autorización

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

Sintaxis



Descripción

ALL PRIVILEGES
Revoca todos los privilegios que retiene un *nombre-autorización* para la variable global especificada. Si no se especifica ALL, debe especificarse READ o WRITE. No se debe especificar READ o WRITE más de una vez.

READ
Revoca el privilegio para leer el valor de la variable global especificada.

WRITE
Revoca el privilegio para asignar un valor a la variable global especificada.

ON VARIABLE *nombre-variable*
Identifica la variable global sobre la que van a revocarse uno o más privilegios. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (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.

ROLE
Especifica que el *nombre-autorización* identifica una función existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios especificados de PUBLIC.

BY ALL
Revoca todos los privilegios especificados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

RESTRICT
Especifica que la sentencia dará error si se revoca algún objeto que dependa del privilegio. Este es el comportamiento por omisión.

Normas

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.VARIABLEAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092.
- Si alguna función SQL, método SQL, procedimiento, vista, desencadenante u otra variable global contiene una variable global y depende de que se revoque el privilegio, la operación de revocación fallará (SQLSTATE 42893).

Notas

- Si el privilegio READ se revoca en una variable global, los paquetes con una dependencia para grabar el valor de la variable global (por ejemplo, por medio de la sentencia SET) no resultan afectados, ya que la grabación en una variable global está controlada por medio del privilegio WRITE en dicha variable global.
- Si el privilegio WRITE se revoca en una variable global, los paquetes con una dependencia para leer el valor de la variable global no resultan afectados, ya que la lectura de una variable global está controlada por medio del privilegio READ en dicha variable global.
- La revocación de un privilegio no afecta necesariamente la posibilidad de realizar la acción. Es posible que un usuario no pueda proceder en el caso de que se retenga el privilegio requerido por medio de la pertenencia a una función o grupo diferente o por medio de PUBLIC.

Ejemplo

Revoque el privilegio WRITE en la variable global MYSCHEMA.MYJOB_PRINTER del usuario ZUBIRI.

```
REVOKE WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER FROM ZUBIRI
```

REVOKE (privilegios de índice)

Esta forma de la sentencia REVOKE revoca el privilegio CONTROL en un índice.

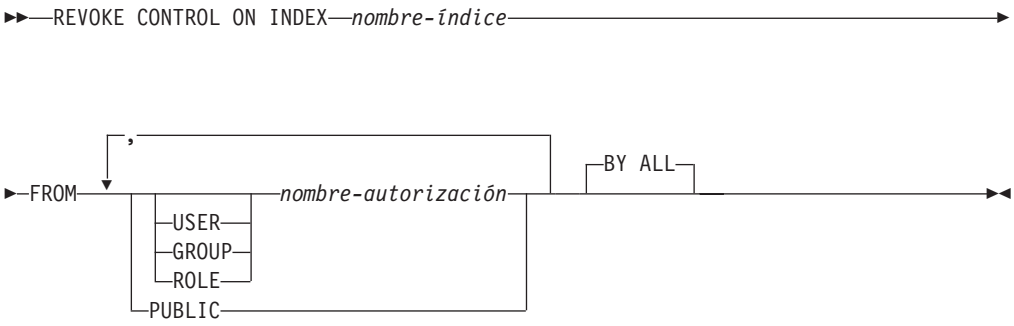
Invocación

Esta sentencia 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

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

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca 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

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.INDEXAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o una función tienen otros privilegios o si el usuario tiene autorizaciones como, por ejemplo, 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 KIESLER
```

Ejemplo 2: Revoque el privilegio para eliminar un índice LUNCHITEMS del usuario CHEF y del grupo WAITERS.

```
REVOKE CONTROL ON INDEX LUNCHITEMS
  FROM USER CHEF, GROUP WAITERS
```

REVOKE (privilegios de paquete)

Esta forma de la sentencia REVOKE revoca los privilegios CONTROL, BIND y EXECUTE en un paquete.

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).

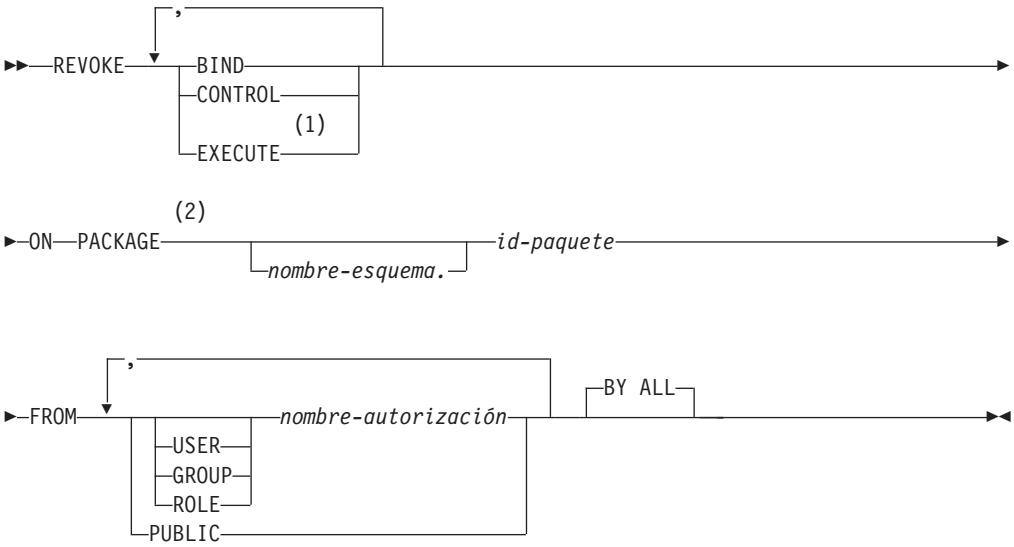
Autorización

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

Es necesaria la autorización DBADM o SYSADM para revocar el privilegio CONTROL.

Sintaxis



Notas:

1 RUN se puede utilizar como sinónimo de EXECUTE.

2 PROGRAM puede utilizarse como sinónimo de PACKAGE.

Descripción

BIND
Revoca el privilegio para ejecutar BIND o REBIND para el paquete al que se hace referencia o bien para añadir una nueva versión del paquete al que se hace referencia.

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.PACKAGEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o una función tienen otros privilegios o si el usuario tiene privilegios como, por ejemplo, ALTERIN en el esquema de un paquete.

Ejemplos

Ejemplo 1: Revoque el privilegio EXECUTE en el paquete CORPDATA.PKGA de PUBLIC.

```
REVOKE EXECUTE
  ON PACKAGE CORPDATA.PKGA
  FROM PUBLIC
```

Ejemplo 2: Revoque la autorización CONTROL en el paquete RRSP_PKG para el usuario FRANK y para PUBLIC.

```
REVOKE CONTROL
  ON PACKAGE RRSP_PKG
  FROM USER FRANK, PUBLIC
```

REVOKE (Función)

Este formato de la sentencia REVOKE revoca funciones de usuarios, grupos u otras funciones.

Invocación

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

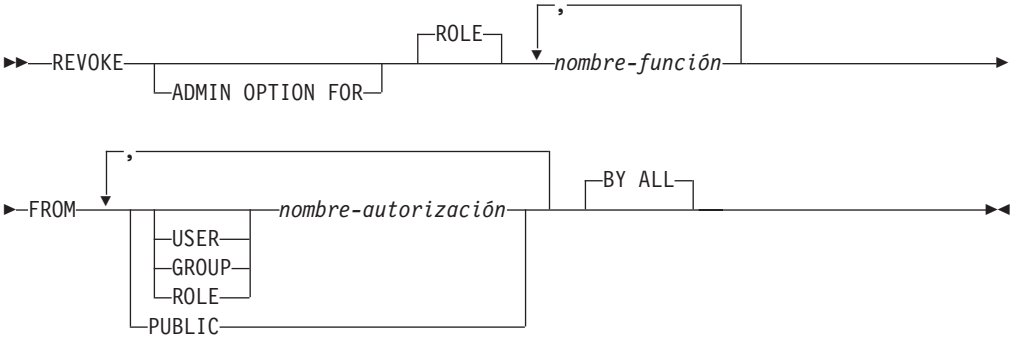
Autorización

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

- WITH ADMIN OPTION en la función
- Autorización SECADM

Se necesita la autorización SECADM para revocar ADMIN OPTION FOR *nombre-función* de un *nombre-autorización* o para revocar un *nombre-función* de un *nombre-autorización* que tiene WITH ADMIN OPTION en dicha función.

Sintaxis



Descripción

ADMIN OPTION FOR
Revoca WITH ADMIN OPTION en *nombre-función*. WITH ADMIN OPTION en *nombre-función* debe retenerse por medio de *nombre-autorización* o por medio de PUBLIC, si se especifica PUBLIC (SQLSTATE 42504). Si se especifica la cláusula ADMIN OPTION FOR, sólo se revoca WITH ADMIN OPTION en ROLE *nombre-función*, no la propia función.

ROLE *nombre-función*
Especifica la función que va a revocarse. El *nombre-función* debe identificar una función existente en el servidor actual (SQLSTATE 42704) que se haya otorgado a *nombre-autorización* o a PUBLIC, si se ha especificado PUBLIC (SQLSTATE 42504).

FROM
Especifica a quién se revoca la función.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

ROLE
Especifica que el *nombre-autorización* identifica un rol existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca las funciones especificadas de PUBLIC.

BY ALL
Revoca el *nombre-función* de cada *nombre-autorización* especificado al que se otorgó explícitamente dicha función, sin tener en cuenta quién la otorgó. Este es el comportamiento por omisión.

Normas

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.ROLEAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092.
- El *nombre-función* no debe identificar una función o una función que contenga *nombre-función*, si la función tiene privilegio EXECUTE sobre una rutina o privilegio USAGE sobre una secuencia y un objeto SQL que no sea un paquete depende de la rutina o secuencia (SQLSTATE 42893). El propietario del objeto SQL es *nombre-autorización* o cualquier usuario que sea miembro de *nombre-autorización*, donde *nombre-autorización* es una función.

Notas

- Si se revoca una función de un *nombre-autorización* o de PUBLIC, todos los privilegios que retiene la función ya no están a disposición del *nombre-autorización* o de PUBLIC a través de dicha función.
- La revocación de una función no revoca necesariamente la posibilidad de realizar una acción concreta por medio de un privilegio otorgado a dicha función. Un usuario puede proseguir si PUBLIC, un grupo al que pertenezca el usuario u otra función otorgada al usuario tienen otros privilegios o si el usuario tiene un nivel de autorización más alto como, por ejemplo, DBADM.

Ejemplos

Ejemplo 1: Revoque la función INTERN de la función DOCTOR y la función DOCTOR de la función SPECIALIST.

```
REVOKE ROLE INTERN FROM ROLE DOCTOR

REVOKE ROLE DOCTOR FROM ROLE SPECIALIST
```

Ejemplo 2: Revoque la función INTERN de PUBLIC.

```
REVOKE ROLE INTERN FROM PUBLIC
```

Ejemplo 3: Revoque la función SPECIALIST del usuario BOB y del grupo TORONTO.

```
   REVOKE ROLE SPECIALIST
FROM USER BOB, GROUP TORONTO BY ALL
```

REVOKE (privilegios de rutina)

Esta forma de la sentencia REVOKE revoca privilegios para una rutina (función, método o procedimiento).

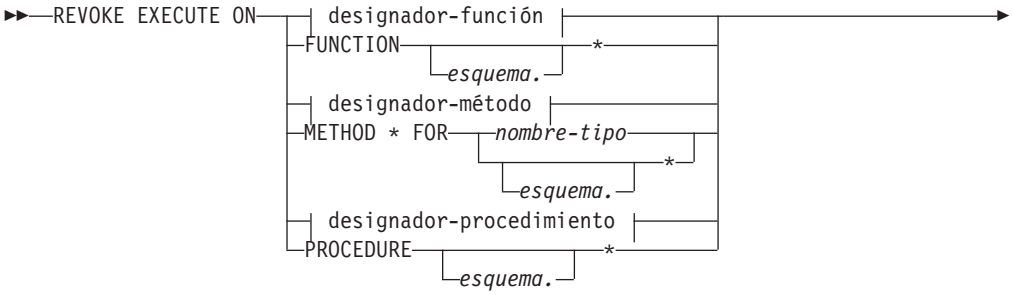
Invocación

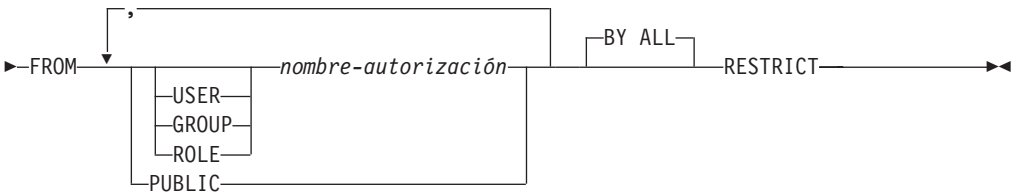
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

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

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca el privilegio EXECUTE de PUBLIC.

BY ALL
Revoca el privilegio EXECUTE de todos los usuarios especificados a los que se ha otorgado explícitamente el privilegio, independientemente de qué usuario lo ha otorgado. 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 fuente.
- La pérdida del privilegio EXECUTE puede dar lugar a que el propietario de la vista, del activador, de la restricción, de la extensión de índice, de la función de SQL, del método de SQL, del grupo de transformación o de la función con fuente ya no pueda ejecutar la rutina especificada.

Normas

- No es posible revocar el privilegio EXECUTE para una función o método que se ha definido con el esquema 'SYSIBM' o 'SYSFUN' (SQLSTATE 42832).
- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.ROUTINEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- Si un paquete depende de una rutina (función, método o procedimiento) y se revoca el privilegio EXECUTE sobre dicha rutina de PUBLIC, un usuario o una función, el paquete se convierte en no operativo si la rutina es una función o método y se convierte en no válido si la rutina es un procedimiento, a menos que el propietario del paquete todavía tenga el privilegio EXECUTE sobre la rutina. El propietario del paquete todavía tiene el privilegio EXECUTE si:
  - Se ha otorgado explícitamente el privilegio EXECUTE al propietario del paquete
  - El propietario del paquete es miembro de una función que tiene el privilegio EXECUTE
  - Se ha otorgado el privilegio EXECUTE a PUBLIC

  Dado que los privilegios de grupo no se consideran para los paquetes estáticos, el paquete se convierte en no operativo (en el caso de una función o de un método) o en no válido (en el caso de un procedimiento) incluso si un grupo al que pertenece el propietario del paquete tiene el privilegio EXECUTE.

Ejemplos

Ejemplo 1: Revoque el privilegio EXECUTE para la función CALC_SALARY del usuario JONES. Se da por supuesto que, en el esquema, sólo existe una función con el nombre de función CALC_SALARY.

```
REVOKE EXECUTE ON FUNCTION CALC_SALARY FROM JONES RESTRICT
```

Ejemplo 2: Revoque el privilegio EXECUTE para el procedimiento VACATION_ACCR de todos los usuarios del servidor actual.

```
REVOKE EXECUTE ON PROCEDURE VACATION_ACCR FROM PUBLIC RESTRICT
```

Ejemplo 3: Revoque el privilegio EXECUTE para la función NEW_DEPT_HIRES de HR (Recursos humanos). La función tiene dos parámetros de entrada de tipo INTEGER y CHAR(10) respectivamente. Se da por supuesto que el esquema tiene más de una función denominada NEW_DEPT_HIRES.

```
REVOKE EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
  FROM HR RESTRICT
```

Ejemplo 4: Revoque el privilegio EXECUTE para el método SET_SALARY para el tipo EMPLOYEE del usuario Jones.

```
REVOKE EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE FROM JONES RESTRICT
```

REVOKE (privilegios de esquema)

Esta forma de la sentencia REVOQUE revoca los privilegios en un esquema.

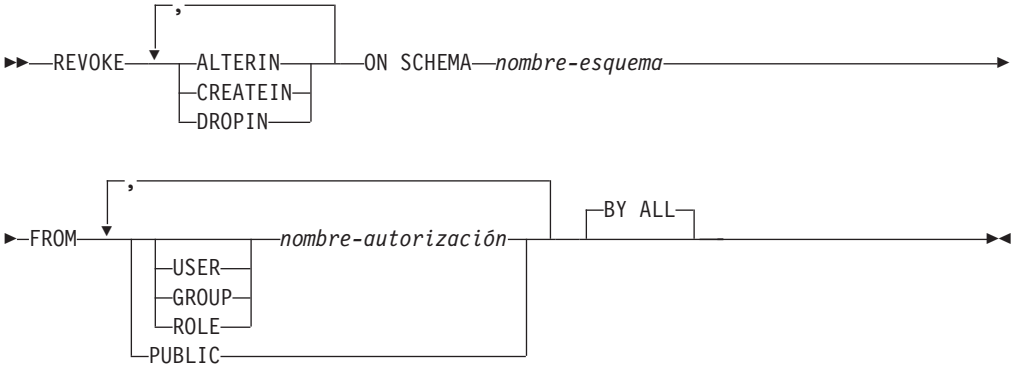
Invocación

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

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

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SCHEMAAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o una función tienen otros privilegios o si el usuario tiene un grado de autorización más alto como, por ejemplo, DBADM.

Ejemplos

Ejemplo 1: Suponiendo que USER4 sea sólo un usuario y no un grupo, revoque el privilegio para crear objetos en el esquema DEPTIDX del usuario USER4.

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

Ejemplo 2: Revoque el privilegio de eliminar objetos en el esquema LUNCH del usuario CHEF y del grupo WAITERS.

```
REVOKE DROPIN ON SCHEMA LUNCH
  FROM USER CHEF, GROUP WAITERS
```

REVOKE (Etiqueta de seguridad)

Este tipo de sentencia REVOKE revoca una etiqueta de seguridad de control de acceso basado en etiquetas (LBAC).

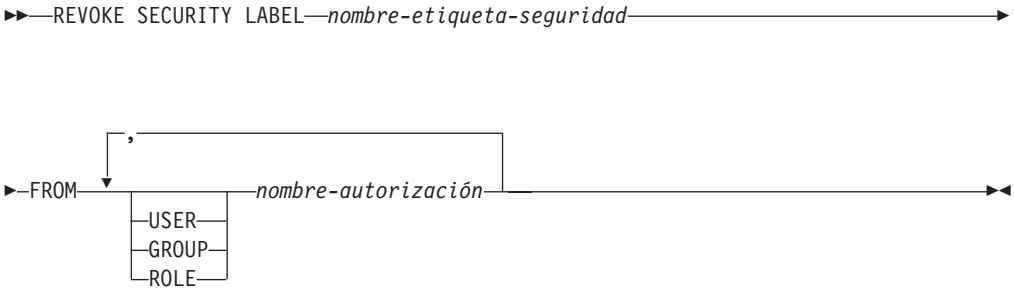
Invocación

Esta sentencia 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

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

Sintaxis



Descripción

SECURITY LABEL *nombre-etiqueta-seguridad*
Revoca la etiqueta de seguridad *nombre-etiqueta-seguridad*. El nombre debe calificarse con una política de seguridad (SQLSTATE 42704) y debe identificar una etiqueta de seguridad que exista en el servidor actual (SQLSTATE 42704) y que esté retenida por *nombre-autorización* (SQLSTATE 42504).

FROM
Especifica a quién se le revoca la etiqueta de seguridad especificada.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función. El nombre de función debe existir en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:

- Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SECURITYLABELACCESS en la que el otorgado es *nombre-autorización*:
  - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
  - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
  - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
  - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Ejemplos

Ejemplo 1: Revocar la etiqueta de seguridad EMPLOYEESECLABEL, que forma parte de la política de seguridad DATA_ACCESS, del usuario WALID.

```
REVOKE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  FROM USER WALID
```

REVOKE (privilegios de secuencia)

Esta forma de la sentencia REVOKE revoca los privilegios en una secuencia.

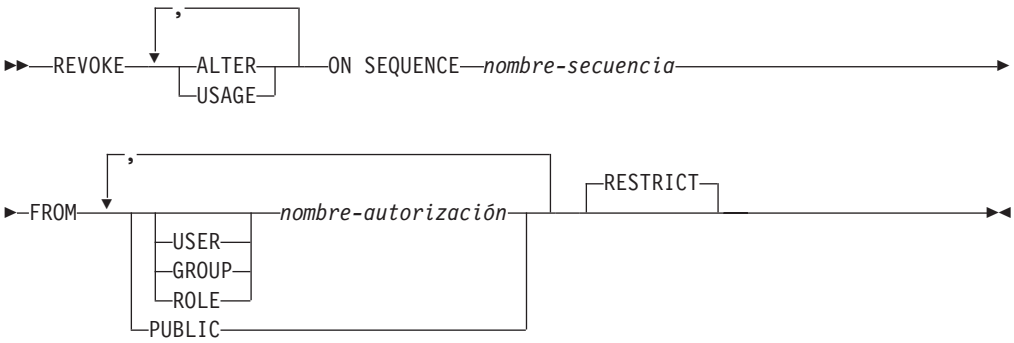
Invocación

Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente. 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 DBADM o SYSADM.

Sintaxis



Descripción

ALTER
Revoca el privilegio de cambiar las propiedades de una secuencia o reiniciar la generación de números de secuencia mediante la sentencia ALTER SEQUENCE.

USAGE
Revoca el privilegio de poder hacer referencia a una secuencia mediante una *expresión-nextval* o *expresión-prevval*.

ON SEQUENCE *nombre-secuencia*
Identifica la secuencia para la que van a revocarse los privilegios especificados. El nombre de la secuencia, incluido un calificador de esquema implícito o explícito, debe identificar de forma exclusiva a una secuencia existente en el servidor actual. Si no existe ninguna secuencia con este nombre, se devuelve un error (SQLSTATE 42704).

FROM
Especifica a quién se revocarán los privilegios.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios especificados de PUBLIC.

RESTRICT
Esta palabra clave opcional indica que la sentencia dará error si se revoca algún objeto que dependa del privilegio.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.SEQUENCEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- Al revocar un privilegio sobre una secuencia del ID de autorización bajo el cual se ha vinculado un paquete, el paquete se convierte en no válido si el ID de autorización no sigue teniendo el privilegio sobre la secuencia a través de diferentes medios como, por ejemplo, ser miembro de una función que tiene el privilegio.
- La revocación de un privilegio específico no elimina necesariamente la capacidad de realizar una acción. Un usuario puede continuar si PUBLIC o un grupo al que pertenezca el usuario tienen otros privilegios, o si el usuario tiene un nivel de autorización más alto como, por ejemplo, DBADM.

Ejemplos

Ejemplo 1: Revoque al usuario 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
```

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.

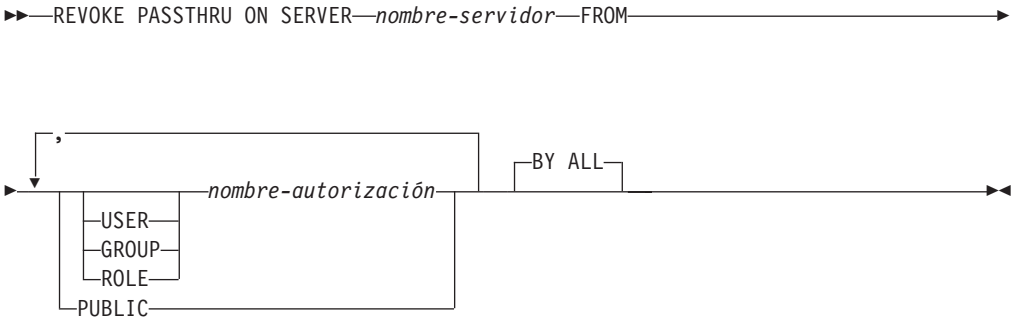
Invocación

Esta sentencia puede incorporarse 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

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

Sintaxis



Descripción

SERVER *nombre-servidor*
Nombra la fuente de datos para la cual se está revocando el privilegio para utilizarse en modalidad de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

FROM
Especifica a quién se revoca el privilegio.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca de PUBLIC el privilegio de realizar un paso a través de *nombre-servidor*.

BY ALL
Revoca el privilegio de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.PASSTHRUAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Ejemplos

Ejemplo 1: Revoque el privilegio que USER6 tiene para utilizar la modalidad de paso a través para la fuente de datos MOUNTAIN.

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

Ejemplo 2: Revoque el privilegio que el grupo D024 tiene para usar la modalidad de paso a través para la fuente de datos EASTWING.

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

Los miembros del grupo D024 ya no podrán utilizar su ID de grupo para realizar un paso a través para EASTWING. Pero si cualquier miembro tiene el privilegio para usar el paso a través para EASTWING con su propio ID de usuario, conservará este privilegio.

REVOKE (privilegio SETSESSIONUSER)

Este formato de la sentencia REVOKE revoca uno o más privilegios SETSESSIONUSER de uno o más ID de autorización.

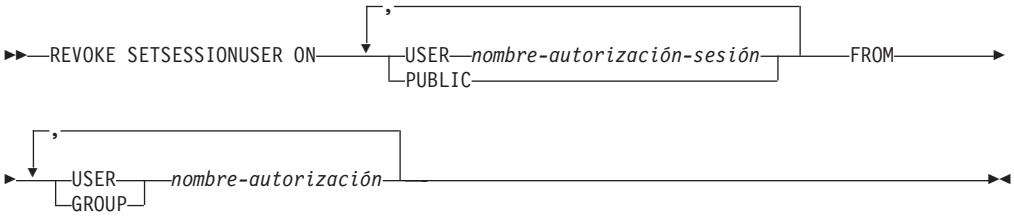
Invocación

Esta sentencia 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

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

Sintaxis



Descripción

SETSESSIONUSER ON
Revoca el privilegio de asumir la identidad de un nuevo ID de autorización.

USER *nombre-autorización-sesión*
Especifica el ID de autorización que el *nombre-autorización* es capaz de asumir, mediante la sentencia SET SESSION AUTHORIZATION. El *nombre-autorización-sesión* debe identificar a un usuario que el *nombre-autorización* pueda suponer, no un grupo (SQLSTATE 42504).

PUBLIC
Especifica que se revocarán todos los privilegios para configurar la autorización de la sesión.

FROM
Especifica a quién se revoca el privilegio.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica un nombre de grupo.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios o grupos.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

Ejemplos

Ejemplo 1: El usuario PAUL tiene el privilegio de configurar la autorización de la sesión a WALID, y por ello ejecutar las sentencias de SQL como usuario WALID. La sentencia siguiente revoca ese privilegio.

```
REVOKE SETSESSIONUSER ON USER WALID
  FROM USER PAUL
```

Ejemplo 2: El usuario GUYLAINE tiene el privilegio de configurar la autorización de la sesión a BOBBY, RICK o KEVIN, y por ello ejecutar las sentencias de SQL como BOBBY, RICK o KEVIN. La sentencia siguiente revoca el privilegio de utilizar dos de esos ID de autorización. Después de que se ejecute esta sentencia, GUYLAINE sólo podrá configurar la autorización de la sesión a KEVIN.

```
REVOKE SETSESSIONUSER ON USER BOBBY, USER RICK
  FROM USER GUYLAINE
```

Ejemplo 3: El grupo ACCTG y el usuario WALID pueden configurar la autorización de la sesión a cualquier ID de autorización. La sentencia siguiente revoca ese privilegio de ACCTG y WALID.

```
REVOKE SETSESSIONUSER ON PUBLIC
  FROM USER WALID, GROUP ACCTG
```

REVOKE (privilegios de espacio de tablas)

Esta forma de la sentencia REVOKE revoca el privilegio USE para una tabla.

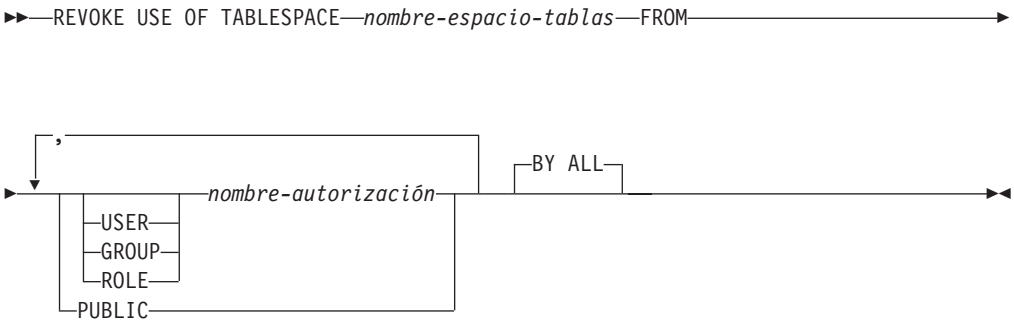
Invocación

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

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

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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

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 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

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado en la vista de catálogo SYSCAT.TBSPACEAUTH en la que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- La revocación del privilegio USE no revoca necesariamente la capacidad para crear tablas en ese espacio de tablas. Un usuario puede todavía crear tablas en ese espacio de tablas si el privilegio USE se otorga a PUBLIC o a un grupo, o si el usuario tiene una autorización de nivel superior, tal como DBADM.

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
```

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 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).

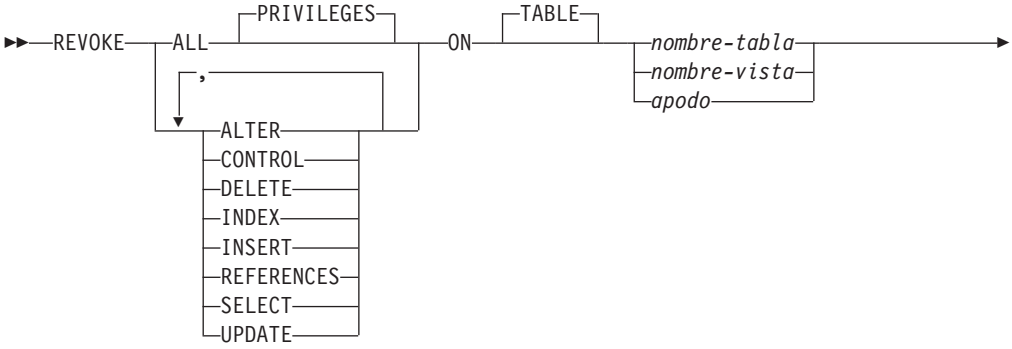
Autorización

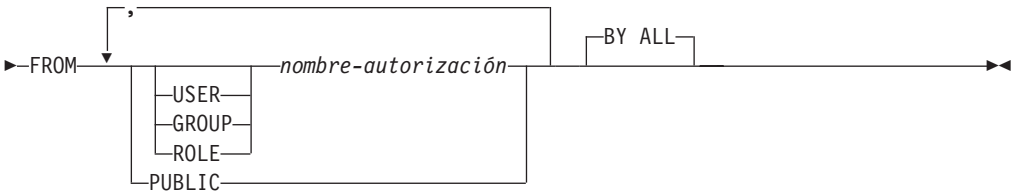
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio CONTROL sobre la tabla, vista o apodo referenciado.
- Autorización SYSADM o DBADM

Es necesaria la autorización DBADM o SYSADM para revocar el privilegio CONTROL o para revocar privilegios para tablas y vistas de catálogo.

Sintaxis





Descripción

ALL o ALL PRIVILEGES

Revoca todos los privilegios (excepto CONTROL) del nombre-autorización para las tablas, vistas o apodos especificados.

Si no se utiliza ALL, debe utilizar 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.

ROLE
Especifica que el *nombre-autorización* identifica un nombre de función.

nombre-autorización,...
Lista los ID de autorización de uno o varios usuarios, grupos o funciones.

La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca los privilegios de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Normas

- Para cada *nombre-autorización* especificado, si no se especifica USER, GROUP ni ROLE, entonces:
  - Para todas las filas del objeto especificado de las vistas de catálogo SYSCAT.TABAUTH y SYSCAT.COLAUTH en las que el otorgado es *nombre-autorización*:
    - Si todas las filas tienen un GRANTEETYPE que es 'U', se supone USER.
    - Si todas las filas tienen un GRANTEETYPE que es 'G', se supone GROUP.
    - Si todas las filas tienen un GRANTEETYPE que es 'R', se supone ROLE.
    - Si todas las filas no tienen el mismo valor para GRANTEETYPE, se devuelve un error (SQLSTATE 56092).

Notas

- Si se revoca un privilegio del *nombre-autorización* que es el propietario de la vista (como se ha registrado en la columna OWNER en SYSCAT.VIEWS), dicho privilegio también se revoca de las vistas dependientes.
- Si el propietario de la vista pierde un privilegio SELECT para algún objeto del que depende la definición de vista (o si se elimina un objeto del que depende la definición de vista o se convierte en no operativo en el caso de otra vista), la vista quedará no operativa.

  Sin embargo, si DBADM o SYSADM revoca explícitamente todos los privilegios del propietario sobre la vista, el registro del propietario (OWNER) no aparecerá en SYSCAT.TABAUTH, pero no le sucederá nada a la vista - continuará operativa.
- Los privilegios en vistas no operativas no pueden revocarse.
- Un paquete puede convertirse en no válido cuando el ID de autorización en el que se ha vinculado el paquete pierde un privilegio sobre un objeto del que depende el paquete. El privilegio puede perderse de una de las formas siguientes:
  - El privilegio se revoca desde el ID de autorización
  - El privilegio se revoca desde una función de la cual es miembro el ID de autorización
  - El privilegio se revoca desde PUBLIC

  Un paquete continúa siendo no válido hasta que se ejecuta satisfactoriamente una operación de enlace lógico o de volver a enlazar lógicamente en la aplicación, o la aplicación se ejecuta y el gestor de bases de datos vuelve a enlazar la aplicación satisfactoriamente (utilizando la información almacenada en

los catálogos). Los paquetes marcados como no válidos debido a una revocación pueden volverse a enlazar satisfactoriamente sin ninguna operación de otorgar adicional.

Por ejemplo, si un paquete propiedad de USER1 contiene SELECT de la tabla T1 y se revoca el privilegio SELECT para la tabla T1 desde USER1, el paquete se marcará como no válido. Si se vuelve a otorgar la autorización SELECT, o si el usuario tiene la autorización DBADM, el paquete se vuelve a enlazar satisfactoriamente al ejecutarse.

Otro ejemplo es un paquete propiedad de USER1, que es miembro de la función R1. El paquete contiene SELECT de la tabla T1 y se revoca el privilegio SELECT para la tabla T1 desde la función R1. El paquete se marcará como no válido, partiendo de que USER1 no tiene el privilegio SELECT para la tabla T1 por otros medios.

- Los paquetes, activadores o vistas que incluyen la utilización de OUTER(*Z*) en la cláusula FROM dependen de tener el privilegio SELECT en cada subtabla o subvista de *Z*. De modo similar, los paquetes, activadores y vistas que incluyen la utilización de DEREF(*Y*) donde *Y* es un tipo de referencia con una vista o tabla de destino *Z* dependen de tener un privilegio SELECT en cada subtabla o subvista de *Z*. Estos paquetes pueden convertirse en no válidos y las vistas o activadores pueden pasar a ser no operativos cuando el ID de autorización bajo el que se han vinculado los paquetes o el propietario de los activadores o vistas pierde el privilegio SELECT. El privilegio SELECT puede perderse de una de las formas siguientes:
  - El privilegio SELECT se revoca desde el ID de autorización
  - El privilegio SELECT se revoca desde una función de la cual es miembro el ID de autorización
  - El privilegio SELECT se revoca desde PUBLIC
- Los privilegios de tabla, vista o apodo no pueden revocarse de un *nombre-autorización* con CONTROL en el objeto sin revocar también el privilegio CONTROL (SQLSTATE 42504).
- La revocación de un privilegio específico no revoca necesariamente la posibilidad de realizar la acción. Un usuario puede seguir realizando una tarea si PUBLIC, un grupo o una función tienen otros privilegios o si el usuario tiene privilegios como, por ejemplo, ALTERIN en el esquema de una tabla o vista.
- Si el propietario de la tabla de consultas materializadas pierde un privilegio SELECT en 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 eliminará.

  Sin embargo, si DBADM o SYSADM revoca explícitamente todos los privilegios del propietario sobre la tabla de consultas materializadas, el registro de SYSTABAUTH de OWNER se suprimirá, pero no le sucederá nada a la tabla de consultas materializadas - seguirá estando operativa.
- Revocar privilegios de apodo no tiene efecto sobre los privilegios de objeto de la fuente de datos (tabla o vista).
- La revocación del privilegio SELECT para una tabla o vista a la que se hace referencia directa o indirectamente en una función de SQL o cuerpo de método podría no ejecutarse satisfactoriamente si la función de SQL o el cuerpo de método no puede eliminarse porque algún otro objeto depende de éste (SQLSTATE 42893).
- La revocación del privilegio SELECT hace que se descarte un cuerpo de método o de función SQL cuando:

- El propietario del cuerpo de método o de función SQL pierde el privilegio SELECT para algún objeto del que depende la definición de cuerpo de método o de función SQL; tenga en cuenta que el privilegio se puede perder porque se ha revocado desde PUBLIC o desde un rol del que es miembro el propietario.
- Se descarta un objeto del que depende la definición del cuerpo de método o de función SQL

Sin embargo, la revocación falla si otro objeto depende de la función o método (SQLSTATE 42893).

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
```

REVOKE (privilegios de carga de trabajo)

Este formato de la sentencia REVOKE revoca el privilegio USAGE en una carga de trabajo.

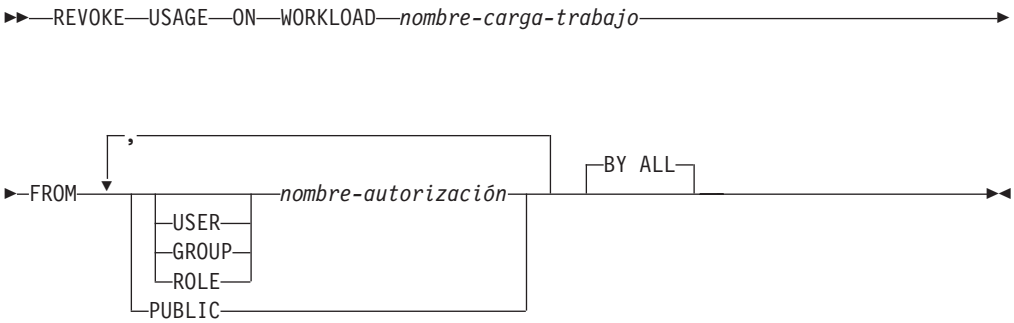
Invocación

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

Autorización

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

Sintaxis



Descripción

USAGE
Revoca el privilegio para utilizar una carga de trabajo.

ON WORKLOAD *nombre-carga-trabajo*
Identifica la carga de trabajo en la que va a revocarse el privilegio USAGE. Este nombre consta de una sola parte. El *nombre-carga-trabajo* debe identificar una carga de trabajo que exista en el servidor actual (SQLSTATE 42704). El nombre no puede ser 'SYSDEFAULTADMWORKLOAD' (SQLSTATE 42832).

FROM
Especifica a quién se le revoca el privilegio USAGE.

USER
Especifica que el *nombre-autorización* identifica a un usuario.

GROUP
Especifica que el *nombre-autorización* identifica a un grupo.

ROLE
Especifica que el *nombre-autorización* identifica una función existente en el servidor actual (SQLSTATE 42704).

nombre-autorización,...
Lista los ID de autorización de uno o más usuarios, grupos o roles. La lista de los ID de autorización no puede incluir el ID de autorización del usuario que emite la sentencia (SQLSTATE 42502).

PUBLIC
Revoca el privilegio USAGE de PUBLIC.

BY ALL
Revoca el privilegio USAGE de todos los usuarios especificados a los que se ha otorgado explícitamente ese privilegio, independientemente de qué usuario lo ha otorgado. Este es el comportamiento por omisión.

Normas

- Por cada *nombre-autorización* especificado, si no se ha especificado ninguna de las palabras clave USER, GROUP o ROLE, para todas las filas del objeto especificado en la vista de catálogos de SYSCAT.WORKLOADAUTH en la que el otorgado es *nombre-autorización*:
  - Si GRANTEETYPE es 'U', se asume USER.
  - Si GRANTEETYPE es 'G', se asume GROUP.
  - Si GRANTEETYPE es 'R', se asume ROLE.
  - Si GRANTEETYPE no tiene el mismo valor, se devuelve un error (SQLSTATE 56092.

Notas

- La sentencia REVOKE no surte efecto hasta después de que se confirme, incluso para la conexión que emite la sentencia.

Ejemplo

Revoque el privilegio para utilizar la carga de trabajo CAMPAIGN del usuario LISA.

```
REVOKE USAGE ON WORKLOAD CAMPAIGN FROM USER LISA
```

REVOKE (privilegios de objeto XSR)

Este formato de la sentencia REVOKE otorga el privilegio USAGE sobre un objeto XSR.

Invocación

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

Autorización

Se necesita una de las autorizaciones siguientes:

- SYSADM o DBADM

Sintaxis

Descripción

ON XSROBJECT *nombre-objetoxsr*
Este nombre identifica el objeto XSR sobre el que se ha revocado el privilegio USAGE. El *nombre-objetoxsr* (incluido el calificador de esquema implícito o explícito) debe designar de forma exclusiva un objeto XSR existente en el servidor actual. Si no existe ningún objeto XSR con este nombre en el esquema especificado, se producirá un error (SQLSTATE 42704).

FROM PUBLIC
Revoca el privilegio USAGE de PUBLIC.

BY ALL
Revoca todos los privilegios indicados de todos los usuarios especificados a los que se han otorgado explícitamente tales privilegios, independientemente de qué usuario los ha otorgado. Este es el comportamiento por omisión.

Ejemplo

Revoca los privilegios de uso sobre el esquema XML MYSCHEMA de PUBLIC:

```
REVOKE USAGE ON XSROBJECT MYSCHEMA FROM PUBLIC
```

ROLLBACK

La sentencia ROLLBACK se utiliza para restituir los cambios que se han hecho en la base de datos dentro de una unidad de trabajo o punto de salvaguarda.

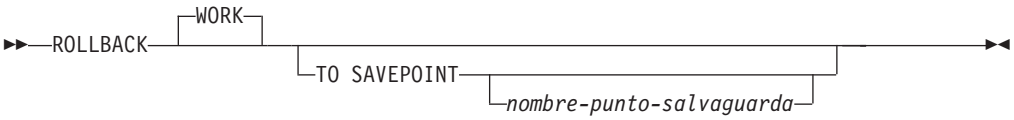
Invocación

Esta sentencia puede incorporarse a 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

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.

La modificación de los valores de variables globales no está bajo el control de la transacción. Las sentencias ROLLBACK no afectan a los valores asignados a las variables globales.

TO SAVEPOINT
Especifica que debe realizarse una retrotracción parcial (ROLLBACK TO SAVEPOINT). Si no hay ningún punto de salvaguarda activo en el nivel del punto de salvaguarda actual (consulte la sección “Normas” de la descripción de la sentencia SAVEPOINT), se devuelve un error (SQLSTATE 3B502). Tras una retrotracción satisfactoria, el punto de salvaguarda continúa existiendo, pero se liberan los puntos de salvaguarda anidados y ya no existen. Se considera que los puntos de salvaguarda anidados, si existen, se han retrotraído y después liberado como parte de la retrotracción hasta el punto de salvaguarda actual. Si no se facilita un *nombre-puntosalvaguarda*, la retrotracción se produce hasta el punto de salvaguarda definido más recientemente dentro del nivel de punto de salvaguarda actual.

Si se omite esta cláusula, la sentencia ROLLBACK retrotrae toda la transacción. Además, se liberan todos los puntos de salvaguarda definidos dentro de la transacción.

nombre-punto-salvaguarda
Especifica el punto de salvaguarda que se debe utilizar en la operación de retrotracción. El *nombre-puntosalvaguarda* especificado no puede empezar por 'SYS' (SQLSTATE 42939). Tras una operación de retrotracción satisfactoria, el punto de salvaguarda nombrado continúa existiendo. Si el nombre de punto de salvaguarda no existe, se devuelve un error (SQLSTATE 3B001). Se deshacen los cambios que se han hecho en datos y esquemas desde que se definió el punto de salvaguarda.

Notas

- Cuando se ejecuta un ROLLBACK de la unidad de trabajo se liberan todos los bloqueos retenidos. 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
```

SAVEPOINT

Utilice la sentencia SAVEPOINT para definir un punto de salvaguarda dentro de una transacción.

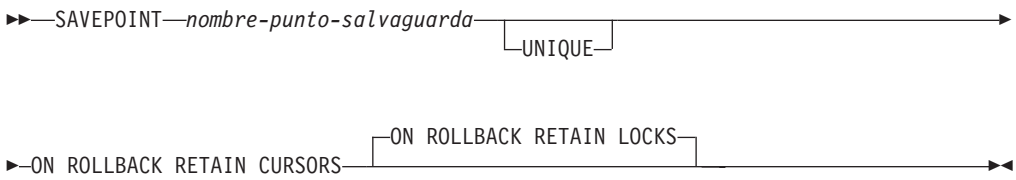
Invocación

Esta sentencia puede incorporarse en un programa de aplicación (también en procedimientos almacenados) o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-punto-salvaguarda
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 tiene la intención de volver a utilizar este nombre de punto de salvaguarda mientras la salvaguarda esté activa en el nivel 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 es posible, los cursores no se ven afectados por una retrotracción en una operación de salvaguarda. Para situaciones en las que los cursores se vean afectados por la retrotracción en 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.

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

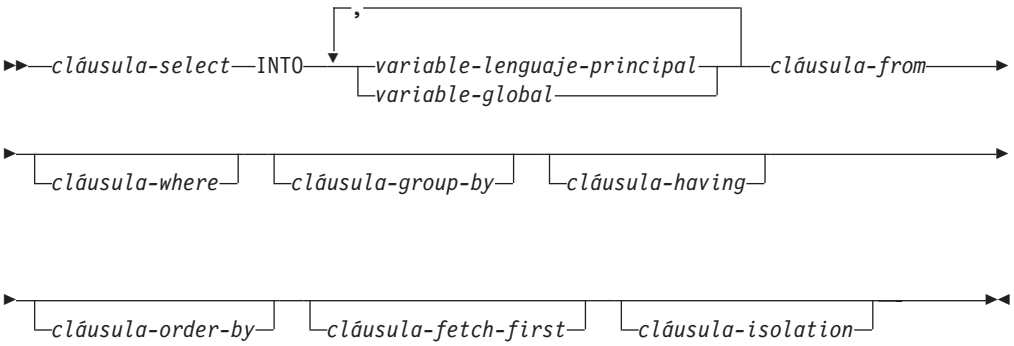
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Privilegio SELECT para la tabla, vista o apodo
- Privilegio CONTROL sobre 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 INTO es un apodo, los privilegios para el objeto en la fuente de datos no se toman en consideración hasta que se ejecuta la sentencia en la fuente de datos. En ese momento, el ID de autorización utilizado para conectarse a la fuente de datos debe disponer de los privilegios necesarios a fin de realizar la operación con el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis



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 o variables globales.

variable-lenguaje-principal
Identifica una variable que está descrita en el programa bajo las normas para la declaración de variables del lenguaje principal.

variable-global
Identifica una variable global que existe en el servidor actual.

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 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 ninguna variable.

Normas

- Las variables globales no pueden ser asignadas en activadores, funciones, métodos o sentencias compuestas dinámicas o en un procedimiento al que uno de estos objetos llame de forma directa o indirecta (SQLSTATE 428GX)

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 };
```

Ejemplo 4: este ejemplo C coloca el salario máximo de la tabla EMP en la variable global GV_MAXSALARY.

```
EXEC SQL SELECT MAX(SALARY)
  INTO GV_MAXSALARY
  FROM EMP;
```

SELECT

La sentencia SELECT es una forma de consulta. Puede incorporarse en un programa de aplicación o emitirse interactivamente.

SET COMPILATION ENVIRONMENT

La sentencia SET COMPILATION ENVIRONMENT cambia el entorno de compilación actual en la conexión para que coincida con los valores incluidos en el entorno de compilación proporcionado por un supervisor de sucesos. Esta sentencia cambia los valores de uno o más registros especiales; estos cambios, a su vez, afectarán a la compilación de cualquier sentencia de SQL dinámico subsiguiente.

La sentencia no está bajo el control de la transacción.

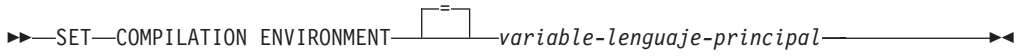
Invocación

La sentencia puede incorporarse en un programa de aplicación. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

variable-lenguaje-principal
Una variable de tipo BLOB que contiene un entorno de compilación proporcionado por un supervisor de sucesos. No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815). Si el formato del entorno de compilación es incorrecto, se devolverá un error y los valores de la conexión permanecerán sin modificar (SQLSTATE 51040).

Notas

- Para restablecer el entorno de compilación en los valores por omisión originales, termine y luego reinicie la conexión. Puede conseguir el mismo efecto emitiendo esta sentencia dentro de una rutina de SQL, de modo que ningún cambio realizado en los registros especiales se refleje en la conexión al volver de esa rutina.
- Utilice la función de tabla COMPILATION_ENV a fin de observar los elementos individuales que contiene el entorno de compilación.

Ejemplos

Ejemplo 1: Establecer el entorno de compilación de la sesión actual en los valores incluidos en un entorno de compilación capturado anteriormente por un supervisor de sucesos de punto muerto. Un supervisor de sucesos de punto muerto que se ha creado especificando la opción WITH DETAILS HISTORY capturará el entorno de compilación de las sentencias de SQL dinámico. Este entorno capturado es el que se acepta como entrada en la sentencia.

```
SET COMPILATION ENVIRONMENT = :hv1
```

SET CONNECTION

La sentencia SET CONNECTION cambia el estado de una conexión de inactivo a actual, convirtiendo la ubicación especificada en el servidor actual. No está bajo el control de la transacción.

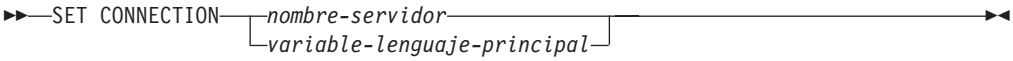
Invocación

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.

Sintaxis



Descripción

nombre-servidor **o** *variable-lenguaje-principal*

Identifica el servidor de aplicaciones mediante el *nombre-servidor* especificado o mediante una *variable-lenguaje-principal* que contenga el *nombre-servidor*.

Si se especifica una *variable-lenguaje-principal*, debe ser una variable de serie de caracteres con un atributo de longitud que no sea mayor que 8, y no debe contener una variable indicadora. El *nombre-servidor* contenido en la *variable-lenguaje-principal* 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-lenguaje-principal* debe identificar una conexión existente del proceso de aplicación. Si no identifican ninguna conexión existente, se genera un error (SQLSTATE 08003).

Si SET CONNECTION se aplica a la conexión actual, no se cambian los estados de ninguna de las conexiones del proceso de aplicación.

Conexión satisfactoria

Si la sentencia SET CONNECTION se ejecuta satisfactoriamente:

- No se realiza ninguna conexión. El registro especial CURRENT SERVER se actualiza con el *nombre-servidor* especificado.
- La conexión actual previa, si la hay, se coloca en el estado inactivo (suponiendo que se haya especificado un *nombre-servidor* distinto).
- El registro especial CURRENT SERVER y la SQLCA se actualizan de la misma forma que con “CONNECT (Tipo 1)”, donde encontrará información adicional.

Conexión no satisfactoria

Si la sentencia SET CONNECTION falla:

- 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 "Opciones que rigen la semántica de la unidad de trabajo distribuida") no impide utilizar SET CONNECTION, pero la sentencia no es necesaria, porque en su lugar se pueden utilizar sentencias CONNECT de tipo 2.
- Cuando se utiliza una conexión, se inactiva y después se restaura al estado actual en la misma unidad de trabajo, dicha conexión refleja su última utilización por el proceso de aplicación en relación con el estado de bloqueos, cursores y sentencias preparadas.

Ejemplos

Ejecute las sentencias de SQL de IBMSTHDB, ejecute las sentencias de SQL de IBMTOKDB y después ejecute más sentencias de SQL de IBMSTHDB.

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
EXEC SQL CONNECT TO IBMTOKDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMTOKDB */
EXEC SQL SET CONNECTION IBMSTHDB;
/* Ejecuta las sentencias que hacen referencia a objetos de IBMSTHDB */
```

Observe que la primera sentencia CONNECT crea la conexión IBMSTHDB, la segunda sentencia CONNECT lo coloca en el estado inactivo y la sentencia SET CONNECTION la devuelve al estado actual.

SET CURRENT DECFLOAT ROUNDING MODE

La sentencia SET CURRENT DECFLOAT ROUNDING MODE verifica que la modalidad de redondeo especificada sea el valor que está establecido actualmente para el registro especial CURRENT DECFLOAT ROUNDING MODE.

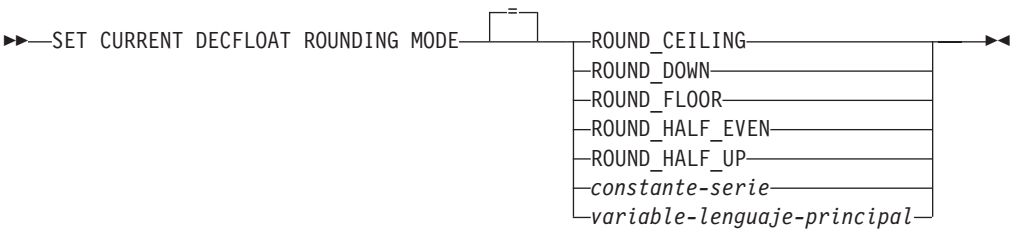
Invocación

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

Autorización

No se necesita.

Sintaxis



Descripción

ROUND_CEILING
Redondea el valor hacia el infinito positivo. Si todos los dígitos descartados son cero o si el signo es negativo, el resultado no cambia (a excepción de la eliminación de los dígitos descartados). De lo contrario, el coeficiente de resultado se incrementa en 1.

ROUND_DOWN
Redondea el valor hacia 0 (truncamiento). Se ignoran los dígitos descartados.

ROUND_FLOOR
Redondea el valor hacia el infinito negativo. Si todos los dígitos descartados son cero o si el signo es positivo, el resultado no cambia (a excepción de la eliminación de los dígitos descartados). De lo contrario, el signo es negativo y el coeficiente de resultado se incrementa en 1.

ROUND_HALF_EVEN
Redondea el valor al valor más próximo. Si los valores son equidistantes, redondea el valor de forma que el dígito final sea par. Si los dígitos descartados representan más de la mitad del valor de un número en la siguiente posición izquierda, el coeficiente de resultado se incrementa en 1. Si representan menos de la mitad, el coeficiente de resultado no se ajusta (es decir, se ignoran los dígitos descartados). De lo contrario, el coeficiente de resultado no se modifica si el dígito situado más a la derecha es par o se incrementa en 1 si el dígito situado más a la derecha es impar (para convertirlo en dígito par).

ROUND_HALF_UP
Redondea el valor al valor más próximo. Si los valores son equidistantes,

redondea el valor por exceso. Si los dígitos descartados representan la mitad o más de la mitad del valor de un número en la siguiente posición izquierda, el coeficiente de resultado se incrementa en 1. De lo contrario, los dígitos descargados se ignoran.

constante-serie
Constante de serie de caracteres con una longitud máxima de 15 bytes, después de que se hayan eliminado los espacios en blanco de cola. El valor debe ser una serie ajustada por la izquierda que especifique una de las cinco palabras clave de modalidad de redondeo (no sensibles a las mayúsculas y minúsculas).

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. El valor de la variable de lenguaje principal debe ser una serie ajustada por la izquierda que especifique una de las cinco palabras clave de modalidad de redondeo (no sensibles a las mayúsculas y minúsculas). La longitud real del contenido de *variable-lenguaje-principal* no debe tener más de 15 bytes, después de que se hayan eliminado los espacios en blanco de cola. El valor se debe rellenar a la derecha con espacios de blanco cuando se utilice una variable de lenguaje principal de caracteres de longitud fija. La variable del lenguaje principal no se puede establecer en el valor nulo.

Normas

- El valor de modalidad de redondeo especificado debe ser el mismo valor que el del registro especial CURRENT DECFLOAT ROUNDING MODE (SQLSTATE 42815).

Notas

- Esta sentencia no cambia el valor del registro especial CURRENT DECFLOAT ROUNDING MODE en un servidor DB2 para Linux, UNIX y Windows. Sin embargo, cuando un servidor DB2 para z/OS o un servidor DB2 para i5/OS procesa la sentencia, ésta se puede utilizar para cambiar el valor del registro especial CURRENT DECFLOAT ROUNDING MODE en dicho servidor.

Ejemplo

Ejemplo 1: La sentencia siguiente verifica si el valor de modalidad de redondeo especificado para el cliente coincide con el valor de modalidad de redondeo que está establecido actualmente en el servidor.

```
SET CURRENT DECFLOAT ROUNDING MODE = ROUND_CEILING
```

SET CURRENT DEFAULT TRANSFORM GROUP

La sentencia SET CURRENT DEFAULT TRANSFORM GROUP cambia el valor del registro especial CURRENT DEFAULT TRANSFORM GROUP. La sentencia no está bajo el control de la transacción.

Invocación

Esta sentencia puede incorporarse a 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

No se necesita.

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 128 bytes de longitud (SQLSTATE 42815). Cuando se establece el registro especial, no se realiza ninguna validación de que el *nombre-grupo* esté definido para cualquier tipo estructurado. Sólo se comprueba la validez de la definición del grupo de transformación mencionado cuando se referencia explícitamente un tipo estructurado.

Normas

- Si el valor especificado no se adapta a las normas para un *nombre-grupo*, se emite un error (SQLSTATE 42815)
- Las funciones TO SQL y FROM SQL definidas en el grupo de transformación *nombre-grupo* se utilizan para intercambiar datos de tipo estructurado, definidos por el usuario, con un programa del lenguaje principal.

Notas

- El valor inicial del registro especial CURRENT DEFAULT TRANSFORM GROUP es la serie de caracteres vacía.

Ejemplos

Ejemplo 1: Establecimiento del grupo de transformación por omisión en MYSTRUCT1. Este ejemplo utiliza las funciones TO SQL y FROM SQL definidas en el grupo de transformación MYSTRUCT1 para intercambiar variables de tipo estructurado, definidas por el usuario, con el programa actual del lenguaje principal.

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

SET CURRENT DEGREE

La sentencia SET CURRENT DEGREE asigna un valor al registro especial CURRENT DEGREE. La sentencia no está bajo el control de la transacción.

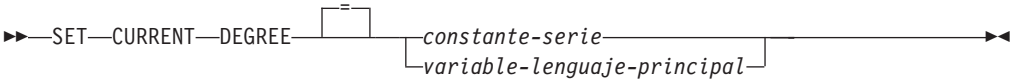
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

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-lenguaje-principal
El tipo de datos de la *variable-lenguaje-principal* 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-lenguaje-principal* 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 **intra_parallel** 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.

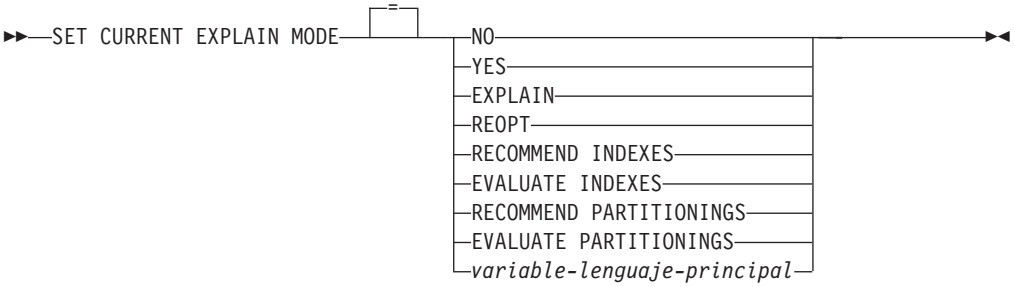
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

NO
Inhabilita el recurso Explain. No se captura la información de Explain. NO es el valor inicial del registro especial.

YES
Habilita el recurso Explain y provoca que la información de Explain se inserte en las tablas de Explain para las sentencias de SQL dinámico elegibles. Todas las sentencias de SQL dinámico se compilan y ejecutan normalmente.

EXPLAIN
Habilita el recurso Explain y provoca que se capte la información de Explain para cualquier sentencia de SQL 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 están disponibles los valores actuales de las variables del lenguaje principal, los registros especiales, las variables globales o los marcadores de parámetro.

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 de base de datos para cada tabla a la que accede una consulta específica. Las mejores particiones de base de datos se graban en una tabla ADVISE_PARTITION. La consulta no se ejecuta.

EVALUATE PARTITIONINGS
Especifica que el compilador debe obtener el rendimiento estimado de una consulta utilizando las particiones de base de datos virtuales especificadas en la tabla ADVISE_PARTITION.

variable-lenguaje-principal
El tipo de datos de la *variable-lenguaje-principal* debe ser CHAR o VARCHAR y su longitud no debe ser mayor que 254. Si se proporciona un campo más largo, se devolverá un error (SQLSTATE 42815). El valor especificado debe ser NO, YES, EXPLAIN, RECOMMEND INDEXES o EVALUATE INDEXES. Si el valor real que se proporciona es mayor que el valor de sustitución especificado, la entrada se debe rellenar por la derecha con blancos. Los blancos iniciales no están permitidos (SQLSTATE 42815). Todos los valores de entrada se tratan como si no fuesen sensibles a las mayúsculas y minúsculas. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable no debe indicar un valor nulo (SQLSTATE 42815).

Notas

- El recurso Explain utiliza los ID siguientes como el esquema al calificar tablas Explain que se están llenando de datos:
  - El ID de autorización de sesión SQL dinámico
  - El ID de autorización de sentencia para SQL estático

  El esquema se puede asociar con un conjunto de tablas Explain o alias que apuntan a un conjunto de tablas Explain en un esquema distinto. Si no se encuentran tablas Explain en el esquema, el recurso Explain comprueba la existencia de tablas Explain en el esquema SYSTOOLS e intenta utilizar dichas tablas.
- La información de Explain para las sentencias de SQL estático se puede capturar utilizando la opción EXPLAIN del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLAIN y el valor de registro 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
```

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.

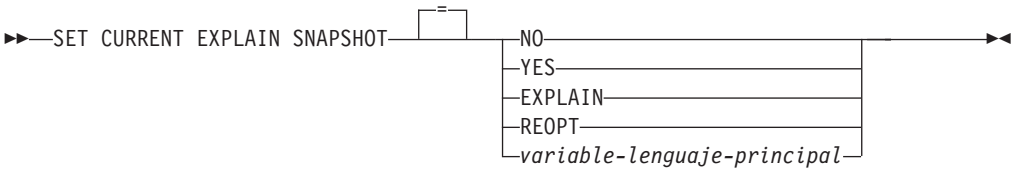
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

NO
Inhabilita el recurso 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 están disponibles los valores actuales de las variables del lenguaje principal, los registros especiales, las variables globales o los marcadores de parámetro.

variable-lenguaje-principal
El tipo de datos de la *variable-lenguaje-principal* debe ser CHAR o VARCHAR y la longitud de su contenido no debe ser mayor que 8. Si 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-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Notas

- El recurso Explain utiliza los ID siguientes como el esquema al calificar tablas Explain que se están llenando de datos:
  - El ID de autorización de sesión SQL dinámico
  - El ID de autorización de sentencia para SQL estático

  El esquema se puede asociar con un conjunto de tablas Explain o alias que apuntan a un conjunto de tablas Explain en un esquema distinto. Si no se encuentran tablas Explain en el esquema, el recurso Explain comprueba la existencia de tablas Explain en el esquema SYSTOOLS e intenta utilizar dichas tablas.
- Las instantáneas de Explain para las sentencias de SQL estático se pueden capturar utilizando la opción EXPLSNAP del mandato PREP o BIND. Si se especifica el valor ALL de la opción EXPLSNAP y el valor de registro CURRENT EXPLAIN SNAPSHOT es NO, se capturarán instantáneas de Explain para las sentencias de SQL dinámico en tiempo de ejecución. Si el valor del registro CURRENT EXPLAIN SNAPSHOT no es NO, la opción EXPLSNAP se pasará por alto.
- Si se activa el recurso de instantánea de Explain, el ID de autorización actual debe tener el privilegio INSERT para las tablas de Explain o se genera un error (SQLSTATE 42501).
- Cuando las sentencias de SQL se explican a partir de una rutina, la rutina debe definirse con un indicador de acceso a datos de SQL de MODIFIES SQL DATA (SQLSTATE 42985).
- Si el registro especial se establece en REOPT, y la sentencia de SQL no está 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;
```

SET CURRENT FEDERATED ASYNCHRONY

La sentencia SET CURRENT FEDERATED ASYNCHRONY asigna un valor al registro especial CURRENT FEDERATED ASYNCHRONY. No está bajo el control de la transacción.

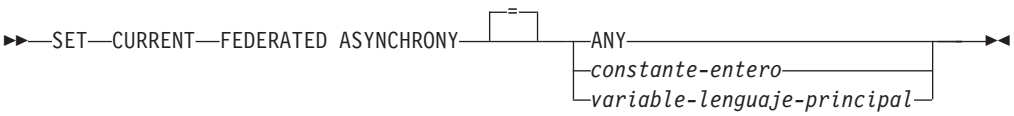
Invocación

La sentencia se puede incorporar a un programa de aplicación o emitir a través del uso de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

ANY
Especifica el valor -1 para CURRENT FEDERATED ASYNCHRONY, lo que significa que la ejecución de sentencias puede implicar asincronía al utilizar un grado determinado por el gestor de bases de datos.

constante-entera
Especifica un valor de entero entre 0 y 32.767, incluido. La ejecución de sentencias puede implicar asincronía al utilizar el grado especificado. Si el valor es 0 cuando una sentencia de SQL se prepara dinámicamente, la ejecución de esa sentencia no utilizará la asincronía.

variable-lenguaje-principal
Una variable de tipo INTEGER. El valor debe estar entre 0 y 32.767, incluido, o -1 (representa ANY). Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Notas

- El grado de asincronía para las sentencias de SQL estático se puede controlar utilizando la opción FEDERATED_ASYNCHRONY del mandato PREP o BIND.
- El valor inicial del registro especial CURRENT FEDERATED ASYNCHRONY lo determina el parámetro **asincronía_federada** de configuración del gestor de bases de datos si la sentencia dinámica se emite a través del procesador de línea de mandatos (CLP). La opción de vinculación FEDERATED_ASYNCHRONY determina el valor inicial si la sentencia dinámica forma parte de una aplicación que se está vinculando.

Ejemplos

Ejemplo 1: La sentencia siguiente inhabilita la asincronía, estableciendo en 0 el valor del registro especial CURRENT FEDERATED ASYNCHRONY.

```
SET CURRENT FEDERATED ASYNCHRONY = 0
```

Ejemplo 2: La sentencia siguiente establece en 5 el grado de asincronía.

```
SET CURRENT FEDERATED ASYNCHRONY 5
```

Ejemplo 3: La sentencia siguiente establece en -1 el valor del registro especial CURRENT FEDERATED ASYNCHRONY, lo que especifica que el gestor de bases de datos debe determinar el grado de asincronía.

```
SET CURRENT FEDERATED ASYNCHRONY ANY
```

SET CURRENT IMPLICIT XMLPARSE OPTION

La sentencia SET CURRENT IMPLICIT XMLPARSE OPTION cambia el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION. La sentencia no está bajo el control de la transacción.

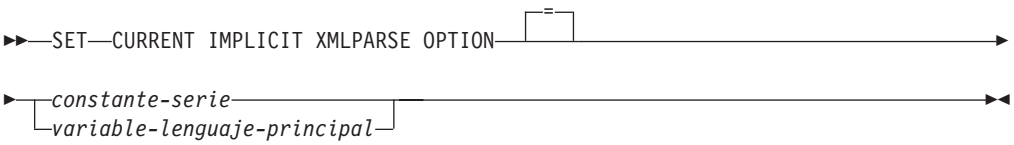
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

constante-serie
Constante de serie de caracteres. El valor debe ser una serie justificada a la izquierda que sea 'PRESERVE WHITESPACE' o 'STRIP WHITESPACE' (no sensible a mayúsculas ni minúsculas) sin caracteres adicionales en blanco entre las palabras clave.

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. El valor de la variable del lenguaje principal debe ser una serie justificada a la izquierda que sea 'PRESERVE WHITESPACE' o 'STRIP WHITESPACE' (no sensible a mayúsculas ni minúsculas) sin caracteres adicionales en blanco entre las palabras clave. El valor debe rellenarse con blancos por la derecha utilizando una *variable-lenguaje-principal* de carácter de longitud fija. La variable del lenguaje principal no puede ser nulo.

Notas

- El valor inicial del registro especial CURRENT IMPLICIT XMLPARSE OPTION es 'STRIP WHITESPACE'.
- Las sentencias de SQL dinámico y estático se ven afectadas por este registro especial.

Ejemplo

Establezca el valor del registro especial CURRENT IMPLICIT XMLPARSE OPTION en 'PRESERVE WHITESPACE'.

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

SET CURRENT ISOLATION

La sentencia SET CURRENT ISOLATION asigna un valor al registro especial CURRENT ISOLATION. La sentencia no está bajo el control de la transacción.

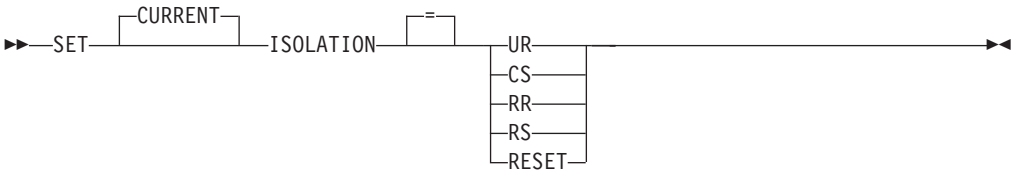
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

El valor del registro especial CURRENT ISOLATION se sustituye por el valor especificado o se establece en blancos si se especifica RESET.

Notas

- *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

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.

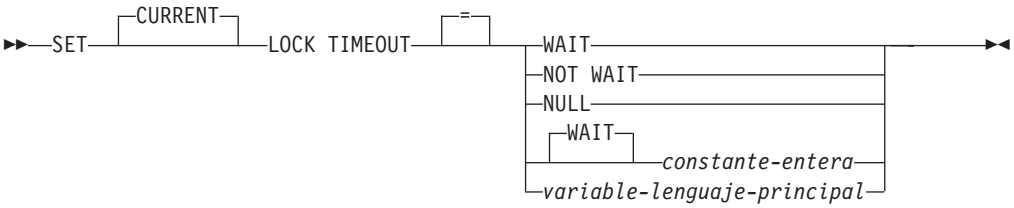
Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

El valor especificado debe ser un entero entre -1 y 32767, inclusive (SQLSTATE 428B7) o el valor nulo.

WAIT
Especifica un valor CURRENT LOCK TIMEOUT de -1, lo que significa que el gestor de bases de datos debe esperar hasta que se libere el bloqueo o se detecte un punto muerto (SQLSTATE 40001 o 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-entera*
Especifica un valor de entero entre -1 y 32767. Un valor de -1 es equivalente a especificar la palabra clave WAIT sin un valor entero. Un valor de 0 es equivalente a especificar la cláusula NOT WAIT. Si el valor está entre 1 y 32767, el gestor de bases de datos esperará ese número de segundos (si no se puede obtener un bloqueo) antes de devolver un error (SQLSTATE 40001 ó SQLSTATE 57033).

variable-lenguaje-principal
Una variable de tipo INTEGER. El valor debe estar entre -1 y 32767. Si *variable-lenguaje-principal* tiene una variable de indicador asociada y su valor

especifica un valor nulo, se desestablece el valor CURRENT LOCK TIMEOUT. Es equivalente a especificar la palabra clave NULL.

Notas

- Un valor actualizado del registro especial surte efecto inmediatamente después de la ejecución satisfactoria de esta sentencia. Puesto que el valor de registro especial que se debe utilizar durante la ejecución de la sentencia se fija al principio de la ejecución de la sentencia, sólo devolverán un valor actualizado del registro especial CURRENT LOCK TIMEOUT las sentencias que inician la ejecución después de que la sentencia SET LOCK TIMEOUT se haya completado satisfactoriamente.
- ***Compatibilidades***
  - Para mantener la compatibilidad con Informix:
    - Se puede especificar MODE en lugar de TIMEOUT.
    - Se puede especificar TO en lugar del operador igual (=).

Ejemplos

Ejemplo 1: Establecer el valor de tiempo de espera excedido para que espere 30 segundos antes de devolver un error.

```
SET CURRENT LOCK TIMEOUT 30
```

Ejemplo 2: Desestablecer 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
```

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.

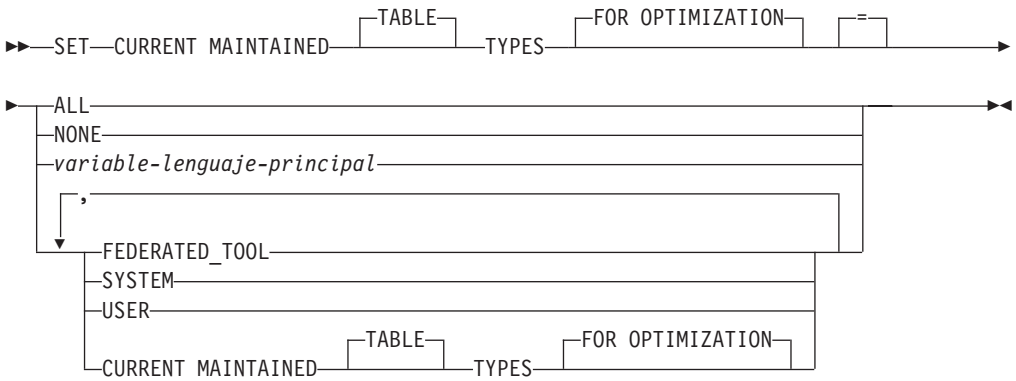
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

ALL
Especifica que todos los tipos posibles de tablas mantenidas que este registro especial controla, ahora y en el futuro, van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

NONE
Especifica que ninguno de los tipos de objetos que este registro especial controla van a considerarse cuando se optimice el proceso de las consultas de SQL dinámico.

FEDERATED_TOOL
Especifica que las tablas de 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-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la variable del lenguaje principal no debe exceder de 254 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de *variable-lenguaje-principal* deben estar justificados por la izquierda. El contenido de la *variable-lenguaje-principal* debe ser una serie que forme una lista de palabras clave separadas por comas y que coincidan con lo que se puede especificar como palabras claves para el registro especial. Estas palabras clave se deben especificar exactamente en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas. El valor deberá rellenarse con blancos por la derecha si su longitud es menor que la de la variable del lenguaje principal.

Notas

- El valor inicial del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION es SYSTEM.
- El registro especial CURRENT REFRESH AGE debe establecerse en un valor distinto de cero para que los tipos de tablas que se han especificado se consideren cuando se optimice el proceso de las consultas de SQL dinámico.

Ejemplos

Ejemplo 1: Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION SYSTEM = USER
```

Ejemplo 2: Recupere el valor actual del registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION en una variable del lenguaje principal denominada CURMAINTYPES.

```
EXEC SQL VALUES (CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION)
  INTO :CURMAINTYPES
```

Ejemplo 3: Establezca el registro especial CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION de modo que no tenga ningún valor.

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = NONE
```

SET CURRENT MDC ROLLOUT MODE

La sentencia SET CURRENT MDC ROLLOUT MODE asigna un valor al registro especial CURRENT MDC ROLLOUT MODE. El valor especifica el tipo de borrado de lanzamiento que va a realizarse sobre sentencias DELETE de calificación para tablas de clúster de varias dimensiones (MDC).

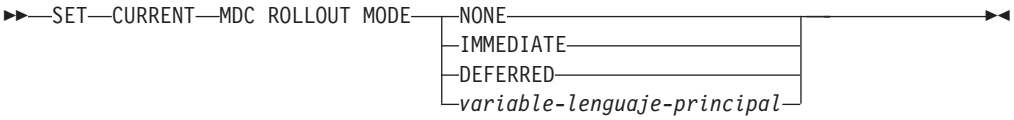
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

NONE
Especifica que no va a utilizarse la optimización de clasificación de MDC durante las operaciones de supresión. La sentencia DELETE se procesa del mismo modo que una sentencia DELETE que no se califica para la clasificación.

IMMEDIATE
Especifica que la optimización de clasificación de MDC va a utilizarse si se califica la sentencia DELETE. Si la tabla tiene índices RID, los índices se actualizan inmediatamente durante el proceso de supresión. Los bloques suprimidos pueden volverse a utilizar después de que se confirme la transacción.

DEFERRED
Especifica que la optimización de clasificación de MDC va a utilizarse si se califica la sentencia DELETE. Si la tabla tiene índices RID, las actualizaciones de los índices se difieren hasta que hayan acabado las confirmaciones de las transacciones. Con esta opción, la supresión de procesos es más rápida y utiliza menos espacio de archivo de anotaciones, pero los bloques suprimidos no pueden volver a utilizarse hasta que se completen las actualizaciones de índice.

variable-lenguaje-principal
Una variable de tipo VARCHAR. La longitud *variable-lenguaje-principal* debe ser igual o inferior a 17 bytes (SQLSTATE 42815). El valor de la variable del lenguaje principal debe ser una serie justificada a la izquierda es decir una serie 'NONE', 'IMMEDIATE' o 'DEFERRED' (no sensible a mayúsculas ni minúsculas). Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Notas

- Las sucesivas sentencias DELETE que pueden elegirse para el proceso de lanzamiento respecto del valor del registro especial CURRENT MDC ROLLOUT MODE. En la actualidad las sesiones en ejecución no resultan afectadas por un cambio en este registro especial.
- Los efectos de la ejecución de la sentencia SET CURRENT MDC ROLLOUT MODE no se retrotraen en el caso de que se retrotraiga la unidad de trabajo en la que se ejecuta la sentencia.

Ejemplo

Especifique comportamiento de borrado diferido para la siguiente sentencia DELETE que se califique para el proceso de lanzamiento.

```
SET CURRENT MDC ROLLOUT MODE IMMEDIATE
```

SET CURRENT OPTIMIZATION PROFILE

La sentencia SET CURRENT OPTIMIZATION PROFILE asigna un valor al registro especial CURRENT OPTIMIZATION PROFILE. El valor especifica el perfil de optimización que el optimizador debe utilizar al preparar sentencias de DML dinámico. La sentencia no está bajo el control de la transacción.

Cuando se evalúa la sentencia, se comprueba la validez del nombre del perfil de optimización, pero el perfil no se procesa hasta que el optimizador se encuentra con una sentencia de DML dinámico.

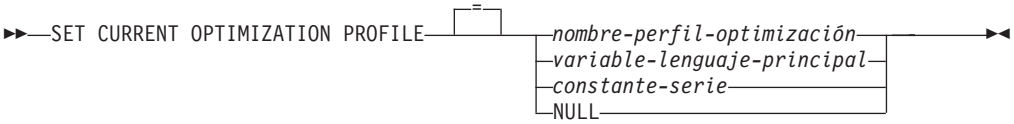
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-perfil-optimización
Es el nombre de dos parte del perfil de optimización. El nombre se puede especificar con un literal, una variable de lenguaje principal o un registro especial. El nombre especificado es el nombre que se entra en el registro especial CURRENT OPTIMIZATION PROFILE.

Si el nombre-perfil-optimización especificado no está calificado, se utilizará el valor del registro CURRENT DEFAULT SCHEMA como calificador implícito. El valor por omisión del registro especial es nulo.

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR que incluye el nombre del perfil de optimización. Un variable de lenguaje principal que incluye un indicador nulo indica que se debe utilizar el valor de la opción de enlace OPTPROFILE si dicho valor se especifica para el paquete actual. Una variable de lenguaje principal de longitud cero, o con sólo un espacio en blanco, indica que no se debe utilizar ningún perfil de optimización.

La variable de lenguaje principal debe cumplir las especificaciones siguientes:

- El contenido de la serie es un identificador de una parte o de dos partes (separado por un punto), sin espacios en blanco iniciales.
- El identificador o los identificadores pueden ser delimitados o no delimitados.
- El contenido de la serie no se convierte a mayúsculas.

- En las series no delimitadas no se pueden utilizar caracteres en minúscula ni caracteres especiales.
- Si el primer carácter es una comilla doble, deberá haber una comilla doble de cierre antes de un punto o como último carácter (exceptuando los espacios en blanco) de la serie.
- Si el primer carácter a continuación de un punto es una comilla doble, deberá haber una comilla doble de cierre como último carácter (exceptuando los espacios en blanco) de la serie.
- Si el identificador está delimitado, para incluir comillas dobles en el identificador se deberá especificar el carácter dos veces.
- Cualquier punto que no esté dentro de un identificador delimitado se tratará como un separador y sólo puede existir un punto separador en la serie.

constante-serie
Especifica una constante como serie de caracteres del nombre del perfil de optimización. El contenido de una constante de tipo serie debe cumplir las mismas especificaciones que una variable de lenguaje principal.

NULL
Establece en nulo el registro CURRENT OPTIMIZATION PROFILE.

La Tabla 34 proporciona ejemplos de literales de serie e identificadores que se pueden utilizar para asignar el registro de acuerdo con las normas de denominación del perfil de optimización. El valor de la columna SCHEMA y NAME representa un nombre de perfil de optimización tal como aparecería en la tabla OPT_PROFILE. La columna de Literales de serie válidos muestra los literales de serie que coinciden con el perfil de optimización al que dan nombre los correspondientes valores de la columna SCHEMA y NAME. La columna Identificadores válidos muestra los identificadores que identificarían este mismo perfil de optimización.

Tabla 34. Ejemplos de literales de serie e identificadores

SCHEMA	NAME	Literales de serie válidos	Identificadores válidos
SIMMEN	BIG_PROF	'BIG_PROF' 'SIMMEN.BIG_PROF' '"BIG_PROF"' '"SIMMEN"."BIG_PROF"'	BIG_PROF SIMMEN.BIG_PROF "BIG_PROF" "SIMMEN"."BIG_PROF"
SIMMEN	low_profile	'"low_profile"' 'SIMMEN."low_profile"' '"SIMMEN"."low_profile"'	"low_profile" SIMMEN."low_profile" "SIMMEN"."low_profile"
eliaz	DBA3	'DBA3' '"DBA3"' '"eliaz".DBA3' '"eliaz"."DBA3"'	DBA3 "eliaz".DBA3 "eliaz"."DBA3"
SNOW	PROFILE1.0	'"PROFILE1.0"' 'SNOW."PROFILE1.0"' '"SNOW"."PROFILE1.0"'	"PROFILE1.0" SNOW."PROFILE1.0" "SNOW"."PROFILE1.0"

Notas

- Si el valor del registro especifica el nombre de un perfil de optimización existente, el perfil de optimización especificado se utilizará al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es nulo, especifica el nombre de un perfil de optimización existente, el perfil de optimización especificado por la opción de enlace OPTPROFILE, si existe, se utilizará al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es nulo y no se ha establecido la opción de enlace OPTPROFILE, no se utilizará ningún perfil de optimización al preparar posteriores sentencias de DML dinámico.
- Si el valor del registro es una serie vacía (“”), no se utilizará ningún perfil de optimización al preparar posteriores sentencias de DML dinámico, independientemente de si la opción de enlace OPTPROFILE se ha establecido o no.
- Los cambios posteriores de CURRENT DEFAULT SCHEMA no afectan al perfil de optimización. El valor de registro CURRENT OPTIMIZATION PROFILE se establece con el nombre de dos partes que está en vigor al mismo tiempo que se evalúa la sentencia SET CURRENT OPTIMIZATION PROFILE. Sólo otra sentencia SET CURRENT OPTIMIZATION PROFILE puede cambiar el perfil de optimización que se utiliza.

Ejemplos

Ejemplo 1: El perfil de optimización RICK.FOO se utiliza para las sentencias 1, 2 y 3. TOM.FOO se utiliza para la sentencia 4.

```
SET CURRENT SCHEMA = 'RICK'
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
   sentencia 1
   sentencia 2
SET CURRENT SCHEMA = 'TOM'
   sentencia 3
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
   sentencia 4
```

Ejemplo 2: Una aplicación con las sentencias siguientes se ha enlazado con las opciones OPTPROFILE("Foo") y QUALIFIER("John"). El perfil de optimización KAAREL.BAR se utiliza para la sentencia 1 y el perfil de optimización "John"."Foo" se utiliza para la sentencia 2.

```
SET CURRENT SCHEMA = 'KAAREL'
SET CURRENT OPTIMIZATION PROFILE = 'BAR'
   sentencia 1
SET CURRENT SCHEMA = "Tom"
SET CURRENT OPTIMIZATION PROFILE NULL
   sentencia 2
```

Ejemplo 3: La serie vacía es un valor especial que indica que no se debe utilizar ningún perfil de optimización. El perfil de optimización "Hamid"."Foo" se utiliza para la sentencia 1 y no se utiliza ningún perfil de optimización para la sentencia 2.

```
SET CURRENT OPTIMIZATION PROFILE = '"Hamid"."Foo"'
   sentencia 1
SET CURRENT OPTIMIZATION PROFILE = ''
   sentencia 2
```

SET CURRENT PACKAGE PATH

La sentencia SET CURRENT PACKAGE PATH asigna un valor al registro especial CURRENT PACKAGE PATH. No está bajo el control de la transacción.

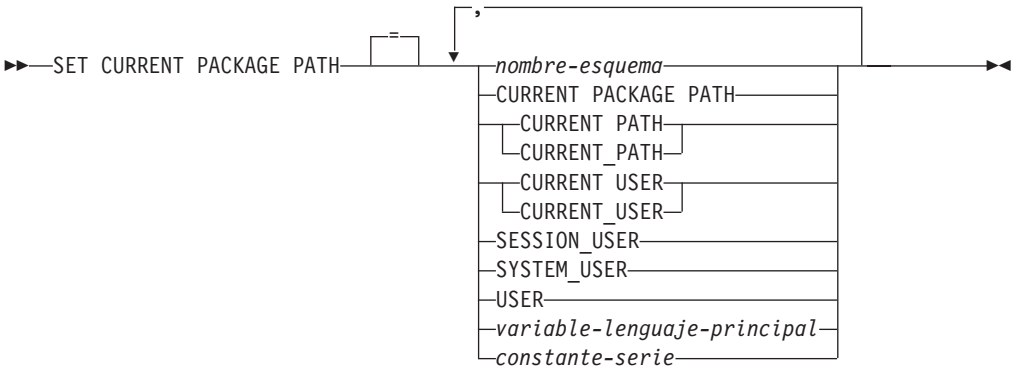
Invocación

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.

Sintaxis



Descripción

nombre-esquema
Identifica un esquema. El nombre no debe ser un identificador delimitado que esté vacío o que sólo contenga espacios en blanco (SQLSTATE 42815).

CURRENT PACKAGE PATH
El valor del registro especial CURRENT PACKAGE PATH antes de que se ejecute esta sentencia.

CURRENT PATH
El valor del registro especial CURRENT PATH.

CURRENT USER
El valor del registro especial CURRENT USER.

SESSION_USER
El valor del registro especial SESSION_USER.

SYSTEM_USER
El valor del registro especial SYSTEM_USER.

USER
El valor del registro especial USER.

variable-lenguaje-principal
Contiene uno o varios nombres de esquema, separados mediante comas. La variable del lenguaje principal debe:

- Ser una variable de serie de caracteres (CHAR o VARCHAR). La longitud real del contenido de la variable del lenguaje principal no debe exceder de la longitud del registro especial CURRENT PACKAGE PATH.
- No ser un valor nulo. Si se proporciona una variable 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 a la derecha con espacios en blanco si la longitud real de la variable de lenguaje principal es mayor que el contenido.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío o que contenga sólo espacios en blanco.

constante-serie
Especifica una constante de serie de caracteres que contiene cero, uno o varios nombres de esquema que están separados por comas. La constante de serie debe:

- Tener una longitud que no exceda de la longitud máxima del registro especial CURRENT PACKAGE PATH.
- No contener CURRENT PACKAGE PATH, CURRENT PATH, CURRENT_PATH, CURRENT USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, PATH ni USER.
- No contener un identificador delimitado que esté vacío o que contenga sólo espacios en blanco.

Normas

- Si el mismo esquema aparece más de una vez en la lista, se utiliza la primera aparición 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: Establecer el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: MYPKGS, 'ABC E', SYSIBM

```
SET CURRENT PACKAGE PATH = MYPKGS, 'ABC E', SYSIBM
```

La sentencia siguiente establece una variable del lenguaje principal en el valor de la lista resultante:

```
SET :hvpklist = CURRENT PACKAGE PATH
```

El valor de la variable del lenguaje principal es: "MYPKGS", "ABC E", "SYSIBM".

Ejemplo 2: Establecer el registro especial CURRENT PACKAGE PATH en la siguiente lista de esquemas: "SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH :hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH4","SCH5".

Ejemplo 3: Establecer el registro especial CURRENT PACKAGE PATH en la lista siguiente de esquemas: "SCH1","SCH#2","SCH3","SCH4","SCH5", donde :hvar1 contiene 'SCH4,SCH5'.

```
SET CURRENT PACKAGE PATH = SCH1,'SCH#2',"SCH3",:hvar1
```

El valor del registro especial CURRENT PACKAGE PATH después de que se ejecute esta sentencia es: "SCH1","SCH#2","SCH3","SCH4","SCH5".

Ejemplo 4: Borrar el registro especial CURRENT PACKAGE PATH.

```
SET CURRENT PACKAGE PATH = ''
```

Ejemplo 5: Añadir temporalmente el esquema "SCH_PROD" (contenido en la variable del lenguaje principal :prodschema) y el esquema "SCH_PROD2" (contenido en la variable del lenguaje principal :prod2schema) al final del registro especial CURRENT PACKAGE PATH para la ejecución del procedimiento SUMMARIZE. Después, volver a cambiar el registro especial CURRENT PACKAGE PATH a este valor anterior.

```
SET :oldCPP = CURRENT PACKAGE PATH

SET CURRENT PACKAGE PATH = CURRENT PACKAGE PATH,:prodschema,:prod2schema

CALL SUMMARIZE(:V1,:V2)

SET CURRENT PACKAGE PATH = :oldCPP
```

Ejemplo 6: Establecer el registro especial CURRENT PACKAGE PATH en una lista de nombres de esquema delimitados: "MY.SCHEMA" (punto intercalado), "OLD SCHEMA" (blanco intercalado). Utilice una sola variable del lenguaje principal que contenga ambos identificadores delimitados:

```
hv = '"MY.SCHEMA", "OLD SCHEMA"'

SET CURRENT PACKAGE PATH = :hv
```

o utilice una sola constante de serie que contenga ambos identificadores delimitados:

```
SET CURRENT PACKAGE PATH = '"MY.SCHEMA", "OLD SCHEMA"'
```

o utilice una lista de esquemas delimitados:

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

SET CURRENT PACKAGESET

La sentencia SET CURRENT PACKAGESET establece el nombre de esquema (identificador de colección) que se utilizará para seleccionar el paquete que se debe utilizar para las sentencias de SQL posteriores. La sentencia no está bajo el control de la transacción.

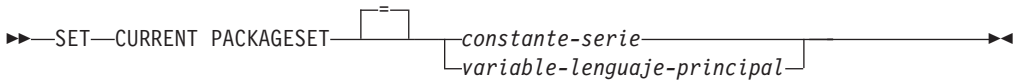
Invocación

Esta sentencia sólo se puede incluir en un programa de aplicación. Se trata de una sentencia ejecutable que no puede prepararse de forma dinámica. Esta sentencia no está soportada en REXX.

Autorización

No se necesita.

Sintaxis



Descripción

constante-serie
Constante de serie de caracteres. Si el valor excede de 128 bytes, sólo se utilizarán los primeros 128 bytes.

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. No puede establecerse en nulo. Si el valor excede de 128 bytes, sólo se utilizarán los primeros 128 bytes.

Notas

- Esta sentencia permite que una aplicación especifique el nombre de esquema utilizado al seleccionar un paquete para una sentencia de SQL ejecutable. La sentencia se procesa en el cliente y no fluye al servidor de aplicaciones.
- Se puede utilizar la opción de enlace lógico COLLECTION para crear un paquete con un nombre de esquema especificado.
- A diferencia de DB2 para z/OS, la sentencia SET CURRENT PACKAGESET se implementa sin soporte para un registro especial denominado CURRENT PACKAGESET.

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.

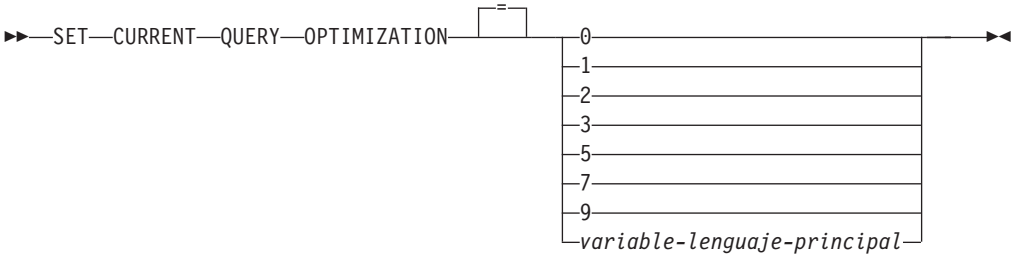
Invocación

Esta sentencia se puede incorporar a un programa de aplicación o emitir mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

clase-optimización

La *clase-optimización* se puede especificar como una constante de enteros o como el nombre de una variable del lenguaje principal que contendrá el valor adecuado en tiempo de ejecución. A continuación se facilita una visión general de las clases.

0 Especifica que se efectúa una optimización mínima para generar un plan de acceso. Esta clase es la más adecuada para el acceso SQL dinámico simple para tablas bien indexadas.

1 Especifica que se efectúa una optimización más o menos comparable a DB2 Versión 1 para generar un plan de acceso.

2 Especifica un nivel de optimización superior al de DB2 Versión 1, pero con un coste de optimización significativamente menor que los niveles 3 y superiores, especialmente para consultas muy complejas.

3 Especifica que se efectúa una optimización moderada para generar un plan de acceso.

5 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Para consultas SQL dinámico complejas, se utilizan las normas heurísticas para limitar el tiempo empleado en seleccionar un plan de acceso. Cuando sea posible, las consultas utilizarán tablas de consultas materializadas en lugar de las tablas base subyacentes.

7 Especifica que se efectúa una optimización significativa para generar un plan de acceso. Similar al 5 pero sin las normas heurísticas.

9 Especifica que se efectúa una optimización máxima para generar un plan de acceso. Puede expandir mucho el número de planes de acceso posibles que se evalúan. Esta clase debe utilizarse para determinar si se puede generar un plan de acceso mejor para las consultas muy complejas y de muy larga ejecución que utilizan tablas grandes. Las medidas de explicación y de rendimiento se pueden utilizar para verificar que se haya generado el plan mejor.

variable-lenguaje-principal
El tipo de datos es INTEGER. El valor debe estar en el rango de 0 a 9 (SQLSTATE 42815) pero debe ser 0, 1, 2, 3, 5, 7 o 9. Si una *variable-lenguaje-principal* tiene asociada una variable 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.

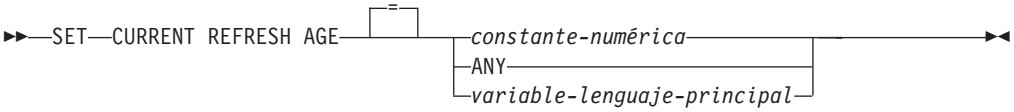
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

Sintaxis



Descripción

constante-numérica
Un valor DECIMAL(20,6) que representa una duración de indicación de fecha y hora. El valor debe ser 0 o 99.999.999.999.999 (la parte de microsegundos del valor se ignora y, por consiguiente, puede ser cualquier valor).

ANY
Es una notación abreviada de 99.999.999.999.999.

variable-lenguaje-principal
Una variable de tipo DECIMAL(20,6) u otro tipo que se pueda asignar a DECIMAL(20,6). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815). El valor de *variable-lenguaje-principal* debe ser 0 o 99.999.999.999.999.

Notas

- El valor inicial del registro especial CURRENT REFRESH AGE es cero.
- El valor de CURRENT REFRESH AGE se sustituye por el valor especificado. El valor debe ser 0 o 99.999.999.999.999. El valor 99.999.999.999.999 representa 9999 años, 99 meses, 99 días, 99 horas, 99 minutos y 99 segundos.

  Si el valor de CURRENT REFRESH AGE es 0, las tablas de 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 en un valor mayor que 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 de 99.999.999.999.999 no se puede utilizar en operaciones aritméticas de indicación de fecha y hora, porque el resultado estará fuera del rango válido para las fechas (SQLSTATE 22008).

Ejemplos

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT REFRESH AGE.

```
SET CURRENT REFRESH AGE ANY
```

Ejemplo 2: El siguiente ejemplo recupera el valor del registro especial CURRENT REFRESH AGE de una variable del lenguaje principal denominada CURMAXAGE. El valor, establecido en el ejemplo anterior, es 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.

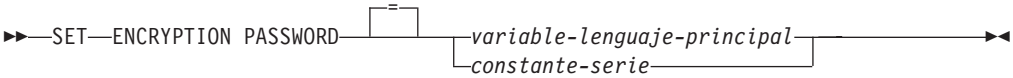
Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

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-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. La longitud de *variable-lenguaje-principal* debe tener una longitud de 6 a 127 bytes (SQLSTATE 428FC). No puede establecerse en nulo. Todos los caracteres se especifican en las mayúsculas y minúsculas que se desean porque no se realiza ninguna conversión a caracteres en mayúsculas.

constante-serie
Constante de serie de caracteres. La longitud debe ser de 6 a 127 bytes (SQLSTATE 428FC).

Notas

- El valor inicial de ENCRYPTION PASSWORD es la serie vacía (`''`).
- La *variable-lenguaje-principal* o *constante-serie* se transmite al servidor de base de datos siguiendo mecanismos de DB2 normales.

Ejemplos

Ejemplo 1: La sentencia siguiente establece la contraseña de cifrado (ENCRYPTION PASSWORD).

```
SET ENCRYPTION PASSWORD = 'Gre89Ea'
```

SET EVENT MONITOR STATE

La sentencia SET EVENT MONITOR STATE activa o desactiva el supervisor de sucesos. El estado actual de un supervisor de sucesos (activo o no activo) se determina utilizando la función incorporada EVENT_MON_STATE. La sentencia SET EVENT MONITOR STATE no está bajo el control de la transacción.

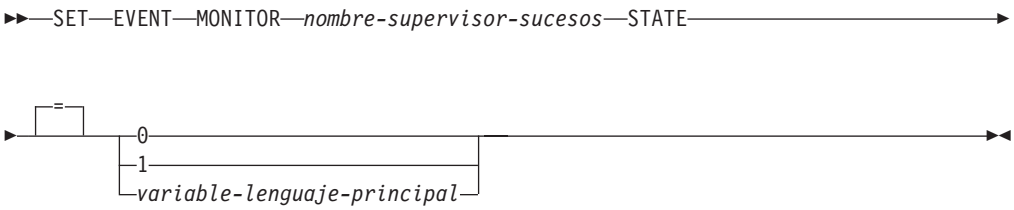
Invocación

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

Autorización

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

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-lenguaje-principal
El tipo de datos es INTEGER. El valor especificado debe ser 0 ó 1 (SQLSTATE 42815). Si una *variable-lenguaje-principal* tiene asociada una variable 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.
- Si la partición de base de datos en la que se debe ejecutar el supervisor de sucesos no está activa, la activación de éste se produce la próxima vez que se activa la partición de base de datos.
- Una vez activado un supervisor de sucesos, éste se comporta como un supervisor de sucesos de inicio automático hasta que el supervisor de sucesos se desactiva explícitamente o se recicla la instancia. Es decir, si un supervisor de sucesos está activo cuando se desactiva una partición de base de datos y ésta se vuelve a activar posteriormente, el supervisor de sucesos también se vuelve a activar explícitamente.
- Si un supervisor de sucesos de actividades está activo cuando se desactiva la base de datos, se descartan los registros de actividades anotados anteriormente de la cola. Para asegurarse de obtener todos los registros del supervisor de sucesos de actividades y que no se descarte ninguno de ellos, desactive explícitamente el supervisor de sucesos de actividades antes de desactivar la base de datos. Cuando se desactiva explícitamente un supervisor de sucesos de actividades, se procesan todos los registros de actividades anotados anteriormente en la cola antes de que se desactive el supervisor de sucesos.

Ejemplos

Ejemplo 1: Active un supervisor de sucesos denominado SMITHPAY.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

Ejemplo 2: Suponga que la MYSAMPLE es una base de datos de varias particiones con dos particiones de base de datos, 0 y 2. La partición 2 todavía no está activa.

```
En la partición de base de datos 0:

CONNECT TO MYSAMPLE;
CREATE EVENT MONITOR MYEVMON ON DBPARTITIONNUM 2;
SET EVENT MONITOR MYEVMON STATE 1;
```

MYEVMON se activa automáticamente siempre que se activa MYSAMPLE en la partición de base de datos 2. Esto se producirá hasta que se emita **SET EVENT MONITOR** MYEVMON **STATE** 0 o se detenga la partición 2.

SET INTEGRITY

La sentencia SET INTEGRITY se utiliza para:

- Sacar una o varias tablas del estado pendiente de establecer integridad (antes denominado "estado pendiente de comprobación") realizando el proceso de integridad necesario en esas tablas.
- Sacar una o varias tablas del estado pendiente de establecer integridad sin efectuar el proceso de integridad necesario en esas tablas.
- Colocar una o varias tablas en estado pendiente de establecer integridad.
- Colocar una o varias tablas en estado de acceso completo.
- Podar el contenido de una o más tablas de etapas.

Cuando la sentencia se utiliza para efectuar el proceso de integridad de una tabla tras haberse cargado o enlazado ésta, el sistema puede realizar un proceso incremental en la tabla comprobando sólo la parte añadida para verificar si existen violaciones de restricciones. Si la tabla sujeto es una tabla de consultas materializadas o una tabla de etapas y se realizan operaciones de carga, enlace o desenlace en sus tablas subyacentes, el sistema puede realizar una renovación incremental en la tabla de consultas materializadas o realizar una propagación incremental en la tabla de etapas utilizando sólo las partes delta de sus tablas subyacentes. Sin embargo, existen algunas situaciones en las que el sistema no podrá realizar tales optimizaciones y, en lugar de ello, realizará el proceso de integridad completo para garantizar la integridad de los datos. El proceso de integridad completo se realiza comprobando toda la tabla para verificar si existen violaciones de restricciones, volviendo a calcular la definición de una tabla de consultas materializadas o marcando una tabla de etapas como incoherente. Esta última acción implica la necesidad de realizar una renovación completa de su tabla de consultas materializadas asociada. Existe también una situación en que puede que desee solicitar explícitamente un proceso incremental especificando la opción INCREMENTAL.

La sentencia SET INTEGRITY está bajo control de transacciones.

Invocación

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

Autorización

Los privilegios necesarios para ejecutar la sentencia SET INTEGRITY dependen de la finalidad, tal como se describe a continuación.

- Sacar tablas del estado pendiente de establecer integridad y efectuar el proceso de integridad necesario.

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

  – Privilegio CONTROL para:

    - Las tablas en las que se realiza el proceso de integridad y, si se proporcionan tablas de excepciones para una o más de dichas tablas, privilegio INSERT para las tablas de excepciones

- Todas las tablas de claves foráneas descendientes, tablas de consultas materializadas inmediatas descendientes y tablas de etapas inmediatas descendientes que la sentencia colocará implícitamente en estado pendiente de establecer integridad

– Autorización LOAD (con condiciones). Para que la autorización LOAD pueda proporcionar privilegios válidos, deben satisfacerse todas las condiciones siguientes:
  - El proceso de integridad necesario no implica las acciones siguientes:
    • Renovar una tabla de consultas materializadas
    • Propagar hacia una tabla de etapas
    • Actualizar una columna generada o de identidad
  - Si se proporcionan tablas de excepciones para una o más tablas, se otorga el acceso necesario para la duración del proceso de integridad a las tablas en las que se realiza el proceso de integridad y a las tablas de excepciones asociadas. Es decir:
    • Privilegio SELECT y DELETE para cada tabla en la que se realiza el proceso de integridad y
    • Privilegio INSERT para las tablas de excepciones
– Autorización SYSADM o DBADM

• Sacar tablas del estado pendiente de establecer integridad sin efectuar el proceso de integridad necesario.

  El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  – Privilegio CONTROL para las tablas que se procesan; privilegio CONTROL para cada tabla de claves foráneas descendiente, tabla de consultas materializadas inmediata descendiente y tabla de etapas inmediata descendiente que la sentencia colocará implícitamente en estado pendiente de establecer integridad
  – Autorización LOAD
  – Autorización SYSADM o DBADM

• Colocar tablas en estado pendiente de establecer integridad.

  El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  – Privilegio CONTROL para:
    - Las tablas especificadas y
    - las tablas de claves foráneas descendientes que la sentencia colocará en estado pendiente de establecer integridad, y
    - las tablas de consultas materializadas inmediatas descendientes que la sentencia colocará en estado pendiente de establecer integridad, y
    - las tablas de etapas inmediatas descendientes que la sentencia colocará en estado pendiente de establecer integridad
  – Autorización LOAD
  – Autorización SYSADM o DBADM

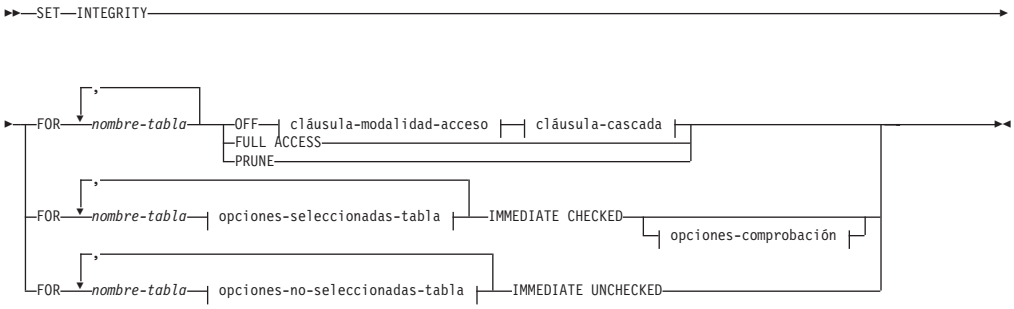
• Colocar una tabla en estado de acceso completo

  El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  – Privilegio CONTROL para las tablas que se colocan en estado de acceso completo
  – Autorización LOAD

  - Autorización SYSADM o DBADM
- Podar una tabla de etapas.

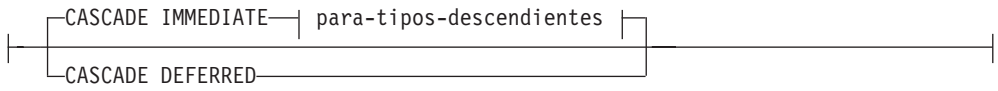
  El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:
  - Privilegio CONTROL sobre la tabla que está podándose
  - Autorización SYSADM o DBADM

Sintaxis

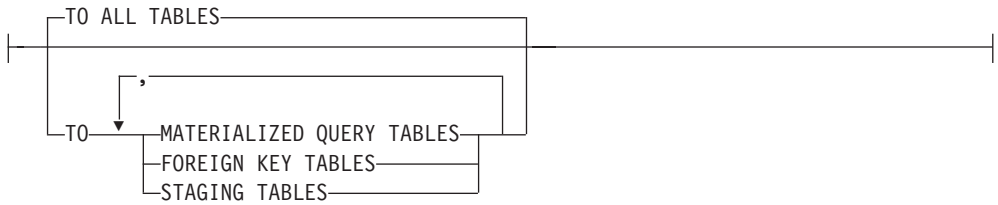


cláusula-modalidad-acceso:

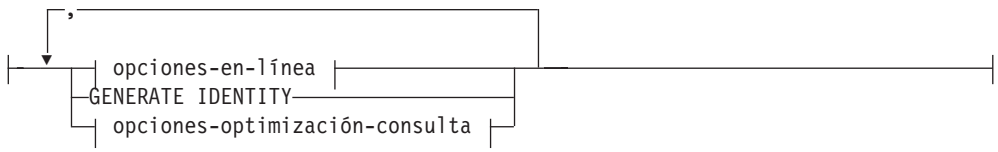
cláusula-cascada:



para-tipos-descendientes:



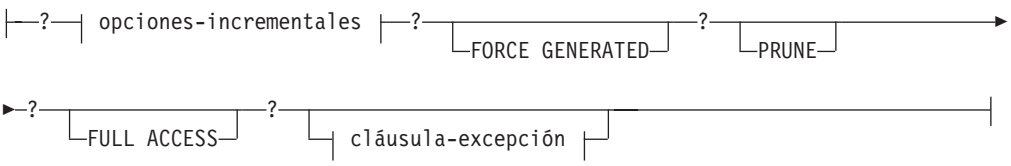
opciones-seleccionadas-tabla:



opciones-en-línea:

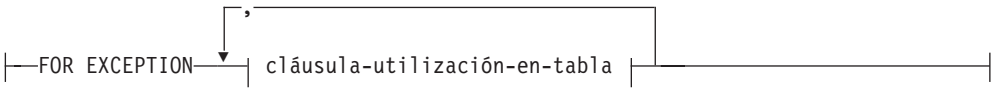
opciones-optimización-consulta:

opciones-comprobación:



opciones-incrementales:

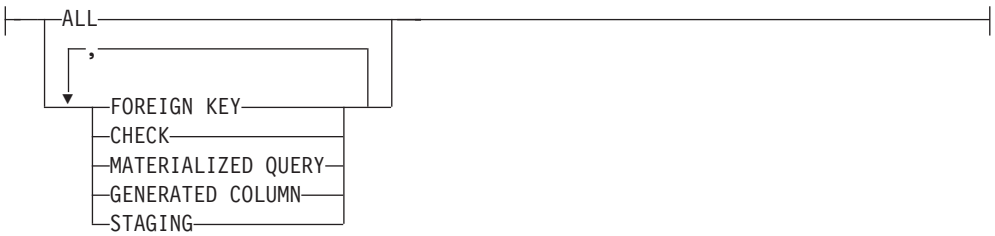
cláusula-excepción:



cláusula-utilización-en-tabla:

opciones-no-seleccionadas-tabla:

opciones-integridad:



Descripción

FOR *nombre-tabla*
Identifica una o más tablas para el proceso de la integridad. Debe ser una tabla descrita en el catálogo y no debe ser una vista, una tabla del catálogo ni una tabla con tipo.

OFF
Especifica que las tablas se colocan en estado pendiente de establecer integridad. En una tabla que se encuentra en estado pendiente de establecer integridad sólo está permitida una actividad muy limitada.

cláusula-modalidad-acceso
Especifica la posibilidad de lectura de la tabla mientras se encuentra en estado pendiente de establecer integridad.

NO ACCESS
Especifica que la tabla debe colocarse en estado sin acceso pendiente de establecer integridad, en que no está permitido el acceso de lectura o grabación a la tabla.

READ ACCESS
Especifica que la tabla debe colocarse en estado de acceso de lectura pendiente de establecer integridad, en que está permitido el acceso de lectura a la parte no añadida de la tabla. Esta opción no está permitida en una tabla en estado sin acceso pendiente de establecer integridad (SQLSTATE 428FH).

cláusula-cascada
Especifica si el estado pendiente de establecer integridad de la tabla a la que se hace referencia en la sentencia SET INTEGRITY debe aplicarse en cascada inmediatamente a las tablas descendientes.

CASCADE IMMEDIATE
Especifica que el estado pendiente de establecer integridad debe extenderse inmediatamente a las tablas descendientes.

para-tipos-descendientes
Especifica el tipo de tablas descendientes a las que se aplica en cascada de inmediato el estado pendiente de establecer integridad.

TO ALL TABLES
Especifica que el estado pendiente de establecer integridad debe aplicarse inmediatamente en cascada a todas las tablas descendientes de las tablas de la lista de invocación. Las tablas descendientes incluyen todas las tablas de claves foráneas descendientes, las tablas de etapas inmediatas y las tablas de consultas materializadas inmediatas que son descendientes de las tablas de la lista de invocación o que son descendientes de las tablas de claves foráneas descendientes.

Especificar TO ALL TABLES equivale a especificar TO FOREIGN KEY TABLES, TO MATERIALIZED QUERY TABLES y TO STAGING TABLES, todas ellas en la misma sentencia.

TO MATERIALIZED QUERY TABLES
Si sólo se especifica TO MATERIALIZED QUERY TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente sólo a las tablas de consultas materializadas inmediatas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad. Si se especifica TO FOREIGN KEY TABLES y también TO MATERIALIZED QUERY TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendientes, a todas las tablas de consultas materializadas inmediatas descendientes de las tablas de la lista de invocación y a todas las tablas de consultas materializadas inmediatas que son descendientes de las tablas de claves foráneas descendientes.

TO FOREIGN KEY TABLES
Especifica que el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a las tablas de claves foráneas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad.

TO STAGING TABLES
Especifica que el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a las tablas de etapas descendientes. Posteriormente, las demás tablas descendientes pueden colocarse en estado pendiente de establecer integridad, si es necesario, al sacar la tabla del estado pendiente de establecer integridad. Si se especifica TO FOREIGN KEY TABLES y también TO STAGING TABLES, el estado pendiente de establecer integridad se aplicará en cascada inmediatamente a todas las tablas de claves foráneas descendientes, a todas las tablas de etapas inmediatas descendientes de las tablas de la lista de invocación y a todas las tablas de etapas inmediatas que son descendientes de las tablas de claves foráneas descendientes.

CASCADE DEFERRED
Especifica que sólo las tablas de la lista de invocación deben colocarse en estado pendiente de establecer integridad. Los estados de las tablas descendientes no se cambiarán. Posteriormente, las tablas de claves foráneas descendientes pueden colocarse implícitamente en estado pendiente de establecer integridad cuando se comprueben sus tablas padre para verificar si existen violaciones de las restricciones. Las tablas de consultas materializadas inmediatas descendientes y las tablas de etapas inmediatas descendientes pueden colocarse implícitamente en estado pendiente de establecer integridad al comprobarse una de sus tablas subyacentes para verificar si existen violaciones de integridad. Una consulta de una tabla que tenga el estado Pendiente de establecimiento de integridad podría ser correcta si, en lugar de acceder a la tabla especificada, la consulta accede a una tabla de consulta materializada elegible que no tiene el estado Pendiente de establecimiento de integridad.

Si no se especifica la *cláusula-cascada*, el estado pendiente de establecer integridad se aplica en cascada inmediatamente a todas las tablas descendientes.

IMMEDIATE CHECKED
Especifica que la tabla debe sacarse del estado pendiente de establecer integridad realizando el proceso de integridad necesario en la tabla. Esto se realiza de acuerdo con la información que se ha establecido en las columnas STATUS y CONST_CHECKED de la vista de catálogo SYSCAT.TABLES. Es decir:

- El valor de la columna STATUS debe ser 'C' (la tabla se encuentra en estado pendiente de establecer integridad) o se devuelve un error (SQLSTATE 51027), a menos que la tabla sea una tabla de claves foráneas descendiente, una tabla de consultas materializadas descendiente o una tabla de etapas descendiente de una tabla especificada en la lista, que se encuentre en estado pendiente de establecer integridad y cuyos ancestros inmediatos también estén en la lista.
- Si la tabla que está comprobándose está en estado pendiente de establecer integridad, el valor de CONST_CHECKED indica qué opciones de integridad deben comprobarse.

Cuando la tabla se saca del estado pendiente de establecer integridad, sus tablas descendientes se colocan, si es necesario, en estado pendiente de establecer integridad. Se devuelve un aviso para indicar que las tablas descendientes se han colocado en estado pendiente de establecer integridad (SQLSTATE 01586).

Si la tabla es una tabla de consultas materializadas mantenida por el sistema, los datos se comprueban con la consulta y se renuevan en función de las necesidades. (No se puede utilizar IMMEDIATE CHECKED para las tablas de consultas materializadas mantenidas por el usuario.) Si la tabla es una tabla de etapas, los datos se comprueban con su definición de consulta y se propagan en función de las necesidades.

Cuando se comprueba la integridad de una tabla hija:

- Ninguno de sus padres puede estar en estado pendiente de establecer integridad, o
- cada uno de sus padres debe comprobarse para verificar si existen violaciones de restricciones en la misma sentencia SET INTEGRITY.

Cuando se renueva una tabla de consultas materializadas inmediata o cuando se propagan deltas a una tabla de etapas:

- Ninguna de sus tablas subyacentes puede estar en estado pendiente de establecer integridad, o
- cada una de sus tablas subyacentes debe comprobarse en la misma sentencia SET INTEGRITY.

De lo contrario, se devolverá un error (SQLSTATE 428A8).

opciones-seleccionadas-tabla

opciones-en-línea
Especifica la accesibilidad de la tabla mientras se está procesando.

ALLOW NO ACCESS
Especifica que ningún otro usuario puede acceder a la tabla mientras se está procesando, excepto si utiliza el nivel de aislamiento de lectura no confirmada.

ALLOW READ ACCESS
Especifica que los demás usuarios tienen acceso de sólo lectura a la tabla mientras se procesa.

ALLOW WRITE ACCESS
Especifica que los demás usuarios tienen acceso de lectura y grabación a la tabla mientras se procesa.

GENERATE IDENTITY
Especifica que, si la tabla contiene una columna de identidad, la sentencia SET INTEGRITY genera los valores. Por omisión, cuando se especifica la opción GENERATE IDENTITY, la sentencia SET INTEGRITY únicamente genera los valores de la columna de identidad de las filas enlazadas. Debe especificarse la opción NOT INCREMENTAL con la opción GENERATE IDENTITY para que la sentencia SET INTEGRITY genere los valores de la columna de identidad de todas las filas de la tabla, entre ellas las filas enlazadas, las filas cargadas y las filas existentes. Si no se especifica la opción GENERATE IDENTITY, los actuales valores de la columna de identidad de todas las filas de la tabla permanecen invariables.

opciones-optimización-consulta
Especifica las opciones de optimización de consulta para el mantenimiento de las tablas de consultas materializadas REFRESH DEFERRED.

ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY
Especifica que, si el registro especial CURRENT REFRESH AGE se establece en 'ANY', el mantenimiento del *nombre-tabla* permitirá el uso de tablas de consultas materializadas REFRESH DEFERRED para optimizar la consulta que mantiene el *nombre-tabla*. Si el *nombre-tabla* no es una tabla de consultas materializadas REFRESH DEFERRED, se devuelve un error (SQLSTATE 428FH). Las tablas de consultas materializadas REFRESH IMMEDIATE siempre se tienen en cuenta en la optimización de consulta.

opciones-comprobación

opciones-incrementales

INCREMENTAL
Especifica la aplicación del proceso de integridad a la parte añadida (si existe) de la tabla. Si no puede satisfacerse una petición de este tipo (es decir, si el sistema detecta que debe comprobarse toda la tabla para verificar la integridad de los datos), se devuelve un error (SQLSTATE 55019).

NOT INCREMENTAL
Especifica la aplicación del proceso de integridad a toda la tabla. Si la tabla es una tabla de 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 causa el proceso completo de las tablas de claves foráneas descendientes y las tablas de consultas materializadas inmediatas descendientes. Si la tabla es una tabla de etapas, se establece en un estado incoherente.

Si no se especifica la cláusula *opciones-incrementales*, el sistema determina si es posible el proceso incremental; si no es posible, se comprueba toda la tabla.

FORCE GENERATED
Si la tabla incluye columnas generadas por expresión, los valores se

calculan basándose en la expresión y se almacenan en la columna. Si no se especifica esta opción, los valores actuales se comparan con el valor calculado de la expresión, como si hubiera en vigor una restricción de comprobación de igualdad. Si se realiza el proceso de integridad de la tabla de manera incremental, las columnas generadas sólo se calculan para la parte añadida.

PRUNE
Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH). Si también se especifica la opción de comprobación INCREMENTAL, se devuelve un error (SQLSTATE 428FH).

FULL ACCESS
Especifica que la tabla debe pasar a ser una tabla de acceso completo tras ejecutarse la sentencia SET INTEGRITY.

Cuando se realiza el proceso incremental de una tabla subyacente (que tiene tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes) de la lista de invocación, la tabla subyacente se establece, en función de las necesidades, en la modalidad sin movimiento de datos tras ejecutarse la sentencia SET INTEGRITY. Cuando se sacan del estado pendiente de establecer integridad todas las tablas de etapas y tablas de consultas materializadas inmediatas dependientes que pueden renovarse de manera incremental, la tabla subyacente pasa automáticamente del estado sin movimiento de datos al estado de acceso completo. Si se especifica la opción FULL ACCESS con la opción IMMEDIATE CHECKED, la tabla subyacente se establece directamente en estado de acceso completo (y elude el estado sin movimiento de datos). Las tablas de consultas materializadas inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH TABLE posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden marcarse como incoherentes.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consultas materializadas inmediatas dependientes ni tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en estado de acceso completo tras ejecutarse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

cláusula-excepción

FOR EXCEPTION
Especifica que cualquier fila en que se produzca la violación de una restricción que se comprueba debe moverse a una tabla de excepciones. Aunque se detecten errores, la tabla se saca del estado pendiente de establecer integridad. Se devuelve un aviso para indicar que se ha movido una o varias filas a las tablas de excepciones (SQLSTATE 01603).

Si no se especifica la opción FOR EXCEPTION y se produce una violación de alguna restricción, sólo se devuelve la primera violación que se ha detectado (SQLSTATE 23514). Si se produce

una violación en alguna de las tablas, todas las tablas quedan en estado pendiente de establecer integridad.

Se recomienda utilizar siempre la opción FOR EXCEPTION al comprobar la existencia de violaciones de restricciones para evitar la retrotracción de la sentencia SET INTEGRITY en caso de hallarse una violación.

IN *nombre-tabla*
Especifica la tabla de la que se deben mover las filas que violan restricciones. Debe haber una tabla de excepciones especificada para cada tabla que se comprueba. Esta cláusula no puede especificarse para una tabla de consultas materializadas o una tabla de etapas (SQLSTATE 428A7).

USE *nombre-tabla*
Especifica la tabla de excepciones a la que se deben mover las filas de errores.

FULL ACCESS
Si se especifica la opción FULL ACCESS como única operación de la sentencia, la tabla se coloca en estado de acceso completo sin volver a comprobar las violaciones de integridad. Sin embargo, las tablas de consultas materializadas inmediatas dependientes que no se han renovado pueden necesitar un nuevo cálculo completo en las sentencias REFRESH TABLE posteriores, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes delta de la tabla pueden establecerse en estado incompleto. Esta opción sólo puede especificarse para una tabla que se encuentra en estado sin movimiento de datos o sin acceso, pero no en estado pendiente de establecer integridad (SQLSTATE 428FH).

PRUNE
Esta opción sólo puede especificarse para las tablas de etapas. Especifica que el contenido de la tabla de etapas va a podarse y que la tabla de etapas va a establecerse en un estado no coherente. Si alguna tabla de la lista *nombre-tabla* no es una tabla de etapas, se devuelve un error (SQLSTATE 428FH).

opciones-no-seleccionadas-tabla

opciones-integridad
Permite definir los tipos de proceso de integridad necesarios que deben eludirse cuando se saca la tabla del estado pendiente de establecer integridad.

ALL
La tabla se sacará inmediatamente del estado pendiente de establecer integridad sin efectuar ningún proceso de integridad necesario.

FOREIGN KEY
No se realizará la comprobación de las restricciones de clave foránea necesaria al sacar la tabla del estado pendiente de establecer integridad.

CHECK
No se realizará la comprobación de las restricciones de comprobación necesaria al sacar la tabla del estado pendiente de establecer integridad.

MATERIALIZED QUERY
No se realizará la renovación necesaria de una tabla de consultas materializadas al sacar la tabla del estado pendiente de establecer integridad.

GENERATED COLUMN
No se realizará la comprobación de las restricciones de columna generada necesaria al sacar la tabla del estado pendiente de establecer integridad.

STAGING
No se realizará la propagación de datos necesaria a una tabla de etapas al sacar la tabla del estado pendiente de establecer integridad.

Si no es necesario ningún otro tipo de proceso de integridad en la tabla una vez marcado como eludido un tipo de proceso de integridad concreto, la tabla se saca de inmediato del estado pendiente de establecer integridad.

FULL ACCESS
Especifica que las tablas van a ser tablas de acceso completo tras ejecutarse la sentencia SET INTEGRITY.

Cuando se realiza el proceso incremental de una tabla subyacente de la lista de invocación y ésta tiene tablas de consultas materializadas inmediatas dependientes o tablas de etapas inmediatas dependientes, la tabla subyacente se establece, en función de las necesidades, en estado sin movimiento de datos tras ejecutarse la sentencia SET INTEGRITY. Una vez sacadas del estado pendiente de establecer integridad todas las tablas de etapas y tablas de consultas materializadas inmediatas dependientes que pueden renovarse de manera incremental, la tabla subyacente pasa automáticamente del estado sin movimiento de datos al estado de acceso completo. Si se especifica la opción FULL ACCESS con la opción IMMEDIATE UNCHECKED, la tabla subyacente se establece directamente en estado de acceso completo (elude el estado sin movimiento de datos). Las tablas de consultas materializadas inmediatas dependientes que no se han renovado pueden someterse a un nuevo cálculo completo en la sentencia REFRESH TABLE posterior, y las tablas de etapas inmediatas dependientes a las que no se han propagado las partes añadidas de la tabla pueden marcarse como incoherentes.

Cuando una tabla subyacente de la lista de invocación necesita un proceso completo o no tiene tablas de consultas materializadas inmediatas dependientes ni tablas de etapas inmediatas dependientes, la tabla subyacente se establece directamente en estado de acceso completo tras ejecutarse la sentencia SET INTEGRITY, con independencia de si se ha especificado o no la opción FULL ACCESS.

Si se ha especificado la opción FULL ACCESS con la opción IMMEDIATE UNCHECKED y la sentencia no saca la tabla del estado pendiente de establecer integridad, se devuelve un error (SQLSTATE 428FH).

IMMEDIATE UNCHECKED
Especifica uno de los siguientes:

- La tabla debe sacarse del estado pendiente de establecer integridad inmediatamente sin ningún proceso de integridad necesario.
- La tabla debe eludir uno o varios tipos de proceso de integridad necesarios al sacarse la tabla del estado pendiente de establecer integridad mediante una sentencia SET INTEGRITY posterior con la opción IMMEDIATE CHECKED.

Antes de utilizar esta opción, examine cómo afecta a la integridad de los datos. Consulte el apartado “Notas” siguiente.

Notas

- Efectos en las tablas que están en uno de los estados restringidos relacionados con establecer integridad:
  - La utilización de INSERT, UPDATE o DELETE no está permitida en una tabla que está en estado de acceso de lectura o sin acceso. Además, se rechazará cualquier sentencia que requiera una modificación de este tipo para una tabla que se encuentra en un estado de este tipo. Por ejemplo, no está permitido suprimir una fila de una tabla padre que se aplica en cascada a una tabla dependiente que está en estado sin acceso.
  - La utilización de SELECT no está permitida en una tabla que está en estado sin acceso. Además, se rechazará cualquier sentencia que requiera acceso de lectura a una tabla que está en estado sin acceso.
  - Normalmente las nuevas restricciones añadidas a una tabla se imponen inmediatamente. Sin embargo, si la tabla está en estado pendiente de establecer integridad, la comprobación de cualquier restricción nueva se diferirá hasta que se saque la tabla del estado pendiente de establecer integridad. Si la tabla está en estado pendiente de establecer integridad, la adición de una nueva restricción causará que la tabla se coloque en estado sin acceso pendiente de establecer integridad, pues la validez de los datos está en peligro.
  - La sentencia CREATE INDEX no puede hacer referencia a ninguna tabla que esté en estado de acceso de lectura o sin acceso. De forma similar, una sentencia ALTER TABLE para añadir una clave primaria o una restricción de unicidad no puede hacer referencia a ninguna tabla que esté en estado de acceso de lectura o sin acceso.
  - El programa de utilidad de importación no puede ejecutarse en una tabla que está en estado de acceso de lectura o sin acceso.
  - El programa de utilidad de exportación no puede ejecutarse en una tabla que está en estado sin acceso, pero sí en una tabla que está en estado de acceso de lectura. Si una tabla está en estado de acceso de lectura, el programa de utilidad de exportación sólo exportará los datos de la parte no añadida.
  - Las operaciones (como REORG, REDISTRIBUTE, la actualización de la clave de distribución, la actualización de la clave de clúster multidimensional, la actualización de la clave de clúster de rango, la actualización de la clave de particionamiento de tabla, etc.) que pueden implicar un movimiento de datos dentro de una tabla no están permitidas en una tabla que está en alguno de los estados siguientes: acceso de lectura, sin acceso o sin movimiento de datos.
  - Los programas de utilidad LOAD, BACKUP, RESTORE, UPDATE STATISTICS, RUNSTATS, REORGCHK, LIST HISTORY y ROLLFORWARD pueden utilizarse en una tabla que está en cualquiera de los estados siguientes: acceso completo, acceso de lectura, sin acceso o sin movimiento de datos.
  - Las sentencias ALTER TABLE, COMMENT, DROP TABLE, CREATE ALIAS, CREATE TRIGGER, CREATE VIEW, GRANT, REVOKE y SET INTEGRITY pueden hacer referencia a una tabla que está en cualquiera de los estados siguientes: acceso completo, acceso de lectura, sin acceso o sin movimiento de datos. Sin embargo, pueden hacer que la tabla se establezca en el estado sin acceso.

– Los paquetes, las vistas y cualquier otro objeto que dependa de una tabla que está en estado sin acceso devolverán un error al accederse a la tabla en tiempo de ejecución. Los paquetes que dependan de una tabla que está en estado de acceso de lectura devolverán un error cuando se intente realizar una operación de inserción, actualización o supresión en la tabla en tiempo de ejecución.

La eliminación de las filas con las violaciones mediante la sentencia SET INTEGRITY no es un suceso de supresión. Por lo tanto, una sentencia SET INTEGRITY nunca activa los activadores. Del mismo modo, la actualización de columnas generadas mediante la opción FORCE GENERATED no produce la activación de activadores.

- 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 cumplen las especificaciones de integridad que se han definido para la tabla y se utiliza la opción IMMEDIATE UNCHECKED, puede que se devuelvan resultados de consulta incorrectos.

    El hecho de haber sacado la tabla del estado pendiente de establecer integridad sin realizar el proceso de integridad necesario quedará registrado en el catálogo (el byte respectivo de la columna CONST_CHECKED de la vista SYSCAT.TABLES se establecerá en 'U'). Esto indica que el usuario ha asumido la tarea de asegurar la integridad de los datos con respecto a las restricciones específicas. Este valor no cambia hasta que:

    - La tabla vuelve a colocarse en estado pendiente de establecer integridad (haciéndose referencia a la tabla en una sentencia SET INTEGRITY con la opción OFF), momento en que los valores 'U' de la columna CONST_CHECKED se cambian por valores 'W', para indicar que el usuario había asumido anteriormente la responsabilidad de comprobar la integridad de los datos y el sistema debe verificar los datos.
    - Se eliminan todas las restricciones no comprobadas de la tabla.

    El estado 'W' se diferencia del estado 'N' en que registra el hecho de que el usuario comprobaba anteriormente la integridad, pero que el sistema todavía no lo hace. Si el usuario emite la sentencia SET INTEGRITY ... IMMEDIATE CHECKED con la opción NOT INCREMENTAL, el sistema vuelve a comprobar toda la tabla para verificar la integridad de los datos (o realiza una renovación completa en una tabla de 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 cada vez que la tabla subyacente se saca del estado pendiente de establecer integridad y cada vez 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 cada vez que se saca la tabla subyacente del estado pendiente de establecer integridad. Estos valores 'U' propagados de las columnas CONST_CHECKED de las tablas de 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 cuyo proceso de integridad necesario se ha eludido con la opción 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 materializadas 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 con la opción IMMEDIATE CHECKED y la tabla padre necesita que se realice una comprobación completa de sus restricciones, se comprobarán las restricciones de clave foránea de la tabla hija, con independencia de si la tabla hija tiene o no un valor 'U' en la columna CONST_CHECKED para las restricciones de clave foránea.

- Después de haber añadido datos utilizando LOAD INSERT o ALTER TABLE ATTACH, la sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED comprueba la tabla para verificar si existen violaciones de restricciones. El sistema determina si es posible o no el proceso incremental en la tabla. Si es posible, sólo se comprueba la parte añadida para las violaciones de integridad. Si no es posible, el sistema comprueba toda la tabla para verificar si existen violaciones de integridad.
- Veamos la sentencia siguiente:

```
SET INTEGRITY FOR T IMMEDIATE CHECKED
```

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 mientras se encuentra en estado pendiente de establecer integridad.
– Cuando ha tenido lugar una operación LOAD REPLACE para T, su tabla padre o sus tablas subyacentes.
– Cuando se ha activado la opción NOT LOGGED INITIALLY WITH EMPTY TABLE tras la última comprobación de integridad realizada en T, en sus tablas padre o en sus tablas subyacentes.

  - El efecto de aplicación en cascada del proceso completo, cuando se ha comprobado la integridad de T (o una tabla subyacente, si T es una tabla de consultas materializadas o una tabla de etapas) de forma no incremental.
  - Si el espacio de tablas que contiene la tabla o su tabla padre (o tabla subyacente de una tabla de 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 tablas.
  - 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`.
- Si se produce un error durante el proceso de integridad, se retrotraerán todos los efectos del proceso (incluidas la supresión realizada en la tabla original y la inserción en las tablas de excepciones).
- Si la emisión de una sentencia SET INTEGRITY con la opción FORCE GENERATED no se ejecuta satisfactoriamente porque el espacio de anotaciones cronológicas es insuficiente, incremente el espacio de anotaciones cronológicas activo disponible y vuelva a emitir la sentencia SET INTEGRITY. También puede utilizar la sentencia SET INTEGRITY con las opciones GENERATED COLUMN e IMMEDIATE UNCHECKED para eludir la comprobación de las columnas generadas para la tabla. A continuación, emita una sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED y sin la opción FORCE GENERATED para comprobar si en la tabla existen otras violaciones de la integridad (si corresponde) y sacar la tabla del estado pendiente de establecer integridad. Cuando se haya sacado la tabla del estado pendiente de establecer integridad, las columnas generadas podrán actualizarse para que tomen sus valores por omisión (generados) asignándolos a la palabra clave DEFAULT en una sentencia UPDATE. Esta acción se realiza utilizando varias sentencias de actualización buscada basadas en rangos (cada una de ellas seguida de una confirmación) o bien haciendo uso de un enfoque basado en cursores utilizando confirmaciones intermitentes. Deberá utilizarse un cursor "con retención" si van a retenerse bloqueos tras las confirmaciones intermitentes del enfoque basado en cursores.
- Cuando una tabla se ha colocado en estado pendiente de establecer integridad mediante la opción CASCADE DEFERRED de la sentencia SET INTEGRITY o el mandato LOAD, o mediante la sentencia ALTER TABLE con la cláusula ATTACH, y se ha comprobado para verificar si existen violaciones de integridad mediante la opción IMMEDIATE CHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendientes, tablas de consultas materializadas inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad, según proceda:
  - Si se comprueba toda la tabla para verificar si existen violaciones de integridad, sus tablas de claves foráneas descendientes, tablas de consultas materializadas inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad.
  - Si la tabla se comprueba para verificar si existen violaciones de integridad de manera incremental, sus tablas de consultas materializadas inmediatas descendientes y tablas de etapas se colocarán en estado pendiente de establecer integridad y sus tablas de claves foráneas descendientes seguirán en sus estados originales.

  - Si la tabla no necesita ninguna comprobación, sus tablas de consultas materializadas inmediatas descendientes, tablas de etapas descendientes y tablas de claves foráneas descendientes seguirán en sus estados originales.
- Cuando una tabla se ha colocado en estado pendiente de establecer integridad mediante la opción CASCADE DEFERRED (de la sentencia SET INTEGRITY o del mandato LOAD) y se ha sacado del estado pendiente de establecer integridad mediante la opción IMMEDIATE UNCHECKED de la sentencia SET INTEGRITY, sus tablas de claves foráneas descendientes, tablas de consultas materializadas inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad, según proceda:
  - Si la tabla se ha cargado utilizando la modalidad REPLACE, sus tablas de claves foráneas descendientes, tablas de consultas materializadas inmediatas descendientes y tablas de etapas inmediatas descendientes se colocarán en estado pendiente de establecer integridad.
  - Si la tabla se ha cargado utilizando la modalidad INSERT, sus tablas de consultas materializadas inmediatas descendientes y tablas de etapas se colocarán en estado pendiente de establecer integridad y sus tablas de claves foráneas descendientes seguirán en sus estados originales.
  - Si la tabla no se ha cargado, sus tablas de consultas materializadas inmediatas descendientes, tablas de etapas descendientes y tablas de claves foráneas descendientes seguirán en sus estados originales.
- SET INTEGRITY normalmente es una sentencia de larga ejecución. Por ello, para reducir el riesgo de retrotracción de toda la sentencia debido a un tiempo de espera excedido de bloqueo, puede emitir la sentencia SET CURRENT LOCK TIMEOUT con la opción WAIT antes de ejecutar la sentencia SET INTEGRITY y, a continuación, restaurar el valor anterior del registro especial una vez confirmada la transacción. No obstante, tenga presente que el registro especial CURRENT LOCK TIMEOUT sólo afecta a un conjunto de tipos de bloqueo concreto.
- Si utiliza la opción ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY, compruebe que el orden de mantenimiento es correcto para las tablas de consultas materializadas REFRESH DEFERRED. Por ejemplo, imagine dos tablas de consultas materializadas, MQT1 y MQT2, cuyas consultas materializadas comparten las mismas tablas subyacentes. La consulta materializada de MQT2 se puede calcular utilizando MQT1, en lugar de las tablas subyacentes. Si se emplean distintas sentencias para mantener estas dos tablas de consultas materializadas, y se mantiene MQT2 en primer lugar, el sistema puede elegir utilizar el contenido de MQT1, que todavía no se ha mantenido, para mantener MQT2. En ese caso, MQT1 contendría los datos actuales pero MQT2 podría contener todavía datos obsoletos, aunque el mantenimiento de ambas se realizara casi al mismo tiempo. El orden de mantenimiento correcto, si se utilizan dos sentencias SET INTEGRITY en lugar de una, consiste en mantener MQT1 en primer lugar.
- Al utilizar la sentencia SET INTEGRITY para realizar el proceso de integridad en una tabla base cargada o enlazada, se recomienda procesar sus tablas de consultas materializadas REFRESH IMMEDIATE dependientes y sus tablas de etapas PROPAGATE IMMEDIATE en la misma sentencia SET INTEGRITY para evitar colocar estas tablas dependientes en el estado sin acceso pendiente de establecer integridad al final del proceso de SET INTEGRITY. Tenga en cuenta que, en el caso de las tablas base con un gran número de tablas de consultas materializadas REFRESH IMMEDIATE dependientes y tablas de etapas PROPAGATE IMMEDIATE, puede ser imposible procesar todos los dependientes en la misma sentencia que la tabla base debido a restricciones de memoria.

- Si se especifica la opción FORCE GENERATED o GENERATE IDENTITY y la columna generada forma parte de un índice exclusivo, la sentencia SET INTEGRITY devuelve un error (SQLSTATE 23505) y se retrotrae si detecta claves duplicadas en el índice exclusivo. Este error se devuelve aunque haya una tabla de excepciones para la tabla que se procesa.

  Este escenario puede producirse cuando se dan las circunstancias siguientes:

  – Se ejecuta la sentencia SET INTEGRITY después de un mandato LOAD para la tabla, y en la operación de carga se ha especificado el modificador de tipo de archivo GENERATEDOVERRIDE o IDENTITYOVERRIDE. Para evitar que se produzca este escenario, se recomienda utilizar el modificador de tipo de archivo GENERATEDIGNORE o GENERATEDMISSING en lugar de GENERATEDOVERRIDE, y utilizar el modificador IDENTITYIGNORE o IDENTITYMISSING en lugar de IDENTITYOVERRIDE. Si se utilizan los modificadores recomendados, no será necesario procesar columnas generadas por expresión o columnas de identidad durante la ejecución de la sentencia SET INTEGRITY.

  – Se ejecuta la sentencia SET INTEGRITY después de que una sentencia ALTER TABLE modifique la expresión de una columna generada por expresión.

  Para sacar una tabla del estado pendiente de establecer integridad después de producirse un escenario de este tipo:

  – No utilice la opción FORCE GENERATED o GENERATE IDENTITY para volver a generar los valores de columna. Utilice, en cambio, la opción IMMEDIATE CHECKED con la opción FOR EXCEPTION para mover las filas en que se produce una violación de la expresión de columna generada a una tabla de excepciones. A continuación, vuelva a insertar las filas en la tabla a partir de la tabla de excepciones, con lo que se generará la expresión correcta y se realizará la comprobación de la clave exclusiva. Esta operación evita tener que procesar nuevamente toda la tabla, ya que únicamente deben procesarse de nuevo las filas en que se ha producido una violación de la expresión de columna generada.

  – Si la tabla que se procesa tiene particiones enlazadas, desenlace esas particiones realizando las acciones que se describen en el punto anterior. A continuación, vuelva a enlazar las particiones y ejecute una sentencia SET INTEGRITY para efectuar el proceso de integridad en las particiones enlazadas por separado.

- Si se especifica una tabla protegida para la sentencia SET INTEGRITY junto con una tabla de excepciones, deben satisfacerse todos los criterios de tabla siguientes; de lo contrario, se devuelve un error (SQLSTATE 428A5):

  – Las tablas deben estar protegidas con la misma política de seguridad.

  – Si una columna de la tabla protegida tiene el tipo de datos DB2SECURITYLABEL, la columna correspondiente de la tabla de excepciones también debe tener el tipo de datos DB2SECURITYLABEL.

  – Si una columna de la tabla protegida está protegida con una etiqueta de seguridad, la columna correspondiente de la tabla de excepciones también debe estar protegida con la misma etiqueta de seguridad.

- ***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

Ejemplos

Ejemplo 1: El siguiente es un ejemplo de una consulta que proporciona información acerca del estado pendiente de establecer integridad y los estados de restricción de acceso relacionados con establecer integridad de las tablas. SUBSTR se utiliza para extraer bytes individuales de la columna CONST_CHECKED de SYSCAT.TABLES. El primer byte representa las restricciones de clave foránea; el segundo byte representa las restricciones de comprobación; el quinto byte representa la integridad de tabla de consultas materializadas; el sexto byte representa las restricciones de columna generada; el séptimo byte representa la integridad de tabla de etapas, y el octavo byte representa las restricciones de particionamiento de datos. STATUS proporciona el estado pendiente de establecer integridad y ACCESS_MODE proporciona el estado de restricción de acceso relacionado con establecer integridad.

```
SELECT TABNAME, STATUS, ACCESS_MODE,
  SUBSTR(CONST_CHECKED,1,1) AS FK_CHECKED,
  SUBSTR(CONST_CHECKED,2,1) AS CC_CHECKED,
  SUBSTR(CONST_CHECKED,5,1) AS MQT_CHECKED,
  SUBSTR(CONST_CHECKED,6,1) AS GC_CHECKED,
  SUBSTR(CONST_CHECKED,7,1) AS STG_CHECKED,
  SUBSTR(CONST_CHECKED,8,1) AS DP_CHECKED
FROM SYSCAT.TABLES
```

Ejemplo 2: Coloque la tabla PARENT en estado sin acceso pendiente de establecer integridad y aplique en cascada de inmediato el estado pendiente de establecer integridad a sus descendientes.

```
SET INTEGRITY FOR PARENT OFF
  NO ACCESS CASCADE IMMEDIATE
```

Ejemplo 3: Coloque la tabla PARENT en estado de acceso de lectura pendiente de establecer integridad sin aplicar en cascada de inmediato el estado pendiente de establecer integridad a sus descendientes.

```
SET INTEGRITY FOR PARENT OFF
  READ ACCESS CASCADE DEFERRED
```

Ejemplo 4: Compruebe la integridad de una tabla denominada FACT_TABLE. Si no se detecta ninguna violación de integridad, la tabla se saca del estado pendiente de establecer integridad. Si se detecta alguna violación de integridad, toda la sentencia se retrotrae y la tabla permanece en estado pendiente de establecer integridad.

```
SET INTEGRITY FOR FACT_TABLE IMMEDIATE CHECKED
```

Ejemplo 5: Compruebe la integridad de las tablas SALES y PRODUCTS y mueva las filas en que se produce una violación de integridad a las tablas de excepciones denominadas SALES_EXCEPTIONS y PRODUCTS_EXCEPTIONS. Las tablas SALES y PRODUCTS se quitan del estado de pendiente de establecimiento de integridad, tanto si existen violaciones de integridad como si no existen.

```
SET INTEGRITY FOR SALES, PRODUCTS IMMEDIATE CHECKED
  FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS,
  IN PRODUCTS USE PRODUCTS_EXCEPTIONS
```

Ejemplo 6: Habilite la comprobación de restricciones FOREIGN KEY en la tabla MANAGER y la comprobación de restricciones CHECK en la tabla EMPLOYEE de modo que se eludan con la opción IMMEDIATE UNCHECKED.

```
SET INTEGRITY FOR MANAGER FOREIGN KEY,
  EMPLOYEE CHECK IMMEDIATE UNCHECKED
```

Ejemplo 7: Añada una restricción de comprobación y una clave foránea a la tabla EMP_ACT utilizando dos sentencias ALTER TABLE. La sentencia SET INTEGRITY con la opción OFF se utiliza para colocar la tabla en estado pendiente de establecer integridad, de modo que las restricciones no se comprueban de inmediato al ejecutarse las dos sentencias ALTER TABLE. La única sentencia SET INTEGRITY con la opción IMMEDIATE CHECKED se utiliza para comprobar las dos restricciones añadidas en un solo recorrido por la tabla.

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK
  (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY
  (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
  FOR EXCEPTION IN EMP_ACT USE EMP_ACT_EXCEPTIONS
```

Ejemplo 8: Actualice las columnas generadas con los valores correctos.

```
SET INTEGRITY FOR SALES IMMEDIATE CHECKED
  FORCE GENERATED
```

Ejemplo 9: Realice adiciones (utilizando LOAD INSERT) de distintas fuentes a una tabla subyacente (SALES) de una tabla de consultas materializadas REFRESH IMMEDIATE (SALES_SUMMARY). Compruebe la integridad de los datos de SALES de manera incremental y renueve SALES_SUMMARY de manera incremental. En este escenario, la comprobación de la integridad que se realiza para SALES y la renovación de SALES_SUMMARY son incrementales, pues el sistema elige el proceso incremental. La opción ALLOW READ ACCESS se utiliza en la tabla SALES para permitir las lecturas simultáneas de los datos existentes mientras se realiza la comprobación de integridad de la parte cargada de la tabla.

```
LOAD FROM 2000_DATA.DEL OF DEL
  INSERT INTO SALES ALLOW READ ACCESS;
LOAD FROM 2001_DATA.DEL OF DEL
  INSERT INTO SALES ALLOW READ ACCESS;
SET INTEGRITY FOR SALES ALLOW READ ACCESS IMMEDIATE CHECKED
  FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS;
REFRESH TABLE SALES_SUMMARY;
```

Ejemplo 10: Enlace una partición nueva a una tabla particionada de datos denominada SALES. Efectúe una comprobación incremental de las violaciones de restricciones en los datos enlazados de la tabla SALES y renueve de modo incremental la tabla SALES_SUMMARY dependiente. La opción ALLOW WRITE ACCESS se utiliza en ambas tablas para permitir las actualizaciones simultáneas mientras se realiza la comprobación de integridad.

```
ALTER TABLE SALES
  ATTACH PARTITION STARTING (100) ENDING (200)
  FROM SOURCE;
SET INTEGRITY FOR SALES ALLOW WRITE ACCESS, SALES_SUMMARY ALLOW WRITE ACCESS
  IMMEDIATE CHECKED FOR EXCEPTION IN SALES
  USE SALES_EXCEPTIONS;
```

Ejemplo 11: Desenlace una partición de una tabla particionada de datos denominada SALES. Renueve de manera incremental la tabla SALES_SUMMARY dependiente.

```
ALTER TABLE SALES
  DETACH PARTITION 2000_PART INTO ARCHIVE_TABLE;
SET INTEGRITY FOR SALES_SUMMARY
  IMMEDIATE CHECKED;
```

Ejemplo 12: Una nueva tabla de consultas materializadas mantenida por el usuario se quita del estado de pendiente de establecimiento de integridad.

```
CREATE TABLE YEARLY_SALES
  AS (SELECT YEAR, SUM(SALES)AS SALES
  FROM FACT_TABLE GROUP BY YEAR)
  DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY USER

SET INTEGRITY FOR YEARLY_SALES
  ALL IMMEDIATE UNCHECKED
```

SET PASSTHRU

La sentencia SET PASSTHRU abre y cierra una sesión para someter directamente el SQL nativo de una fuente de datos a esa fuente de datos. La sentencia no está bajo el control de la transacción.

Invocación

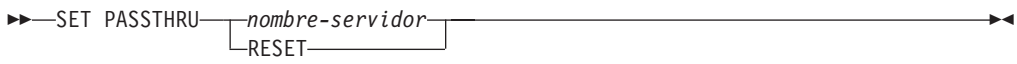
Esta sentencia puede emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben proporcionar autorización para:

- Realizar un paso a través para la fuente de datos
- Satisfacer las medidas de seguridad en la fuente de datos

Sintaxis



Descripción

nombre-servidor
Indica la fuente de datos para la que se debe abrir una sesión de paso a través. *nombre-servidor* debe identificar una fuente de datos que esté descrita en el catálogo.

RESET
Cierra una sesión de paso a través.

Notas

- A las fuentes de datos de Microsoft SQL Server, Sybase y Oracle se aplican las restricciones siguientes:
  - Las transacciones definidas por el usuario no pueden utilizarse para fuentes de datos de Microsoft SQL Server y Sybase en modalidad de paso a través, pues Microsoft SQL Server y Sybase restringen qué sentencias de SQL pueden especificarse dentro de una transacción definida por el usuario. Puesto que DB2 no analiza las sentencias de SQL que se procesan en modalidad de paso a través, no es posible detectar si el usuario ha especificado una sentencia de SQL que está permitida dentro de una transacción definida por el usuario.
  - La cláusula COMPUTE no recibe soporte en fuentes de datos de Microsoft SQL Server y Sybase.
  - Las sentencias de DDL no están sujetas a la semántica de transacción en fuentes de datos de Microsoft SQL Server, Oracle y Sybase. Microsoft SQL Server, Oracle o Sybase confirman la operación automáticamente cuando ésta se ha completado. Si se produce una retrotracción, el DDL no se retrotrae.

Ejemplos

Ejemplo 1: Inicie una sesión de paso a través para la fuente de datos BACKEND.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
```

Ejemplo 2: Inicie una sesión de paso a través con una sentencia PREPARE.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;
```

Ejemplo 3: Finalice una sesión de paso a través.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

Ejemplo 4: Utilice las sentencias PREPARE y EXECUTE para finalizar una sesión de paso a través.

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

Ejemplo 5: Abra una sesión para que tenga lugar el paso a través para una fuente de datos, cree un índice de clúster para una tabla en esta fuente de datos y cierre la sesión de paso a través.

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT                 modalidad de paso a través
  FROM "CREATE UNIQUE
        CLUSTERED INDEX TABLE_INDEX
        ON USER2.TABLE               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.

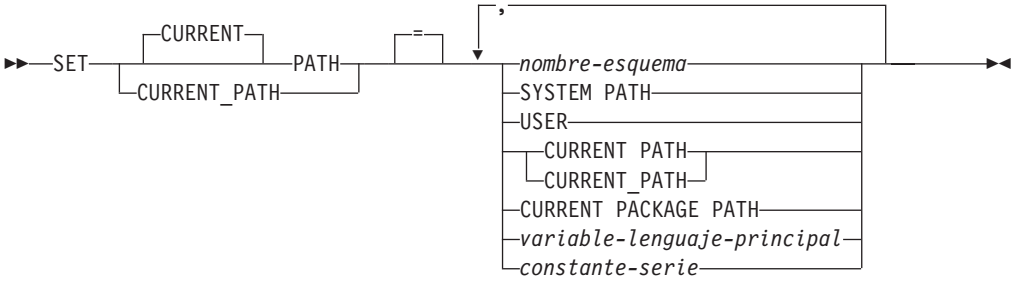
Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-esquema
Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. No se efectúa ninguna validación de que existe el esquema en el momento en que se establece la vía de acceso. Si, por ejemplo, un *nombre-esquema* está mal escrito, el error no se capturará y podría afectar a las operaciones posteriores de SQL.

SYSTEM PATH
Este valor equivale a especificar los nombres de esquema *"SYSIBM","SYSFUN","SYSPROC","SYSIBMADM"*.

USER
El valor del registro especial USER.

CURRENT PATH
El valor del registro especial CURRENT PATH antes que se ejecute esta sentencia.

CURRENT PACKAGE PATH
El valor del registro especial CURRENT PACKAGE PATH.

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lenguaje-principal* no debe ser mayor 128 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lenguaje-principal* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lenguaje-principal*, deben especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea que estén ya que no se realiza ninguna conversión a caracteres en mayúsculas.

constante-serie
Una constante de serie de caracteres con una longitud máxima de 128 bytes.

Normas

- Un nombre de esquema no puede aparecer más de una vez en la vía de acceso a SQL (SQLSTATE 42732).
- El 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 2048 bytes (SQLSTATE 42907).

Notas

- El valor inicial del registro especial CURRENT PATH es "SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","X" donde la X es el valor del registro especial USER.
- No es necesario especificar el esquema SYSIBM. Si no se incluye en la vía de acceso de SQL, se supone implícitamente que es el primer esquema (en este caso, no se incluye en el registro especial CURRENT PATH).
- El registro especial CURRENT PATH especifica la vía de acceso 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.
- ***Compatibilidades***
  - Para mantener la compatibilidad con las versiones anteriores de DB2:
    - CURRENT FUNCTION PATH puede utilizarse en lugar de CURRENT PATH

Ejemplos

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT PATH.

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

Ejemplo 2: El ejemplo siguiente recupera el valor actual del registro especial CURRENT PATH en la variable del lenguaje principal denominada CURPATH.

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

El valor sería "FERMAT","McDrw #8","SYSIBM" si se estableciese por el ejemplo anterior.

SET ROLE

La sentencia SET ROLE comprueba si el ID de autorización de la sesión es un miembro de una función específica. Un ID de autorización llega a ser miembro de una función cuando dicha función se otorga al ID de autorización o a un grupo o función en el que es miembro el ID de autorización.

Invocación

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

Autorización

No se necesita.

Sintaxis

Descripción

nombre-función
Especifica un ID de función en la que ha de verificarse su pertenencia como miembro del ID de autorización de la sesión. El *nombre-función* debe identificar una función existente en el servidor actual (SQLSTATE 42704). Si el ID de autorización de la sesión no es miembro de *nombre-función*, se devuelve un error (SQLSTATE 42501).

Notas

- Todas las funciones que se han otorgado a un ID de autorización se utilizan para la comprobación de autorizaciones. La sentencia SET ROLE no afecta a las funciones que se utilizan para esta comprobación de autorizaciones. Utilice las sentencias GRANT ROLE y REVOKE ROLE para cambiar las funciones de las que es miembro un ID de autorización.

Ejemplos

Ejemplo 1: Al usuario WALID se le ha otorgado la función EDITOR, pero no la función AUTHOR. Compruebe que WALID sea miembro de la función EDITOR.

```
SET ROLE EDITOR
```

Ejemplo 2: Compruebe que WALID no sea miembro de la función AUTHOR. La siguiente sentencia devuelve un error (SQLSTATE 42501).

```
SET ROLE AUTHOR
```

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.

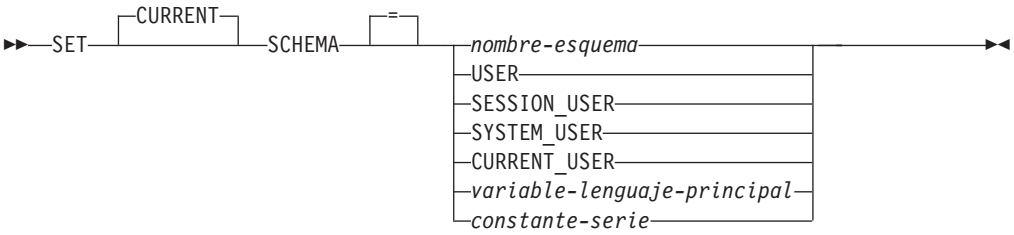
Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

No se necesita.

Sintaxis



Descripción

nombre-esquema
Este nombre, que consta de una sola parte, identifica un esquema que existe en el servidor de aplicaciones. La longitud no debe ser mayor que 128 bytes (SQLSTATE 42815). No se efectúa ninguna validación de que existe el esquema en el momento en que se establece el esquema. Si *nombre-esquema* está escrito incorrectamente, no se capturará y ello puede afectar a la forma en la que se ejecutan las sentencias 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-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de la *variable-lenguaje-principal* no debe ser mayor que 128 bytes (SQLSTATE 42815). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 42815).

Los caracteres de la *variable-lenguaje-principal* deben estar justificados por la izquierda. Cuando se especifica el *nombre-esquema* con una *variable-lenguaje-principal*, deben especificarse todos los caracteres en mayúsculas o en minúsculas exactamente tal como se desea que estén ya que no se realiza ninguna conversión a caracteres en mayúsculas.

constante-serie
Una constante de serie de caracteres con una longitud máxima de 128 bytes.

Normas

- Si el valor especificado no se adapta a las normas para un *nombre-esquema*, se produce un error (SQLSTATE 3F000).
- El valor del registro especial CURRENT SCHEMA se utiliza como nombre de esquema en todas las sentencias de SQL dinámico, con la excepción de la sentencia CREATE SCHEMA, donde existe una referencia no calificada a un objeto de base de datos.
- La opción de enlace QUALIFIER especifica el nombre de esquema que debe utilizarse como calificador para los nombres de objetos de base de datos no calificados en las sentencias de SQL estático.

Notas

- El valor inicial del registro especial CURRENT SCHEMA equivale a USER.
- El establecimiento del registro especial CURRENT SCHEMA no afecta al registro especial CURRENT PATH. En consecuencia, el registro especial CURRENT SCHEMA no se incluirá en la vía de acceso de SQL y es posible que la resolución de funciones, procedimientos y tipos definidos por el usuario no encuentre estos objetos. Para incluir el valor del esquema actual en la vía de acceso de SQL, siempre que se emita la sentencia SET SCHEMA, emita también la sentencia SET PATH incluyendo el nombre de esquema de la sentencia SET SCHEMA.
- CURRENT SQLID se acepta como sinónimo de CURRENT SCHEMA y el efecto de una sentencia SET CURRENT SQLID será idéntico al de una sentencia SET CURRENT SCHEMA. No tendrán lugar otros efectos, como, por ejemplo, cambios de autorización de la sentencia.

Ejemplos

Ejemplo 1: La sentencia siguiente establece el registro especial CURRENT SCHEMA.

```
SET SCHEMA RICK
```

Ejemplo 2: El ejemplo siguiente recupera el valor actual del registro especial CURRENT SCHEMA en la variable del lenguaje principal denominada CURSCHEMA.

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

El valor sería RICK, establecido en el ejemplo anterior.

SET SERVER OPTION

La sentencia SET SERVER OPTION especifica un valor de opción de servidor que debe permanecer en vigor mientras un usuario o una aplicación esté conectado a la base de datos federada. Cuando finaliza la conexión, se reinstaura el valor anterior de esta opción de servidor. La sentencia no está bajo el control de la transacción.

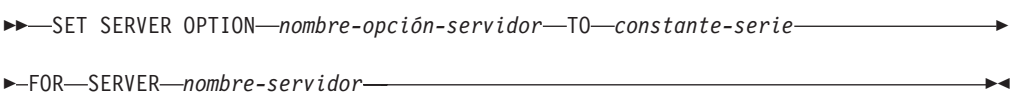
Invocación

Esta sentencia puede emitirse interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Autorización

No se necesita.

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.
- En el caso del SQL estático, si se utiliza la sentencia SET SERVER OPTION sólo se afecta a la ejecución de la sentencia de SQL estático. Si se utiliza la sentencia SET SERVER OPTION, no tiene ningún efecto en los planes generados por el optimizador.

Ejemplos

Ejemplo 1: Se ha definido una fuente de datos 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,"select c1 from ora_t1 where c1 > 100"); /*Generar indicaciones de plan*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

Ejemplo 2: Ha establecido la opción de servidor PASSWORD en 'Y' (validando las contraseñas en la fuente de datos) para todas las fuentes de datos de Oracle 8. Sin embargo, para una sesión en particular en la que una aplicación se conecta con la base de datos federada para acceder a una fuente de datos de Oracle 8 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;
```

SET SESSION AUTHORIZATION

La sentencia SET SESSION AUTHORIZATION cambia el valor del registro especial SESSION_USER. La sentencia no está bajo el control de la transacción. La sentencia SET SESSION AUTHORIZATION tiene como objetivo proporcionar soporte para un solo usuario, suponiendo que los ID de autorización son distintos en la misma conexión y no se debe utilizar en escenarios en los que distintos usuarios reutilizan la misma conexión, lo que normalmente se denomina agrupación de conexiones.

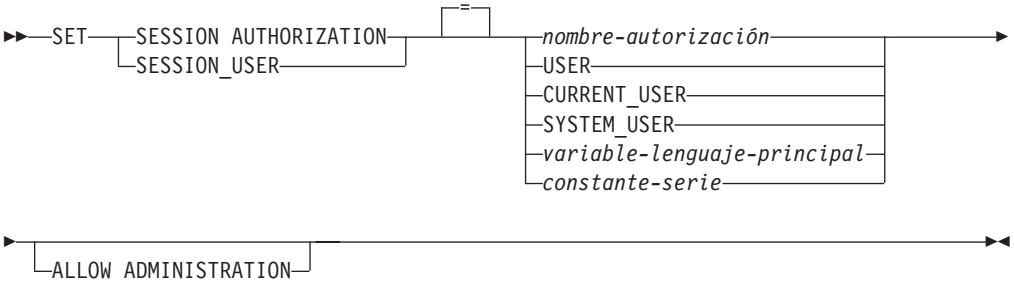
Invocación

La sentencia puede incorporarse en un programa de aplicación o emitirse interactivamente. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

Los privilegios que tiene el ID de autorización de la sentencia deben incluir SETSESSIONUSER en el valor de ID de autorización en el que se establece el registro especial.

Sintaxis



Descripción

nombre-autorización
Especifica el ID de autorización que se debe utilizar como el nuevo valor para el registro especial SESSION_USER.

USER
El valor del registro especial USER.

CURRENT_USER
El valor del registro especial CURRENT USER.

SYSTEM_USER
El valor del registro especial SYSTEM_USER.

variable-lenguaje-principal
Una variable de tipo CHAR o VARCHAR. La longitud del contenido de *variable-lenguaje-principal* no debe ser mayor 128 bytes (SQLSTATE 28000). No puede establecerse en nulo. Si una *variable-lenguaje-principal* tiene asociada una variable indicadora, el valor de dicha variable indicadora no debe indicar un valor nulo (SQLSTATE 28000).

Los caracteres de *variable-lenguaje-principal* deben estar justificados por la izquierda. Cuando se especifica un *nombre-autorización* con una variable de lenguaje principal, todos los caracteres se deben especificar en mayúsculas, porque no hay conversión a caracteres en mayúsculas.

constante-serie
Una constante de serie de caracteres con una longitud máxima de 128 bytes.

ALLOW ADMINISTRATION
Indica que se pueden especificar sentencias de esquema SQL antes de esta sentencia en al misma unidad de trabajo.

Normas

- El valor especificado para el registro especial SESSION_USER que cumple con las normas de un ID de autorización de tipo USER (SQLSTATE 42602).
- La opción de vinculación OWNER especifica el ID de autorización que se debe utilizar para sentencias de SQL estáticas.
- Esta sentencia sólo se puede emitir como la primera sentencia (que no sea una sentencia de registro especial SET) en una nueva unidad de trabajo sin ningún cursor WITH HOLD abierto (SQLSTATE 25001). Esta restricción incluye cualquier petición PREPARE correspondiente a una sentencia que no sea una sentencia de registro especial SET.
- El valor del registro especial SESSION_USER se utiliza como el ID de autorización para todas las sentencias de SQL dinámico de un paquete vinculado con la opción de vinculación DYNAMICRULES(RUN). (Esto incluye INVOKERUN y DEFINERUN cuando el paquete no lo utiliza ninguna rutina). Si un paquete utiliza autorización de propietario, invocador o definidor basado en la opción DYNAMICRULES, esta sentencia no tiene ningún efecto sobre las sentencias de SQL dinámico emitidas desde dentro del paquete.

Notas

- La sentencia SET SESSION AUTHORIZATION le permite modificar el ID de autorización de sesión. El ID de autorización de sesión representa el usuario actual de la conexión y es el ID de autorización que DB2 tiene en cuenta para todos los controles de autorizaciones relativos al SQL dinámico dentro de un paquete de ejecución DYNAMICRULES. El registro especial SESSION_USER se puede utilizar para visualizar el valor actual de este ID de autorización de sesión.
- El valor inicial del registro especial SESSION_USER para una nueva conexión es el mismo que el valor del registro especial SYSTEM_USER.
- La información de grupo correspondiente al ID de autorización de sesión especificado en esta sentencia se adquiere en el momento de ejecutar la sentencia.
- El hecho de establecer el registro especial SESSION_USER no afecta a los registros especiales CURRENT SCHEMA ni CURRENT PATH.
- Si se produce algún error durante el establecimiento del registro especial SESSION_USER, el registro vuelve a tomar su valor anterior.
- Esta sentencia no se debe utilizar para permitir que varios usuarios diferentes reutilicen la misma conexión, puesto que cada usuario heredará la capacidad de cambiar el valor del registro especial SESSION_USER que tenía el propietario original de la conexión. Esta sentencia depende del valor de SYSTEM_USER correspondiente a la comprobación de privilegios y la sentencia SET SESSION AUTHORIZATION no modifica el ID de autorización de conexión inicial.

Además, esta sentencia no está relacionada con los siguientes comportamientos que afectan a la reutilización de la conexión:
  - El privilegio CONNECT no se comprueba para el nuevo ID de autorización
  - El contenido de cualquier registro especial actualizable no se restablece; en concreto, el contenido del registro especial ENCRYPTION PASSWORD no se modifica y está disponible para el nuevo ID de autorización para las funciones de cifrado y descifrado.
  - El contenido de cualquier tabla temporal global declarada no se ve afectado y resulta accesible para el nuevo ID de autorización
  - Los enlaces existentes con servidores remotos no se restablecerán
- Si se especifica la cláusula ALLOW ADMINISTRATION, los siguientes tipos de sentencias u operaciones pueden preceder la sentencia SET SESSION AUTHORIZATION:
  - Lenguaje de definición de datos (DDL), incluida la definición de los puntos de salvaguarda y la declaración de tablas temporales globales, pero no incluyendo SET INTEGRITY
  - Sentencias GRANT y REVOKE
  - LOCK TABLE, sentencia
  - Sentencias COMMIT y ROLLBACK
  - SET de registros especiales
  - SET de variables globales

Ejemplos

Ejemplo 1: La siguiente sentencia establecer el registro especial SESSION_USER.

```
SET SESSION_USER = RAJIV
```

Ejemplo 2: Establezca el ID de autorización de sesión (el registro especial SESSION_USER) de modo que sea un valor del ID de autorización del sistema, que es el ID que ha establecido la conexión en la que se ha emitido la sentencia.

```
SET SESSION AUTHORIZATION SYSTEM_USER
```

SET variable

La sentencia SET variable asigna valores a variables. La sentencia no está bajo el control de la transacción.

Invocación

Esta sentencia puede incorporarse en un programa de aplicación o emitirse de forma interactiva. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

Para hacer referencia a una variable de transición, 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 las columnas a las que se hace referencia en la parte derecha
- Privilegio CONTROL sobre la tabla (tabla sujeto del activador)
- Autorización SYSADM o DBADM

Si se hace referencia a una variable global en la parte derecha de la sentencia de asignación, los privilegios que el ID de autorización de la sentencia tiene deben incluir:

- el privilegio READ sobre la variable global

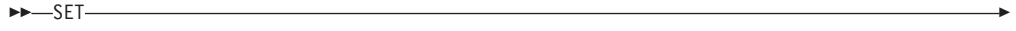
Si se asigna un valor a una variable global en la parte izquierda de la sentencia de asignación, los privilegios del ID de autorización de la sentencia deben incluir:

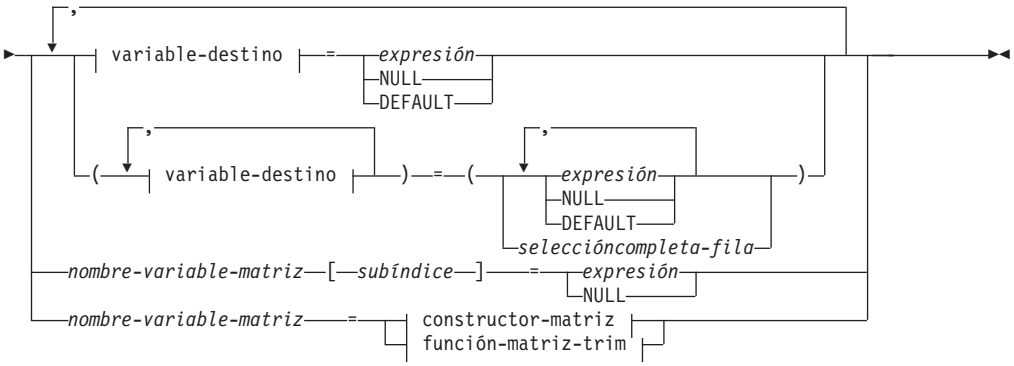
- el privilegio WRITE sobre la variable global

Para ejecutar esta sentencia con una *selecciónconpleta-fila* como parte derecha de la asignación, los privilegios del ID de autorización de la sentencia también debe incluir, como mínimo, uno de los siguientes, para cada tabla o vista:

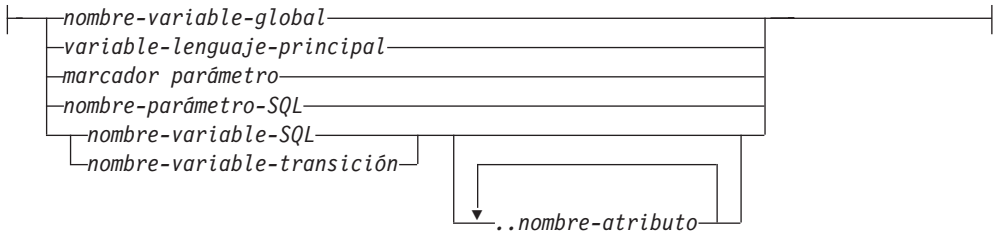
- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM

Sintaxis

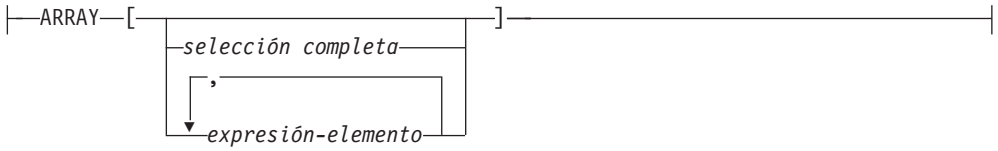




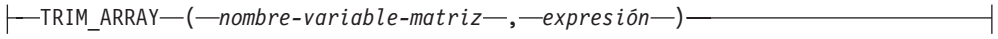
variable-destino:



constructor-matriz:



función-matriz-trim:



Descripción

variable-destino

Identifica la variable de destino de la asignación. Una *variable-destino* que represente la misma variable no debe especificarse más de una vez (SQLSTATE 42701).

nombre-variable-global

Identifica la variable global que es el sujeto de la asignación. El *nombre-variable-global* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

variable-lenguaje-principal

Identifica la variable del lenguaje principal que es el sujeto de la asignación.

marcador-parámetro

Identifica el marcador de parámetro que es el sujeto de la asignación.

nombre-parámetro-SQL

Identifica el parámetro que es el sujeto de la asignación. El parámetro se

debe especificar en la *declaración-parámetro* de la sentencia CREATE PROCEDURE y debe estar definido como parámetro OUT o INOUT.

nombre-variable-SQL
Identifica la variable SQL que es el sujeto de la asignación. Las variables SQL se deben declarar antes de utilizarlas.

nombre-variable-transición
Identifica la columna que se debe actualizar en la fila de transición. Un *nombre-variable-transición* debe identificar una columna en la tabla sujeto de un activador, calificado opcionalmente por un nombre de correlación que identifique el nuevo valor (SQLSTATE 42703).

..nombre atributo
Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*). El *nombre-variable-SQL* o *nombre-variable-transición* especificado debe definirse con 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 incluya la cláusula *..nombre atributo* se denomina asignación convencional.

nombre-variable-matriz
Identifica una variable SQL o parámetro. La variable o el parámetro deben ser de tipo de matriz (SQLSTATE 428H0).

[*subíndice***]**
Expresión numérica que especifica qué elemento de la matriz es el sujeto de la asignación. El subíndice debe ser de tipo numérico exacto con escala cero (SQLSTATE 428H1); no puede ser nulo. Su valor debe estar entre 1 y la cardinalidad máxima definida para la matriz (SQLSTATE 2202E).

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).

constructor-matriz
El constructor de matriz se utiliza para componer un valor para el tipo de datos de la matriz.

selección completa
Una selección completa que devuelve una única columna. Los valores que la selección completa devuelve se convierten en elementos de la matriz. La cardinalidad de la matriz es igual al número de filas que la selección completa devuelve. Se puede utilizar una cláusula ORDER BY de la selección completa para especificar el orden de los elementos de la matriz; de lo contrario, el orden no está definido. El tipo base de la matriz se determina en función de las “Normas para tipos de datos de resultado” y debe encontrarse entre los tipos permitidos de la sentencia CREATE TYPE (matriz) (SQLSTATE 429C2).

expresión-elemento
Expresión que define el valor de un elemento de la matriz. La cardinalidad de la matriz es igual al número de expresiones de elementos. La primera *expresión-elemento* se asigna al elemento de la matriz con el subíndice 1. La

segunda *expresión-elemento* se asigna al elemento de la matriz con el subíndice 2, y así sucesivamente. Todas las expresiones de elementos deben tener tipos de datos compatibles (SQLSTATE 429C2). El tipo base de la matriz se determina en función de las "Normas para tipos de datos de resultado" y debe encontrarse entre los tipos permitidos de la sentencia CREATE TYPE (matriz) (SQLSTATE 429C2).

función-matriz-trim
Devuelve una copia del argumento de la matriz del que se ha eliminado el número especificado de elementos del final de la matriz.

nombre-variable-matriz
Identifica una variable SQL o parámetro. La variable o el parámetro deben ser de tipo de matriz (SQLSTATE 428H0).

expresión
Expresión numérica que especifica el número de elementos que se recortarán de la copia de *nombre-variable-matriz*. La *expresión* debe estar entre 0 y la cardinalidad de *nombre-variable-matriz* (SQLSTATE 2202E). La cardinalidad del resultado disminuye por el número de elementos recortados. El resultado puede ser nulo; si el argumento es nulo, el resultado es el valor nulo.

NULL
Especifica el valor nulo. NULL no puede ser el valor en una asignación de atributos a menos que se convirtiera específicamente al tipo de datos del atributo (SQLSTATE 429B9).

DEFAULT
Especifica que se debe utilizar el valor por omisión.

En procedimientos SQL, la cláusula DEFAULT puede especificarse sólo para sentencias de SQL estático. La excepción es que la cláusula DEFAULT puede especificarse cuando la *variable-destino* es una variable global de una sentencia de SQL dinámico.

Si *variable-destino* es una columna, el valor insertado depende de cómo se haya definido la columna en la tabla.

- Si la columna se ha definido utilizando la cláusula WITH DEFAULT, el valor se establece en el valor por omisión que se ha definido para la columna (consulte *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.

Si la *variable-destino* es una variable global, el valor que se ha insertado es el valor por omisión, tal como se ha especificado en la creación de la variable.

Si la *variable-destino* es una variable SQL o un parámetro SQL de un procedimiento de SQL, una variable del lenguaje principal o un marcador de parámetro, no puede especificarse la palabra clave DEFAULT (SQLSTATE 42608).

selecciónpcompleta-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).
- Las variables globales no pueden ser asignadas en activadores, funciones, métodos o sentencias compuestas dinámicas o en un procedimiento al que uno de estos objetos llame de forma directa o indirecta (SQLSTATE 428GX)
- Si el valor que se asigna es una matriz resultante de un constructor de matriz o de ARRAY_AGG, los tipos base de la matriz y de la variable de destino deben ser idénticos (SQLSTATE 42821).
- Si la asignación es de la forma `SET A[idx] = rhs`, donde *A* es un nombre de variable de matriz, *idx* es una expresión utilizada como subíndice y *rhs* es una expresión del mismo tipo que el elemento de la matriz, entonces:
  1. Si la matriz *A* es el valor nulo, defina *A* en la matriz vacía.
  2. *C* tiene que ser la cardinalidad de la matriz *A*.
  3. Si *idx* es menor que o igual a *C*, el valor en la posición que *idx* identifica se sustituye por el valor de *rhs*.
  4. Si *idx* es mayor que *C*, entonces:
     a. El valor en posición *i*, dado que *i* es mayor que *C* y menor que *idx*, se establece en el valor nulo.
     b. El valor en posición *idx* se establece en el valor de *rhs*.
     c. La cardinalidad de *A* se establece en *idx*.

Notas

- 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. Las asignaciones de series utilizan normas de asignación de recuperación.
- Si una variable se ha declarado con un identificador que coincide con el nombre de un registro especial (por ejemplo, PATH), se debe delimitar la variable para impedir una asignación no intencionada al registro especial (por ejemplo, `SET "PATH" = 1;` para una variable llamada PATH que se ha declarado como entero).

- Si se incluye más de una asignación, se evalúa cada *expresión* y *selección-completa-fila* antes de que se realicen las asignaciones. Por lo tanto, las referencias a variables de destino en una expresión o una selección completa de fila 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 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
```

Ejemplo 7: establezca la variable de matriz de SQL p_phonenumbers en una matriz de números fijos.

```
SET p_phonenumbers = ARRAY[9055553907, 4165554213, 4085553678]
```

Ejemplo 8: establezca la variable de matriz de SQL p_phonenumbers en un matriz de números recuperados de la tabla PHONENUMBER.

```
SET p_phonenumbers = ARRAY[SELECT NUMBER
  FROM PHONENUMBERS
  WHERE EMPID = 624]
```

Ejemplo 9: elimine el último elemento de la variable de matriz de SQL p_phonenumbers.

```
SET p_phonenumbers = TRIM_ARRAY(p_phonenumbers, 1)
```

Ejemplo 10: asigne p_mynumber al primer y décimo elemento de la variable de matriz p_phonenumbers. Después de la primera asignación, la cardinalidad de p_phonenumbers es 1. Después de la segunda asignación, la cardinalidad es 10 y se ha asignado implícitamente a los elementos del 1 al 9 el valor de nulo.

```
SET p_phonenumbers[1] = p_mynumber

SET p_phonenumbers[10] = p_mynumber
```

Ejemplo 11: a partir de una tabla llamada SECURITY.USERS, que tiene una fila para cada usuario que puede conectarse a la base de datos, asigne la hora actual y el nivel de autorización a las variables globales userinfo.gv_connect_time y userinfo.gv_auth_level, respectivamente.

```
SET userinfo.gv_connect_time = CURRENT TIMESTAMP,
  userinfo.gv_auth_level = (
    SELECT AUTHLEVEL FROM SECURITY.USERS
      WHERE USERID = CURRENT USER)
```

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.

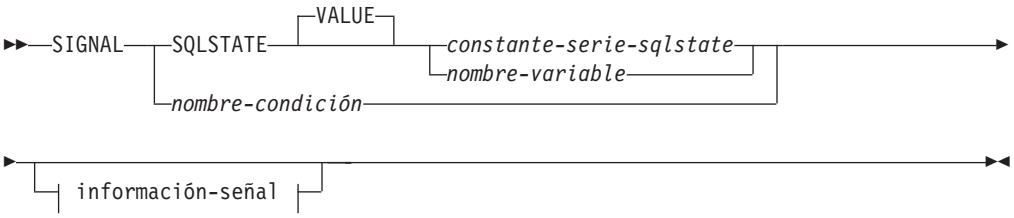
Invocación

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.

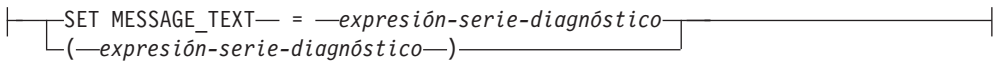
Autorización

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 QLSTATE:

- 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). El valor en tiempo de ejecución de sentencia no debe ser nulo y se debe ajustar a 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 declararse dentro de la sentencia compuesta que contiene la sentencia SIGNAL o dentro de una sentencia compuesta en la que dicha sentencia compuesta esté anidada (SQLSTATE 42737).

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 de SQL para un sistema de gestión de pedidos que señala un error de aplicación cuando la aplicación no reconoce un número de cliente. La tabla ORDERS incluye una clave foránea para la tabla CUSTOMER, lo cual hace necesario que exista el número de cliente (CUSTNO) para poder insertar un pedido.

```
CREATE PROCEDURE SUBMIT_ORDER
  (IN ONUM INTEGER, IN CNUM INTEGER,
   IN PNUM INTEGER, IN QNUM INTEGER)
   SPECIFIC SUBMIT_ORDER
   MODIFIES SQL DATA
  LANGUAGE SQL
  BEGIN
    DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
      SIGNAL SQLSTATE '75002'
        SET MESSAGE_TEXT = 'No se conoce el número del cliente';
    INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
      VALUES (ONUM, CNUM, PNUM, QNUM);
  END
```

TRANSFER OWNERSHIP

La sentencia TRANSFER OWNERSHIP transfiere la propiedad de un objeto de base de datos.

Invocación

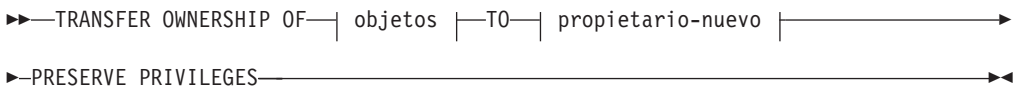
Esta sentencia puede incorporarse a un programa de aplicación o emitirse mediante el uso de sentencias de SQL dinámico. Es una sentencia ejecutable que puede prepararse de forma dinámica sólo si el comportamiento de ejecución de DYNAMICRULES está en vigor para el paquete (SQLSTATE 42509).

Autorización

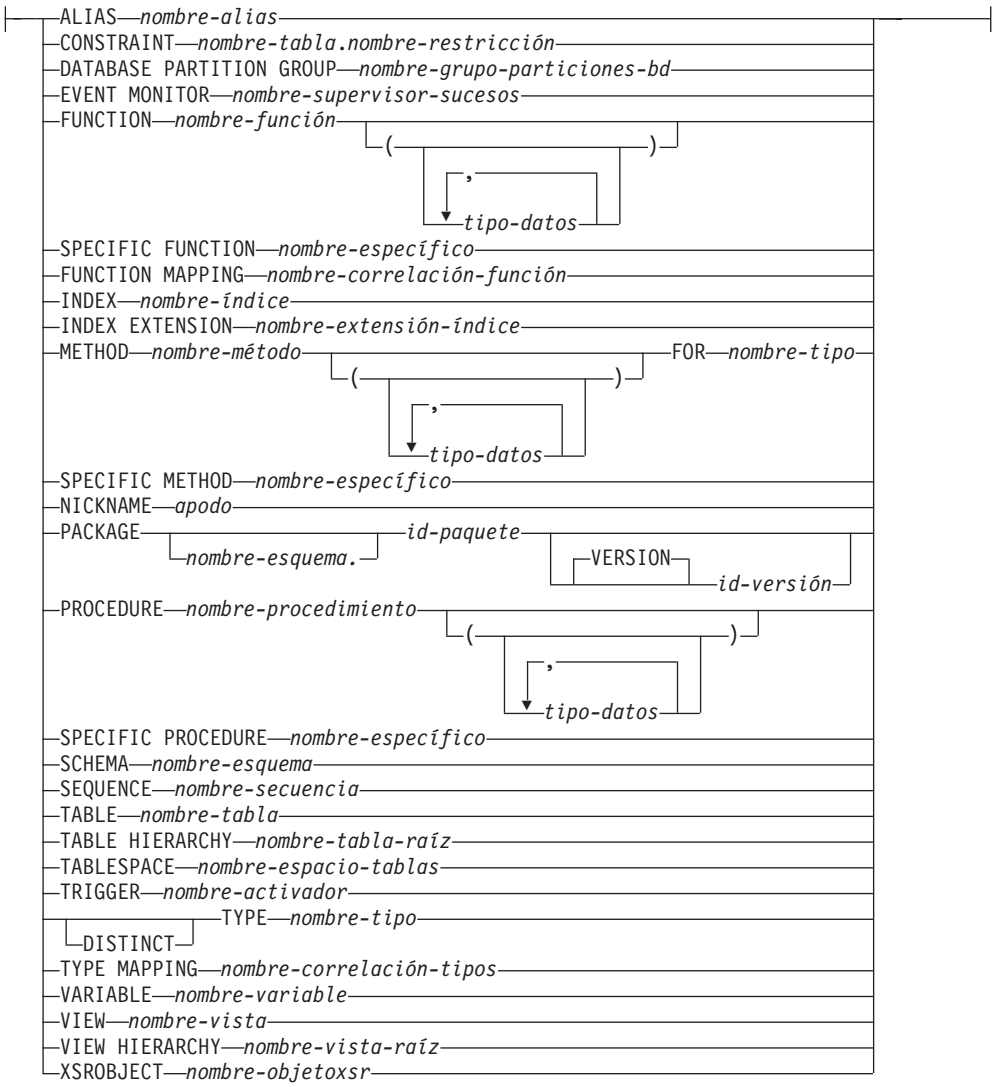
El ID de autorización de la sentencia debe tener al menos uno de los privilegios siguientes:

- Propiedad del objeto
- Autorización SECADM

Sintaxis



objetos:



propietario-nuevo:

Descripción

ALIAS *nombre-alias*
Identifica el alias cuya propiedad se debe transferir. El *nombre-alias* debe designar un alias descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del alias, el valor de la columna OWNER del alias de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.

CONSTRAINT *nombre-tabla.nombre-restricción*
Identifica la restricción cuya propiedad se debe transferir. La combinación *nombre-tabla.nombre-restricción* debe identificar una restricción y la tabla que restringe. El *nombre-restricción* debe identificar una restricción descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la restricción, el valor de la columna OWNER de la restricción de la vista de catálogo SYSCAT.TABCONST se sustituye por el ID de autorización del propietario nuevo.

- Si la restricción es una restricción FOREIGN KEY, la columna OWNER de la vista de catálogo SYSCAT.REFERENCES se sustituye por el ID de autorización del propietario nuevo.
- Si la restricción es una restricción PRIMARY KEY o UNIQUE, la columna OWNER de la vista de catálogo SYSCAT.INDEXES para el índice creado implícitamente para esta restricción se sustituye por el ID de autorización del propietario nuevo. Si el índice ya existía y se reutiliza en este caso, el propietario del índice no se cambia.

DATABASE PARTITION GROUP *nombre-grupo-particiones-bd*
Identifica el grupo de particiones de base de datos cuya propiedad se debe transferir. El *nombre-grupo-particiones-bd* debe identificar un grupo de particiones de base de datos descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del grupo de particiones de base de datos, el valor de la columna OWNER del grupo de particiones de base de datos de la vista de catálogo SYSCAT.DBPARTITIONGROUPS se sustituye por el ID de autorización del propietario nuevo.

EVENT MONITOR *nombre-supervisor-sucesos*
Identifica el supervisor de sucesos cuya propiedad se debe transferir. El *nombre-supervisor-sucesos* debe identificar un supervisor de sucesos que se haya descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del supervisor de sucesos, el valor de la columna OWNER del supervisor de sucesos de la vista de catálogo SYSCAT.EVENTMONITORS se sustituye por el ID de autorización del propietario nuevo.

Si el supervisor de sucesos identificado está activo, se devuelve un error (SQLSTATE 429BT).

Si hay archivos de sucesos en la vía de acceso de destino de un supervisor de sucesos WRITE TO FILE cuya propiedad se transfiere, los archivos de sucesos no se suprimirán.

Cuando se transfiere la propiedad de supervisores de sucesos WRITE TO TABLE, se conserva la información de tabla de la vista de catálogo SYSCAT.EVENTTABLES.

FUNCTION
Identifica la función cuya propiedad se debe transferir. La instancia de función especificada debe ser una función definida por el usuario o una plantilla de función descrita en el catálogo. La propiedad de las funciones generadas implícitamente por las sentencias CREATE TYPE (Diferenciado) y CREATE TYPE (estructurado) no se puede transferir (SQLSTATE 429BT).

Hay varias maneras de identificar la instancia de función.

FUNCTION *nombre-función*
Identifica la función específica cuya propiedad se debe transferir, y sólo es válido si hay exactamente una instancia de función con ese *nombre-función*. La función identificada de esta manera 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 sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el

calificador para nombres de objeto no calificados. Si no existe ninguna función con este nombre en el esquema implícito o especificado, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia específica de la función en el esquema especificado o implícito, se devuelve un error (SQLSTATE 42725).

FUNCTION *nombre-función* **(***tipo-datos***,...)**
Proporciona la signatura de la función, que identifica de forma exclusiva la función cuya propiedad se debe transferir. El algoritmo de selección de funciones no se utiliza.

nombre-función
Especifica el nombre de la función cuya propiedad se debe transferir. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados.

(tipo-datos,...)
Los tipos de datos especificados deben coincidir con los tipos y las posiciones que se han especificado en la sentencia CREATE FUNCTION. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la función específica cuya propiedad se debe transferir.

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.

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).

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.

El atributo FOR BIT DATA no se considera parte de la signatura por lo que respecta a la comparación de patrones. Así, por ejemplo, un CHAR FOR BIT DATA especificado en la signatura solo coincidiría con una función definida con CHAR, y viceversa.

Si no existe ninguna función con la signatura especificada en el esquema implícito o especificado, se devuelve un error (SQLSTATE 42883).

Cuando se transfiere la propiedad de la función, el valor de la columna OWNER de la función de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

SPECIFIC FUNCTION *nombre-específico*
Identifica la función definida por el usuario determinada cuya propiedad se debe transferir, utilizando el nombre concreto que se ha especificado o tomado por omisión al crear la función. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de función específica en el esquema especificado o implícito; de lo contrario, se devuelve un error (SQLSTATE 42704).

Cuando se transfiere la propiedad de la función específica, el valor de la columna OWNER de la función específica de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

FUNCTION MAPPING *nombre-correlación-funciones*
Identifica la correlación de funciones cuya propiedad se debe transferir. El *nombre-correlación-funciones* debe identificar una correlación de funciones que está descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la correlación de funciones, el valor de la columna OWNER de la correlación de funciones de la vista de catálogo SYSCAT.FUNCMAPPINGS se sustituye por el ID de autorización del propietario nuevo.

INDEX *nombre-índice*
Identifica el índice o la especificación de índice cuya propiedad se debe transferir. El *nombre-índice* debe identificar un índice o especificación de índice que se describa en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del índice, el valor de la columna OWNER del índice de la vista de catálogo SYSCAT.INDEXES se sustituye por el ID de autorización del propietario nuevo.

La propiedad de un índice no se puede transferir si la tabla en que está definido el índice es una tabla temporal global (SQLSTATE 429BT).

INDEX EXTENSION *nombre-extensión-índice*
Identifica la extensión de índice cuya propiedad se debe transferir. El *nombre-extensión-índice* debe identificar una extensión de índice que esté descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la extensión de índice, el valor de la columna OWNER de la extensión de índice de la vista de catálogo SYSCAT.INDEXEXTENSIONS se sustituye por el ID de autorización del propietario nuevo.

METHOD
Identifica el método cuya propiedad se debe transferir. El cuerpo del método especificado debe ser un método descrito en el catálogo (SQLSTATE 42704). La propiedad de los métodos generados implícitamente por la sentencia CREATE TYPE no se puede transferir (SQLSTATE 429BT).

Hay varias maneras de identificar el cuerpo del método.

METHOD *nombre-método*
Identifica el método específico cuya propiedad se debe transferir, 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 identificado de esta manera puede tener un número cualquiera de parámetros. Si no existe

ningún método con este nombre para el tipo *nombre-tipos*, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia específica del método para el tipo de datos especificado, se devuelve un error (SQLSTATE 42725).

METHOD *nombre-método* **(***tipo-datos***,...)**
Proporciona la signatura del método, que identifica de forma exclusiva el método cuya propiedad se debe transferir. El algoritmo de selección de métodos no se utiliza.

nombre-método
Especifica el nombre del método cuya propiedad se debe transferir. El nombre debe ser un identificador no calificado.

(*tipo-datos***,...)**
Los tipos de datos especificados deben coincidir con los tipos y las posiciones que se han especificado en la sentencia CREATE TYPE o ALTER TYPE. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar la instancia de método específica cuya propiedad se debe transferir.

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.

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).

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 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 esta signatura para el tipo de datos especificado, se devuelve un error (SQLSTATE 42883).

FOR *nombre-tipo*
Indica el nombre del tipo para el cual se debe transferir la propiedad del método especificado. El nombre debe identificar un tipo 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 sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de tipo no calificados.

Cuando se transfiere la propiedad del método, el valor de la columna OWNER del método de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

SPECIFIC METHOD *nombre-específico*
Identifica el método específico cuya propiedad se debe transferir. 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 sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para un nombre específico no calificado. El *nombre-específico* debe identificar un método; de lo contrario, se devuelve un error (SQLSTATE 42704).

Cuando se transfiere la propiedad del método específico, el valor de la columna OWNER del método específico de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

NICKNAME *apodo*
Identifica el apodo cuya propiedad se debe transferir. El *apodo* debe ser un apodo descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del apodo, el valor de la columna OWNER del apodo de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.

PACKAGE *nombre-esquema.id-paquete*
Identifica el paquete cuya propiedad se debe transferir. Si no se especifica un nombre de esquema, el esquema por omisión califica implícitamente el identificador del paquete. El nombre de esquema y el identificador de paquete, junto con el identificador de versión especificado implícita o explícitamente, deben identificar un paquete que esté descrito en el catálogo (SQLSTATE 42704).

VERSION *id-versión*
Identifica de qué versión del paquete se debe transferir la propiedad. Si no se especifica ningún valor, la versión se establece por omisión en la serie de caracteres vacía y se transfiere la propiedad de este paquete. Si existen varios paquetes con el mismo nombre de paquete pero con distintas versiones, sólo se transfiere la propiedad del paquete cuyo *id-versión* se ha especificado en la sentencia TRANSFER OWNERSHIP. Delimite el identificador de versión con comillas dobles cuando:

- Se genera mediante la opción del precompilador VERSION(AUTO)
- Comienza con un dígito
- Contiene minúsculas o mayúsculas y minúsculas

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

Cuando se transfiere la propiedad del paquete, el valor de la columna BOUNDBY del paquete de la vista de catálogo SYSCAT.PACKAGES se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de los paquetes asociados a procedimientos de SQL (SQLSTATE 429BT).

PROCEDURE
Identifica el procedimiento cuya propiedad se debe transferir. La instancia de procedimiento especificada debe ser un procedimiento descrito en el catálogo.

Hay varias maneras de identificar la instancia de procedimiento.

PROCEDURE *nombre-procedimiento*
Identifica el procedimiento específico cuya propiedad se debe transferir, 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 definidos. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. Si no existe ningún procedimiento con este nombre en el esquema implícito o especificado, se devuelve un error (SQLSTATE 42704). Si existe más de una instancia específica del procedimiento en el esquema especificado o implícito, se devuelve un error (SQLSTATE 42725).

PROCEDURE *nombre-procedimiento* **(***tipo-datos***,...)**
Proporciona la signatura del procedimiento, que identifica de forma exclusiva el procedimiento cuya propiedad se debe transferir.

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

(tipo-datos,...)
Los tipos de datos especificados deben coincidir con los tipos y las posiciones que se han especificado en la sentencia CREATE PROCEDURE. El número de tipos de datos y la concatenación lógica de los tipos de datos se utilizan para identificar el procedimiento específico cuya propiedad se debe transferir.

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.

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).

Sin embargo, si la longitud, la precisión o la escala está codificada, el valor debe coincidir exactamente con el que se especifica en la sentencia CREATE PROCEDURE.

No es necesario que un tipo 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 implícito o especificado, se devuelve un error (SQLSTATE 42883).

Cuando se transfiere la propiedad del procedimiento, el valor de la columna OWNER del procedimiento de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

Al transferir la propiedad de un procedimiento de SQL que tiene un paquete asociado, también se transfiere implícitamente la propiedad del paquete al propietario nuevo.

SPECIFIC PROCEDURE *nombre-específico*
Identifica el procedimiento determinado cuya propiedad se debe transferir, utilizando el nombre concreto que se ha especificado o tomado por omisión al crear el procedimiento. En las sentencias de SQL dinámico, el registro especial CURRENT SCHEMA se utiliza como calificador para un nombre de objeto no calificado. En sentencias de SQL estático, la opción de precompilación o de vinculación QUALIFIER especifica implícitamente el calificador para nombres de objeto no calificados. El *nombre-específico* debe identificar una instancia de procedimiento específica en el esquema especificado o implícito; de lo contrario, se devuelve un error (SQLSTATE 42704).

Cuando se transfiere la propiedad del procedimiento específico, el valor de la columna OWNER del procedimiento específico de la vista de catálogo SYSCAT.ROUTINES se sustituye por el ID de autorización del propietario nuevo.

SCHEMA *nombre-esquema*
Identifica el esquema cuya propiedad se debe transferir. El *nombre-esquema* debe identificar un esquema que se describe en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del esquema, el valor de la columna OWNER y la columna DEFINER para el esquema de la vista de catálogo SYSCAT.SCHEMATA se sustituyen por el ID de autorización del propietario nuevo.

La propiedad de esquemas definidos por el sistema (donde la persona que realiza la definición es SYSIBM) no se puede transferir (SQLSTATE 42832).

SEQUENCE *nombre-secuencia*
Identifica la secuencia cuya propiedad se debe transferir. El *nombre-secuencia* debe identificar una secuencia descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la secuencia, el valor de la columna OWNER del esquema de la vista de catálogo SYSCAT.SEQUENCES se sustituye por el ID de autorización del propietario nuevo.

TABLE *nombre-tabla*
Identifica la tabla cuya propiedad se debe transferir. El *nombre-tabla* debe identificar una tabla que exista en la base de datos (SQLSTATE 42704) y no puede identificar una tabla temporal declarada (SQLSTATE 42995).

Cuando se transfiere la propiedad de la tabla:

- El valor de la columna OWNER de la tabla de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la tabla de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de las subtablas de una jerarquía de tablas (SQLSTATE 429BT).

En un sistema federado, se puede transferir la propiedad de una tabla remota que se ha creado utilizando un DDL transparente. Al transferir la propiedad de una tabla remota, no se transfiere la propiedad del apodo asociado a la tabla. La propiedad de un apodo de este tipo se puede transferir explícitamente con la sentencia TRANSFER OWNERSHIP.

TABLE HIERARCHY *nombre-tabla-raíz*
Identifica la tabla con tipo que es la tabla raíz de una jerarquía de tablas con tipo cuya propiedad se debe transferir. El *nombre-tabla-raíz* debe identificar una tabla con tipo que es la tabla raíz de la jerarquía de tablas con tipo (SQLSTATE 428DR) y debe hacer referencia a una tabla con tipo existente en la base de datos (SQLSTATE 42704).

Cuando se transfiere la propiedad de la jerarquía de tablas:

- El valor de la columna OWNER de la tabla raíz y todas sus subtablas de la vista de catálogo SYSCAT.TABLES se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la tabla y todas sus subtablas de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

TABLESPACE *nombre-espacio-tablas*
Identifica el espacio de tablas cuya propiedad se debe transferir. El *nombre-espacio-tablas* debe identificar un espacio de tablas descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del espacio de tablas, el valor de la columna OWNER del espacio de tablas de la vista de catálogo SYSCAT.TABLESPACES se sustituye por el ID de autorización del propietario nuevo.

TRIGGER *nombre-activador*
Identifica el activador cuya propiedad se debe transferir. El *nombre-activadores* debe identificar un activador que se haya descrito en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del activador, el valor de la columna OWNER del activador de la vista de catálogo SYSCAT.TRIGGERS se sustituye por el ID de autorización del propietario nuevo.

TYPE *nombre-tipo*
Identifica el tipo definido por el usuario cuya propiedad se debe transferir. El *nombre-tipo* debe identificar un tipo descrito en el catálogo (SQLSTATE 42704). Si se especifica DISTINCT, *nombre-tipo* debe identificar un tipo diferenciado que esté descrito en el catálogo (SQLSTATE 42704).

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

Cuando se transfiere la propiedad del tipo, el valor de la columna OWNER del tipo de la vista de catálogo SYSCAT.DATATYPES se sustituye por el ID de autorización del propietario nuevo.

TYPE MAPPING *nombre-correlación-tipos*
Identifica la correlación de tipos de datos definida por el usuario cuya propiedad se debe transferir. El *nombre-correlación-tipos* debe identificar una correlación de tipos de datos que esté descrita en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la correlación de tipos, el valor de la columna OWNER de la correlación de tipos de la vista de catálogo SYSCAT.TYPEMAPPINGS se sustituye por el ID de autorización del propietario nuevo.

VARIABLE *nombre-variable*
Indica que el objeto cuya propiedad se va a transferir es una variable global creada. El *nombre-variable* debe identificar una variable global que exista en el servidor actual (SQLSTATE 42704).

Cuando se transfiere la variable global, el valor de la columna OWNER de la variable global de la vista de catálogo SYSCAT.VARIABLES se sustituye por el ID de autorización del propietario nuevo.

VIEW *nombre-vista*
Identifica la vista cuya propiedad se debe transferir. El *nombre-vista* debe identificar una vista existente en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad de la vista:

- El valor de la columna OWNER de la vista de la vista de catálogo SYSCAT.VIEWS se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la vista de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

No se puede transferir la propiedad de una subvista de una jerarquía de vistas (SQLSTATE 429BT).

VIEW HIERARCHY *nombre-vista-raíz*
Identifica la vista con tipo que es la vista raíz de una jerarquía de vistas con tipo cuya propiedad se debe transferir. El *nombre-vista-raíz* debe identificar una vista con tipo que es la vista raíz de la jerarquía de vistas con tipo (SQLSTATE 428DR) y debe hacer referencia a una vista con tipo existente en la base de datos (SQLSTATE 42704).

Cuando se transfiere la propiedad de la jerarquía de vistas:

- El valor de la columna OWNER de la vista raíz y todas sus subvistas de la vista de catálogo SYSCAT.VIEWS se sustituye por el ID de autorización del propietario nuevo.
- El valor de la columna OWNER de todos los objetos dependientes de la vista y todas sus subvistas de la vista de catálogo SYSCAT.TABDEP se sustituye por el ID de autorización del propietario nuevo.

XSROBJECT *nombre-objetoxsr*
Identifica el objeto XSR cuya propiedad se debe transferir. El *nombre-objetoxsr* debe identificar un objeto XSR que se describe en el catálogo (SQLSTATE 42704).

Cuando se transfiere la propiedad del objeto XSR, el valor de la columna OWNER del objeto XSR de la vista de catálogo SYSCAT.XSROBJECTS se sustituye por el ID de autorización del propietario nuevo.

USER *nombre-autorización*
Especifica el ID de autorización al cual se transfiere la propiedad del objeto.

SESSION_USER
Especifica que el valor del registro especial SESSION_USER se debe utilizar como el ID de autorización al cual se transfiere la propiedad del objeto.

SYSTEM_USER
Especifica que el valor del registro especial SYSTEM_USER se debe utilizar como el ID de autorización al cual se transfiere la propiedad del objeto.

PRESERVE PRIVILEGES
Especifica que el propietario actual de un objeto cuya propiedad se va a transferir seguirá teniendo los privilegios actuales para el objeto tras la

transferencia. Por ejemplo, los privilegios que se otorgaron al creador de una vista cuando se creó esa vista seguirán en poder del propietario original incluso después de que se transfiera la propiedad a otro usuario.

Normas

- No se puede transferir la propiedad de la mayoría de los objetos definidos por el sistema (en que el propietario es SYSIBM) (SQLSTATE 42832). Sin embargo, a partir del fixpack 2, puede transferir la propiedad de objetos de esquema creados de forma explícita que tienen SYSIBM en la columna OWNER y que no tienen SYSIBM en la columna DEFINER.
- No se puede transferir la propiedad de esquemas cuyo nombre empieza por 'SYS' (SQLSTATE 42832).
- No se puede transferir explícitamente la propiedad de los objetos siguientes (SQLSTATE 429BT):
  - Subtablas de una jerarquía de tablas (se transfieren con la tabla de jerarquía raíz)
  - Subvistas de una jerarquía de vistas (se transfieren con la vista de jerarquía raíz)
  - Índices que están definidos en tablas temporales globales
  - Métodos o funciones que se generan implícitamente cuando se crea un tipo definido por el usuario
  - Paquetes que dependen de procedimientos de SQL (se transfieren con el procedimiento de SQL)
  - Supervisores de sucesos que están activos (se pueden transferir cuando no están activos)
- Un ID de autorización con la autorización SECADM no puede transferirse la propiedad de un objeto si no es ya el propietario del objeto (SQLSTATE 42502).

Notas

- Todos los privilegios del propietario actual que se otorgaron como parte de la creación del objeto se transfieren al nuevo propietario. Si se ha revocado un privilegio para el objeto al propietario actual, y posteriormente se ha vuelto a otorgar ese privilegio, el privilegio no se transfiere. A partir del fixpack 2, para objetos de esquema creados de forma implícita que todavía no se hayan transferido, al nuevo propietario se le otorgan los privilegios CREATEIN, DROPIN y ALTERIN sobre el esquema y también pueden otorgar estos privilegios a otros usuarios.
- Al transferir la propiedad de un objeto de base de datos, el nuevo propietario debe poseer el conjunto de privilegios sobre los objetos básicos, tal como indican las dependencias del objeto que son necesarias para mantener la existencia del objeto sin modificaciones. El nuevo propietario no necesita los privilegios para crear el objeto si dichos privilegios no son necesarios para mantener la existencia del objeto.

  Por ejemplo:

  - Imagine una vista con dependencias SELECT e INSERT sobre una tabla subyacente. Entre los privilegios que posee el propietario nuevo de la vista deben encontrarse como mínimo SELECT (con o sin GRANT OPTION) e INSERT (con o sin GRANT OPTION) para que se pueda transferir la propiedad correctamente. Si las dependencias son SELECT WITH GRANT OPTION e INSERT WITH GRANT OPTION, entre los privilegios que posee el propietario nuevo de la vista deben encontrarse como mínimo SELECT WITH GRANT OPTION e INSERT WITH GRANT OPTION.

– Imagine una vista con una dependencia sobre una rutina. Entre los privilegios que posee el propietario nuevo de la vista debe encontrarse como mínimo EXECUTE para la rutina dependiente.
– Imagine un activador con una dependencia sobre una tabla. Entre los privilegios que posee el propietario nuevo del activador debe encontrarse el mismo conjunto de privilegios para la tabla que indican las dependencias del activador. No es necesario el privilegio ALTER para la tabla en que se define el activador.

En la tabla siguiente se indican las vistas de catálogo del sistema que describen los objetos de que dependen otros objetos de base de datos.

Tabla 35. Vistas de catálogo que describen los objetos de que dependen otros objetos

Objeto de base de datos	Vista de catálogo del sistema
CONSTRAINT	SYSCAT.CONSTDEP
FUNCTION	SYSCAT.ROUTINEDEP; SYSCAT.ROUTINES (para una función con fuente)
INDEX	SYSCAT.INDEXDEP
INDEX EXTENSION	SYSCAT.INDEXEXTENSIONDEP
METHOD	SYSCAT.ROUTINEDEP
PACKAGE	SYSCAT.PACKAGEDEP
PROCEDURE	SYSCAT.ROUTINEDEP
TABLE	SYSCAT.TABDEP
TRIGGER	SYSCAT.TRIGDEP
VIEW	SYSCAT.TABDEP
XSROBJECT	SYSCAT.XSROBJECTDEP

Para transferir correctamente la propiedad de un objeto de base de datos que depende de otro objeto, el nuevo propietario del objeto de base de datos debe poseer unos privilegios determinados para el objeto dependiente de dicha dependencia:

– Si el objeto dependiente es una secuencia, el nuevo propietario debe poseer el privilegio USAGE para dicha secuencia.
– Si el objeto dependiente es una función, un método o un procedimiento, el nuevo propietario debe poseer el privilegio EXECUTE para dicha función, método o procedimiento.
– Si el objeto dependiente es un paquete, el nuevo propietario debe poseer el privilegio EXECUTE para dicho paquete.
– Si el objeto dependiente es un objeto XSR, el nuevo propietario debe poseer el privilegio USAGE para dicho objeto XSR.

Para cualquier otro objeto dependiente de una dependencia, utilice la columna TABAUTH de la vista de catálogo del sistema adecuada para determinar qué privilegios debe poseer el nuevo propietario.

- Si se intenta transferir la propiedad de un objeto a su propietario, se devuelve un aviso (SQLSTATE 01676).
- No se puede transferir la propiedad de los siguientes objetos de base de datos porque dichos objetos no tienen propietario: políticas de comprobación, agrupaciones de almacenamientos intermedios, funciones, etiquetas de seguridad, componentes de etiquetas de seguridad, políticas de seguridad, servidores, funciones de transformación, contextos fiables, correlaciones de usuarios y derivadores. Tenga en cuenta que no existe ninguna columna

OWNER en las vistas de catálogo SYSCAT.AUDITPOLICIES, SYSCAT.BUFFERPOOLS, SYSCAT.CONTEXTS, SYSCAT.ROLES, SYSCAT.SECURITYLABELS, SYSCAT.SECURITYLABELCOMPONENTS, SYSCAT.SECURITYPOLICIES, SYSCAT.SERVERS, SYSCAT.TRANSFORMS, SYSCAT.USEROPTIONS y SYSCAT.WRAPPERS.

- El nombre de esquema de un objeto cuya propiedad se ha transferido no cambia automáticamente.
- ***Compatibilidades***
  - Para garantizar la coherencia con otras sentencias de SQL:
    - NODEGROUP puede especificarse en lugar de DATABASE PARTITION GROUP

Ejemplos

Ejemplo 1: Transfiera la propiedad de la tabla T1 a PAUL.

```
TRANSFER OWNERSHIP OF TABLE WALID.T1
  TO USER PAUL PRESERVE PRIVILEGES
```

El valor de la columna OWNER de la tabla WALID.T1 de la vista de catálogo SYSCAT.TABLES se sustituye por 'PAUL'. Se otorgan a Paul implícitamente los privilegios siguientes para la tabla WALID.T1 (suponiendo que el propietario anterior de la tabla no perdió ninguno de los privilegios que tenía para ella): CONTROL y ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE, REFERENCE (WITH GRANT OPTION).

Ejemplo 2: Suponga que JOHN crea las tablas T1 y T2, y que MIKE tiene privilegio SELECT para las tablas JOHN.T1 y JOHN.T2. MIKE crea la vista V1 que depende de las tablas JOHN.T1 y JOHN.T2. Transfiera la propiedad de la vista V1 a HENRY, que tiene la autorización DBADM.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER HENRY PRESERVE PRIVILEGES
```

El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'HENRY'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'HENRY' y TABNAME = 'V1'.

Ejemplo 3: Suponga que HENRY, que tiene la autorización DBADM, crea un activador TR1 que depende de la tabla T1. Transfiera la propiedad del activador TR1 a WALID, que no tiene la autorización DBADM.

```
TRANSFER OWNERSHIP OF TRIGGER TR1
  TO USER WALID PRESERVE PRIVILEGES
```

La propiedad del activador se transfiere correctamente, aunque Walid no tiene la autorización DBADM.

Ejemplo 4: Suponga que JOHN crea las tablas T1 y T2, y que MIKE tiene privilegio SELECT para la tabla JOHN.T1 y privilegio CONTROL para la tabla JOHN.T2. PAUL tiene privilegio SELECT para las tablas JOHN.T1 y JOHN.T2. MIKE crea la vista V1 que depende de las tablas JOHN.T1 y JOHN.T2. La vista tiene una entrada para el privilegio SELECT en SYSCAT.TABAUTH y dos dependencias SELECT en SYSCAT.TABDEP para las tablas JOHN.T1 y JOHN.T2. Transfiera la propiedad de la vista V1 a PAUL, que es un usuario normal.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER PAUL PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, aunque Paul no tiene el privilegio CONTROL para la tabla JOHN.T2. Paul solo necesita el privilegio SELECT para las tablas JOHN.T1 y JOHN.T2 con objeto de mantener la existencia de la vista. (La vista solo tiene el privilegio SELECT ya que Paul no tenía el privilegio CONTROL para ambas tablas al crearse la vista, por lo que no se le otorgó el privilegio CONTROL para la vista.) El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'PAUL'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'PAUL'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'PAUL' y TABNAME = 'V1'.

Ejemplo 5: Suponga que JOHN crea la tabla T1 y que PUBLIC tiene privilegio SELECT para JOHN.T1. PAUL tiene privilegio SELECT para JOHN.T1 explícitamente y crea la vista V1 que depende de la tabla JOHN.T1. Transfiera la propiedad de la vista V1 a MIKE, que no es un usuario DBADM, pero tiene los privilegios necesarios para adquirir la propiedad a través del grupo especial PUBLIC.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER MIKE PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, ya que Mike tiene el privilegio SELECT para la tabla JOHN.T1 por medio de PUBLIC. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'MIKE'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'MIKE'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' y TABNAME = 'V1'.

Ejemplo 6: al igual que en el ejemplo 5, suponga que JOHN crea la tabla T1 y que la función R1 tiene el privilegio SELECT para JOHN.T1. PAUL tiene privilegio SELECT para JOHN.T1 explícitamente y crea la vista V1 que depende de la tabla JOHN.T1. Transfiera la propiedad de la vista V1 a MIKE, que no es un usuario DBADM pero tiene los privilegios necesarios por ser miembro en la función R1 para adquirir la propiedad de vista.

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER MIKE PRESERVE PRIVILEGES
```

La propiedad de la vista se transfiere correctamente, ya que Mike tiene el privilegio SELECT para la tabla JOHN.T1 por ser miembro en la función R1. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.VIEWS se sustituye por 'MIKE'. El valor de la columna OWNER de la vista V1 de la vista de catálogo SYSCAT.TABDEP se sustituye por 'MIKE'. Se añade una fila nueva a SYSCAT.TABAUTH con los valores siguientes: GRANTOR = 'SYSIBM', GRANTEE = 'MIKE' y TABNAME = 'V1'.

UPDATE

La sentencia UPDATE actualiza los valores de las columnas especificadas en las filas de una tabla, vista o apodo, o las tablas, apodos o vistas subyacentes de la selección completa especificada. La actualización de una fila de una vista actualiza una fila de su tabla base, si no se ha definido ningún activador INSTEAD OF para la operación de actualización en esta vista. Si se ha definido un activador de este tipo, en su lugar se ejecutará el activador. La actualización de una fila utilizando un apodo actualiza una fila del objeto de fuente de datos al que hace referencia el apodo.

Las formas de esta sentencia son:

- La forma UPDATE *Con búsqueda* se utiliza para actualizar una o varias filas (determinadas opcionalmente por la condición de búsqueda).
- La forma de UPDATE *Con posición* se utiliza para actualizar exactamente una fila (tal como determina la posición actual de un cursor).

Invocación

Una sentencia UPDATE puede incorporarse en un programa de aplicación o emitirse mediante la utilización de sentencias de SQL dinámico. Se trata de una sentencia ejecutable que puede prepararse de forma dinámica.

Autorización

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

- Privilegio UPDATE para la tabla, vista o apodo de destino
- Privilegio UPDATE para cada una de las columnas que deben actualizarse
- Privilegio CONTROL sobre la tabla, vista o apodo de destino
- Autorización SYSADM o DBADM

Si se incluye una *selecciónconpleta-fila* en la asignación, los privilegios del ID de autorización de la sentencia deben incluir, 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 se haga referencia en 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 precompila con normas SQL92 (opción LANGLEVEL con un valor de SQL92E o MIA) y la forma buscada de una sentencia UPDATE incluye una referencia a una columna de la tabla, vista o apodo en la parte derecha de la *cláusula-asignación*, o en cualquier parte de la *condición-búsqueda*, los privilegios del ID de autorización de la sentencia deben incluir además, como mínimo, uno de los siguientes:

- Privilegio SELECT
- Privilegio CONTROL
- Autorización SYSADM o DBADM

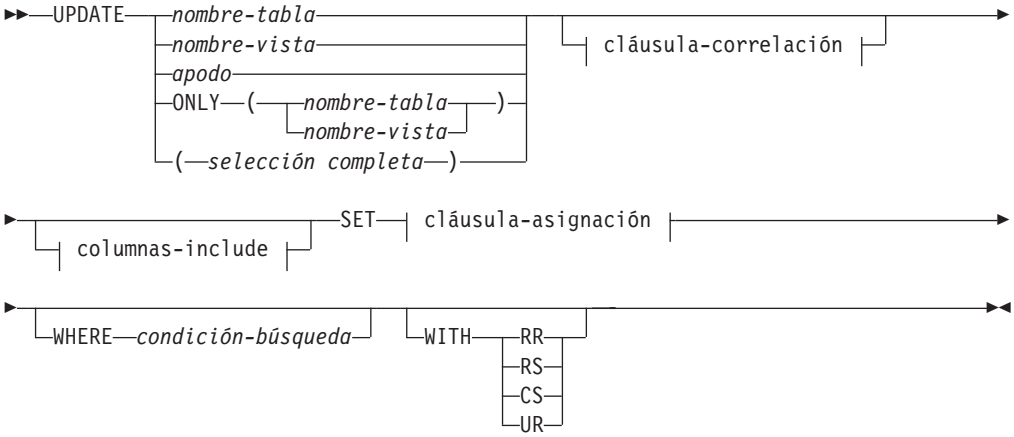
Si la tabla o vista especificada va precedida de la palabra clave ONLY, los privilegios del ID de autorización de la sentencia también deben incluir el privilegio SELECT para cada subtabla o subvista de la tabla o vista especificada.

No se comprueban los privilegios GROUP para las sentencias UPDATE estáticas.

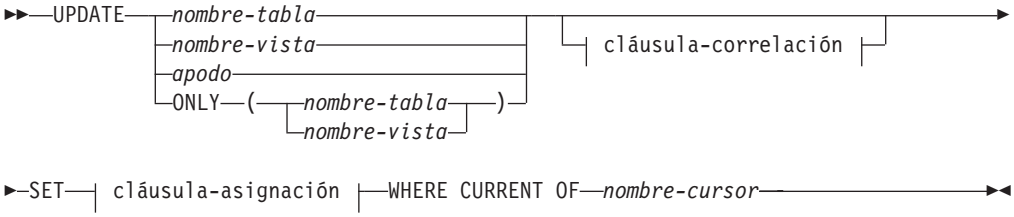
Si el destino de la operación de actualización es un apodo, los privilegios para el objeto en la fuente de datos no se toman en consideración hasta que la sentencia se ejecuta en la fuente de datos. En ese momento, el ID de autorización utilizado para conectarse a la fuente de datos debe disponer de los privilegios necesarios a fin de realizar la operación con el objeto en la fuente de datos. El ID de autorización de la sentencia puede correlacionarse con un ID de autorización distinto en la fuente de datos.

Sintaxis

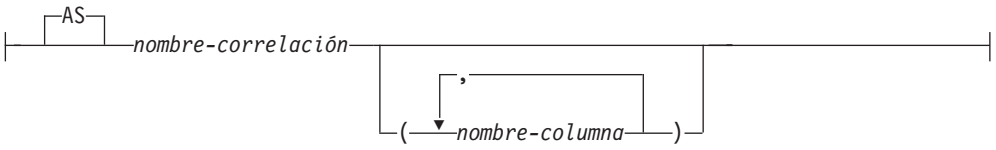
actualización-búsqueda:



actualización-posición:

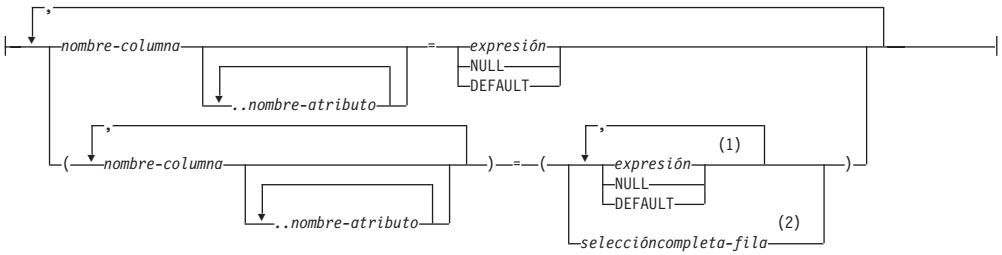


cláusula-correlación:



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 actualizables), una tabla de consultas materializadas mantenida por el sistema o una vista de sólo lectura que no tenga definido el 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 que se haya especificado o una columna INCLUDE. La columna de ID de objeto de una tabla con tipo no es actualizable (SQLSTATE 428DZ). Una columna no debe especificarse más de una vez, a no ser que vaya seguida de *..nombre-atributo* (SQLSTATE 42701).

Si especifica una columna INCLUDE, el nombre de columna no se puede cualificar.

Para una sentencia UPDATE con posición:

- Si se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor, cada nombre de columna de la *cláusula-asignación* también debe aparecer en la *cláusula-actualización*.

- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL MIA o SQL92E al compilarse previamente la aplicación, puede especificarse el nombre de cualquier columna que pueda actualizarse.
- Si no se ha especificado la *cláusula-actualización* en la *sentencia-select* del cursor y se ha especificado LANGLEVEL SAA1 explícitamente o por omisión al compilarse previamente la aplicación, no puede actualizarse ninguna columna.

..nombre-atributo
Especifica el atributo de un tipo estructurado que está definido (esto se denomina *asignación de atributos*). El *nombre-columna* especificado se debe definir con un tipo estructurado definido por el usuario (SQLSTATE 428DP). El nombre de atributo debe ser un atributo del tipo estructurado del *nombre-columna* (SQLSTATE 42703). Una asignación que no incluye la cláusula *..nombre-atributo* se denomina *asignación convencional*.

expresión
Indica el nuevo valor de la columna. La expresión es cualquier expresión del tipo que se describe en el apartado "Expresiones". La expresión no puede incluir una función 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).
- Si la columna se ha definido mediante la cláusula ROW CHANGE TIMESTAMP, el gestor de bases de datos genera el valor.

El único valor que se puede insertar en una columna generada que se ha definido con la cláusula GENERATED ALWAYS es DEFAULT (SQLSTATE 428C9).

La palabra clave DEFAULT no se puede utilizar como valor en una asignación de atributos (SQLSTATE 429B9).

La palabra clave DEFAULT no puede utilizarse como valor en una asignación para realizar una actualización en un apodo donde la fuente de datos no da soporte a la sintaxis DEFAULT.

selecciónconpleta-fila
Una selección completa que devuelve una sola fila con el número de columnas correspondiente al número de *nombres-columna* especificados para la asignación. Los valores se asignan a cada *nombre-columna* correspondiente. Si el resultado de la *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 especificado también debe indicarse en la cláusula FROM de la sentencia SELECT del cursor y la tabla de resultados del cursor no debe ser de sólo lectura. Para ver una explicación sobre las tablas de resultados de sólo lectura, consulte "DECLARE CURSOR".)

Cuando se ejecuta la sentencia UPDATE, el cursor debe posicionarse en una fila; dicha fila se actualiza.

Esta forma de UPDATE no se puede utilizar (SQLSTATE 42828) si el cursor hace referencia a:

- Una vista en la que se ha definido un activador INSTEAD OF UPDATE

- Una vista que incluye una función OLAP en la lista de la selección completa que define la vista
- Una vista que se ha definido, directa o indirectamente, utilizando la cláusula WITH ROW MOVEMENT

WITH
Especifica el nivel de aislamiento en el que se ejecuta la sentencia UPDATE.

RR
Lectura repetible

RS
Estabilidad de lectura

CS
Estabilidad del cursor

UR
Lectura no confirmada

El nivel de aislamiento por omisión de la sentencia es el nivel de aislamiento del paquete en el que está enlazada la sentencia. La cláusula WITH no afecta a los apodos, que siempre utilizan el nivel de aislamiento por omisión de la sentencia.

Normas

- ***Activadores:*** Las sentencias UPDATE pueden dar lugar a que se ejecuten activadores. Un activador puede dar lugar a que se ejecuten otras sentencias o a que se generen condiciones de error basadas en los valores de la actualización. Si una operación de actualización en una vista da lugar a que se ejecute un activador INSTEAD OF, se comprobarán la validez, la integridad referencial y las restricciones de las actualizaciones que se realizan en el activador y no las de la vista que ha dado lugar a la ejecución del activador o sus tablas subyacentes.
- ***Asignación***: Los valores de actualización se asignan a las columnas según las normas de asignación específicas.
- ***Validez***: La fila actualizada se debe ajustar a cualquier restricción impuesta en la tabla (o en la tabla base de la vista) por cualquier índice exclusivo de una columna actualizada.

  Si se utiliza una vista que no se ha definido utilizando WITH CHECK OPTION, se pueden cambiar las filas para que no se ajusten más a la definición de la vista. Dichas filas se actualizan en la tabla base de la vista y ya no aparecen más en la vista.

  Si se utiliza una vista que se ha definido utilizando WITH CHECK OPTION, una fila actualizada debe ajustarse a la definición de la vista. Para obtener información acerca de las normas que rigen esta situación, consulte "CREATE VIEW".
- ***Restricción de comprobación***: El valor de actualización debe satisfacer las condiciones de comprobación de las restricciones de comprobación definidas en la tabla.

  En UPDATE para una tabla con restricciones de comprobación definidas se evalúan las condiciones de restricción para cada columna una vez para cada fila que se actualiza. Cuando se procesa una sentencia UPDATE, sólo se comprueban las restricciones de comprobación que hacen referencia a las columnas actualizadas.
- ***Integridad 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 exclusivas padres pueden actualizarse siempre que cada hijo tenga una clave padre en el momento en que se completa la sentencia de actualización. Un valor de actualización no nulo para una clave foránea debe ser igual a un valor de la clave primaria de la tabla padre de la relación.

- ***Valores XML:*** Cuando se actualiza un valor de la columna XML, el nuevo valor debe ser un documento XML con el formato correcto (SQLSTATE 2200M).
- ***Política de seguridad:*** si se protege la tabla identificada o la tabla base de la vista identificada con una política de seguridad, el ID de autorización de la sesión debe tener las credenciales de control de acceso basado en etiquetas (LBAC) que permiten:
  - Acceso de grabación a todas las columnas protegidas que se están actualizando (SQLSTATE 42512)
  - Acceso de grabación para cualquier valor explícito proporcionado para una columna DB2SECURITYLABEL para las políticas de seguridad que se han creado con la opción RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL (SQLSTATE 23523)
  - Acceso de grabación y de lectura a todas las filas que se están actualizando (SQLSTATE 42519)

  El ID de autorización de la sesión también debe tener otorgada una etiqueta de seguridad para acceso de grabación para la política de seguridad si se utiliza un valor implícito para una columna DB2SECURITYLABEL (SQLSTATE 23523), lo cual puede suceder cuando:
  - La columna DB2SECURITYLABEL no está incluida en la lista de columnas que se van a actualizar (y, por lo tanto, se actualizará implícitamente en la etiqueta de seguridad para el acceso de grabación del ID de autorización de la sesión)
  - Se proporciona explícitamente un valor para la columna DB2SECURITYLABEL pero el ID de autorización de la sesión no tiene acceso de grabación para dicho valor y la política de seguridad se crea con la opción OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

Notas

- Si un valor de actualización viola cualquier restricción o si se produce cualquier otro error durante la ejecución de la sentencia UPDATE, no se actualiza ninguna fila. El orden en que se actualizan múltiples filas no está definido.
- Una actualización de una vista definida utilizando la cláusula WITH ROW MOVEMENT puede ocasionar una operación de supresión y una operación de inserción contra las tablas subyacentes de la vista. Para ver detalles, consulte la descripción de la sentencia CREATE VIEW.
- Cuando una sentencia UPDATE completa su ejecución, el valor de SQLERRD(3) de la SQLCA es el número de filas que se han calificado para la operación de actualización. En el contexto de una sentencia de procedimiento de SQL, el valor puede recuperarse utilizando la variable ROW_COUNT de la sentencia GET DIAGNOSTICS. El campo SQLERRD(5) contiene el número de filas insertadas, suprimidas o actualizadas por todos los activadores activados.
- A menos que ya existan los bloqueos adecuados, se adquieren uno o varios bloqueos por la ejecución de una sentencia UPDATE satisfactoria. Hasta que se liberan los bloqueos, la fila actualizada sólo puede accederse por el proceso de aplicación que ha realizado la actualización (excepto las aplicaciones que utilizan el nivel de aislamiento de Lectura no confirmada). Para obtener más información sobre el bloqueo, consulte las descripciones de las sentencias COMMIT, ROLLBACK y LOCK TABLE.

- Cuando se actualicen las estadísticas de distribución de columna para una tabla con tipo, debe especificarse la subtabla que haya 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".
- Con las tablas particionadas, la operación *nombre-cursor* UPDATE WHERE CURRENT OF puede mover una fila de una partición de datos a otra. Después de este proceso, el cursor ya no se encuentra sobre la fila y ya no se pueden realizar más modificaciones de *nombre-cursor* UPDATE WHERE CURRENT OF para dicha fila. No obstante, puede captarse la siguiente fila del cursor.
- Para una columna definida mediante la cláusula ROW CHANGE TIMESTAMP, el valor siempre se cambia al actualizar la fila. Si la columna no se especifica en la lista SET explícitamente, el gestor de bases de datos todavía genera un valor para dicha fila. El valor es exclusivo para cada partición de tabla de la partición de base de datos y se establece en la indicación de fecha y hora aproximada correspondiente a la actualización de fila.

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 EMPNO, SALARY, COMM,
  AVG(SALARY) OVER (PARTITION BY WORKDEPT),
  AVG(COMM) OVER (PARTITION BY WORKDEPT)
  FROM EMPLOYEE E) AS E(EMPNO, SALARY, COMM, AVGSAL, AVGCOMM)
SET (SALARY, COMM) = (AVGSAL, AVGCOMM)
WHERE EMPNO = '000120'
```

  La sentencia anterior es semánticamente equivalente a la sentencia siguiente, pero sólo necesita un acceso a la tabla EMPLOYEE, mientras que la sentencia siguiente especifica la tabla EMPLOYEE dos veces.

```
UPDATE EMPLOYEE EU
  SET (EU.SALARY, EU.COMM)
  =
(SELECT AVG(ES.SALARY), AVG(ES.COMM)
  FROM EMPLOYEE ES
  WHERE ES.WORKDEPT = EU.WORKDEPT)
  WHERE EU.EMPNO = '000120'
```

- *Ejemplo 5:* En un programa C visualice las filas de la tabla EMPLOYEE y, después, si se le pide hacerlo, cambie el trabajo (JOB) de algunos empleados por el nuevo trabajo escrito.

```
EXEC SQL  DECLARE C1 CURSOR FOR
               SELECT *
                 FROM EMPLOYEE
                 FOR UPDATE OF JOB;

EXEC SQL  OPEN C1;

EXEC SQL  FETCH C1 INTO ...     ;
if ( strcmp (change, "YES") == 0 )
  EXEC SQL  UPDATE EMPLOYEE
              SET JOB = :newjob
              WHERE CURRENT OF C1;

EXEC SQL  CLOSE C1;
```

- *Ejemplo 6:* Mutación de atributos de objetos de columnas.

  Sean los siguientes tipos y tablas:

```
CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
  NOT FINAL WITHOUT COMPARISONS
  MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE
```

  El ejemplo siguiente actualiza la tabla CIRCLES cambiando la columna OWNER y el atributo RADIUS de la columna CIRCLE cuyo ID es 999:

```
UPDATE CIRCLES
  SET OWNER = 'Bruce'
    C..RADIUS = 5
  WHERE ID = 999
```

El ejemplo siguiente traslada las coordenadas X e Y del centro del círculo identificado por 999:

```
UPDATE CIRCLES
  SET C..CENTER..X = C..CENTER..Y,
    C..CENTER..Y = C..CENTER..X
  WHERE ID = 999
```

El ejemplo siguiente constituye otro modo de escribir las dos sentencias anteriores. Este ejemplo combina los efectos de ambos ejemplos anteriores:

```
UPDATE CIRCLES
  SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
    ('Bruce',5,C..CENTER..Y,C..CENTER..X)
  WHERE ID = 999
```

- *Ejemplo 7:* Actualice la columna XMLDOC de la tabla DOCUMENTS con DOCID '001' a la serie de caracteres seleccionada y analizada a partir de la tabla XMLTEXT.

```
UPDATE DOCUMENTS SET XMLDOC =
  (SELECT XMLPARSE(DOCUMENT C1 STRIP WHITESPACE)
    FROM XMLTEXT WHERE TEXTID = '001')
WHERE DOCID = '001'
```

VALUES INTO

La sentencia VALUES INTO produce una tabla de resultados que consta de una fila como máximo y asigna los valores de dicha fila a las variables del lenguaje principal.

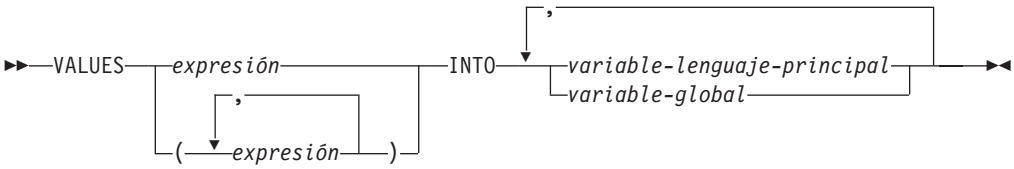
Invocación

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 o variables globales.

variable-lenguaje-principal
Identifica una variable que está descrita en el programa bajo las normas para la declaración de variables del lenguaje principal.

variable-global
Identifica una variable global que existe en el servidor actual.

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 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 ninguna variable.

Normas

- Las variables globales no pueden ser asignadas en activadores, funciones, métodos o sentencias compuestas dinámicas o en un procedimiento al que uno de estos objetos llame de forma directa o indirecta (SQLSTATE 428GX)

Ejemplos

Ejemplo 1: Este ejemplo C recupera el valor del registro especial CURRENT PATH en una variable del lenguaje principal.

```
EXEC SQL VALUES(CURRENT PATH)
         INTO :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;
```

Ejemplo 3: este ejemplo C recupera el valor del registro especial SESSIÓN_USER en una variable global.

```
EXEC SQL VALUES(SESSION_USER)
  INTO GV_SESS_USER;
```

VALUES

La sentencia VALUES es una forma de consulta. Puede incorporarse en un programa de aplicación o emitirse interactivamente.

WHENEVER

La sentencia WHENEVER especifica la acción que se debe tomar cuando se produce una condición de excepción especificada.

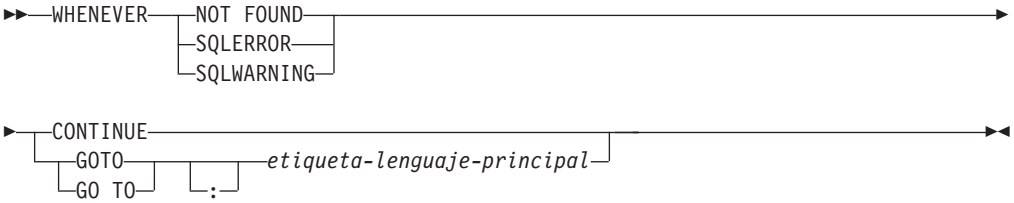
Invocación

Esta sentencia sólo puede incorporarse en un programa de aplicación. No es una sentencia ejecutable. La sentencia no se soporta en REXX.

Autorización

No se necesita.

Sintaxis



Descripción

Las cláusulas NOT FOUND, SQLERROR o SQLWARNING se utilizan para identificar el tipo de condición de excepción.

NOT FOUND
Identifica cualquier condición que dé como resultado un SQLCODE de +100 o un SQLSTATE de '02000'.

SQLERROR
Identifica cualquier condición que dé como resultado un SQLCODE negativo.

SQLWARNING
Identifica cualquier condición que dé como resultado una condición de aviso (SQLWARN0 es 'W'), o que dé como resultado un código de retorno SQL positivo que no sea +100.

Las cláusulas CONTINUE o GO TO se utilizan para especificar lo que ocurre cuando existe el tipo de condición de excepción identificado.

CONTINUE
Provoca que se ejecute la siguiente instrucción secuencial del programa fuente.

GOTO o GO TO *etiqueta-lenguaje-principal*
Provoca que el control pase a la sentencia identificada por la etiqueta-lenguaje-principal. Para etiqueta-lenguaje-principal, utilice un símbolo individual, precedido opcionalmente por dos puntos. El formato del símbolo depende del lenguaje principal.

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;
```

WHILE

La sentencia WHILE repite la ejecución de una sentencia o grupo de sentencias mientras sea verdadera una condición especificada.

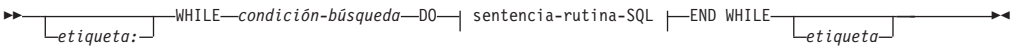
Invocación

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.

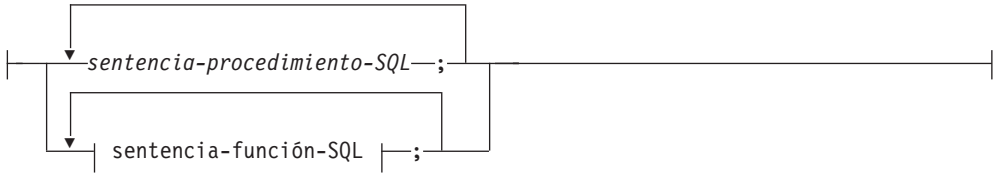
Autorización

No se requieren privilegios para invocar la sentencia WHILE. Sin embargo, el ID de autorización de la sentencia debe mantener los privilegios necesarios para invocar las sentencias de SQL y la condición de búsqueda incorporadas en la sentencia WHILE.

Sintaxis



sentencia-rutina-SQL:



Descripción

etiqueta
Especifica la etiqueta de la sentencia WHILE. Si se especifica la etiqueta inicial, esa etiqueta puede especificarse en sentencias LEAVE e ITERATE. Si se especifica una etiqueta final, ésta deberá ser igual que la etiqueta inicial.

condición-búsqueda
Especifica una condición que se evalúa antes de cada ejecución del bucle. Si la condición es verdadera, se procesan las sentencias de procedimiento de SQL incluidas en el bucle.

sentencia-procedimiento-SQL
Especifica las sentencias de SQL que se deben ejecutar en el bucle. La *sentencia-procedimiento-SQL* sólo se puede aplicar en el contexto de un procedimiento de SQL. Consulte *sentencia-procedimiento-SQL* en "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 "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
```

Apéndice A. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- *Centro de información de DB2*
  - Temas (Tareas, concepto y temas de consulta)
  - Ayuda para herramientas de DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (desde el DVD con PDF de DB2)
  - Manuales en copia impresa
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes

Nota: Los temas del *Centro de información de DB2* se actualizan con más frecuencia que los manuales en PDF o impresos. Para obtener la información más actualizada, instale las actualizaciones de la documentación cuando estén disponibles, o consulte el *Centro de información de DB2* en ibm.com.

Puede acceder a información técnica adicional de DB2 como, por ejemplo, notas técnicas, documentos técnicos y publicaciones IBM Redbooks en línea, en el sitio ibm.com. Acceda al sitio de la biblioteca de software de gestión de información de DB2 en http://www.ibm.com/software/data/sw-library/.

Comentarios sobre la documentación

Agradecemos los comentarios sobre la documentación de DB2. Si tiene sugerencias sobre cómo podemos mejorar la documentación de DB2, envíe un correo electrónico a db2docs@ca.ibm.com. El personal encargado de la documentación de DB2 lee todos los comentarios de los usuarios, pero no puede responderlos directamente. Proporcione ejemplos específicos siempre que sea posible de manera que podamos comprender mejor sus problemas. Si realiza comentarios sobre un tema o archivo de ayuda determinado, incluya el título del tema y el URL.

No utilice esta dirección de correo electrónico para contactar con el Soporte al cliente de DB2. Si tiene un problema técnico de DB2 que no está tratado por la documentación, consulte al centro local de servicio técnico de IBM para obtener ayuda.

Si desea ayudar a IBM para que los productos IBM Information Management sean más fáciles de utilizar, obtenga el cuestionario de consumibilidad: http://www.ibm.com/software/data/info/consumability-survey/.

© Copyright IBM Corp. 1993, 2010

Biblioteca técnica de DB2 en copia impresa o en formato PDF

Las tablas siguientes describen la biblioteca de DB2 que está disponible en el Centro de publicaciones de IBM en www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss. Los manuales de DB2 Versión 9.5 en inglés y las versiones traducidas en formato PDF se pueden descargar de www.ibm.com/support/docview.wss?uid=swg27009727 y www.ibm.com/support/docview.wss?uid=swg27009728 respectivamente.

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

El número de documento se incrementa cada vez que se actualiza un manual. Asegúrese de que lee la versión más reciente de los manuales, tal como aparece en la tabla siguiente:

Nota: El *Centro de información de DB2* se actualiza con más frecuencia que los manuales en PDF o impresos.

Tabla 36. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
Consulta de las API administrativas	SC11-3505-03	Sí	Diciembre de 2010
Rutinas y vistas administrativas	SC11-3507-03	No	Diciembre de 2010
Call Level Interface Guide and Reference, Volume 1	SC23-5844-03	Sí	Diciembre de 2010
Call Level Interface Guide and Reference, Volume 2	SC23-5845-03	Sí	Diciembre de 2010
Consulta de mandatos	SC11-3506-03	Sí	Diciembre de 2010
Data Movement Utilities Guide and Reference	SC23-5847-03	Sí	Diciembre de 2010
Data Recovery and High Availability Guide and Reference	SC23-5848-03	Sí	Diciembre de 2010
Data Servers, Databases, and Database Objects Guide	SC23-5849-03	Sí	Diciembre de 2010
Database Security Guide	SC23-5850-03	Sí	Diciembre de 2010
Developing ADO.NET and OLE DB Applications	SC11-3499-02	Sí	Abril de 2009
Developing Embedded SQL Applications	SC11-3500-02	Sí	Abril de 2009
Desarrollo de aplicaciones Java	SC11-3501-03	Sí	Diciembre de 2010
Desarrollo de aplicaciones Perl y PHP	SC11-3502-02	No	Abril de 2009

Tabla 36. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Desarrollo de rutinas definidas por el usuario (SQL y externas)	SC11-3503-03	Sí	Diciembre de 2010
Iniciación al desarrollo de aplicaciones de bases de datos	GC11-3504-03	Sí	Diciembre de 2010
Iniciación a la instalación y administración de DB2 en Linux y Windows	GC11-3511-03	Sí	Diciembre de 2010
Internationalization Guide	SC23-5858-02	Sí	Abril de 2009
Consulta de mensajes, Volumen 1	GI11-7823-00	No	
Consulta de mensajes, Volumen 2	GI11-7824-00	No	
Guía de migración	GC11-3509-03	Sí	Diciembre de 2010
Net Search Extender Guía de administración y del usuario	SC11-3615-02	Sí	Abril de 2009
Partitioning and Clustering Guide	SC23-5860-03	Sí	Diciembre de 2010
Query Patroller Administration and User's Guide	SC23-8507-01	Sí	Abril de 2009
Guía rápida para clientes de IBM Data Server	GC11-3513-03	No	Diciembre de 2010
Guía rápida para servidores DB2	GC11-3512-03	Sí	Diciembre de 2010
Spatial Extender and Geodetic Data Management Feature Guía del usuario y manual de consulta	SC11-3614-02	Sí	Abril de 2009
Consulta de SQL, Volumen 1	SC11-3509-03	Sí	Diciembre de 2010
Consulta de SQL, Volumen 2	SC11-3508-03	Sí	Diciembre de 2010
System Monitor Guide and Reference	SC23-5865-03	Sí	Diciembre de 2010
Guía de Text Search	SC11-3717-03	Sí	Diciembre de 2010
Troubleshooting Guide	GI11-7857-03	No	Diciembre de 2010
Tuning Database Performance	SC23-5867-03	Sí	Diciembre de 2010
Guía de aprendizaje de Visual Explain	SC11-3518-00	No	
Novedades	SC11-3517-03	Sí	Diciembre de 2010

Tabla 36. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Workload Manager Guide and Reference	SC23-5870-03	Sí	Diciembre de 2010
pureXML Guide	SC23-5871-03	Sí	Diciembre de 2010
XQuery Reference	SC23-5872-02	No	Abril de 2009

Tabla 37. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
Guía rápida para DB2 Connect Personal Edition	GC11-3515-03	Sí	Diciembre de 2010
Guía rápida para servidores DB2 Connect	GC11-3516-03	Sí	Diciembre de 2010
Guía del usuario de DB2 Connect	SC11-3514-03	Sí	Diciembre de 2010

Tabla 38. Información técnica de Information Integration

Nombre	Número de documento	Copia impresa disponible	Última actualización
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	Sí	
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Sí	
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-01	No	
Information Integration: SQL Replication Guide and Reference	SC19-1030-01	Sí	
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-01	Sí	

Pedido de manuales de DB2 en copia impresa

Si necesita manuales de DB2 en copia impresa, puede comprarlos en línea en varios países o regiones, pero no en todos. Siempre puede hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM. Recuerde que algunas publicaciones en copia software del DVD *Documentación en PDF de DB2* no están disponibles en copia impresa. Por ejemplo, no está disponible la publicación *Consulta de mensajes de DB2* en copia impresa.

Las versiones impresas de muchas de las publicaciones de DB2 disponibles en el DVD de Documentación en PDF de DB2 se pueden solicitar a IBM por una

cantidad. Dependiendo desde dónde realice el pedido, podrá solicitar manuales en línea, desde el Centro de publicaciones de IBM. Si la realización de pedidos en línea no está disponible en su país o región, siempre puede hacer pedidos de manuales de DB2 en copia impresa al representante local de IBM. Tenga en cuenta que no todas las publicaciones del DVD de Documentación en PDF de DB2 están disponibles en copia impresa.

Nota: La documentación más actualizada y completa de DB2 se conserva en el Centro de información de DB2 en http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para hacer pedidos de manuales de DB2 en copia impresa:

- Para averiguar si puede hacer pedidos de manuales de DB2 en copia impresa en línea en su país o región, consulte el Centro de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Debe seleccionar un país, región o idioma para poder acceder a la información sobre pedidos de publicaciones y, a continuación, seguir las instrucciones sobre pedidos para su localidad.
- Para hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM:
  1. Localice la información de contacto de su representante local desde uno de los siguientes sitios Web:
     - El directorio de IBM de contactos en todo el mundo en el sitio www.ibm.com/planetwide
     - El sitio Web de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Tendrá que seleccionar su país, región o idioma para acceder a la página de presentación de las publicaciones apropiadas para su localidad. Desde esta página, siga el enlace "Acerca de este sitio".
  2. Cuando llame, indique que desea hacer un pedido de una publicación de DB2.
  3. Proporcione al representante los títulos y números de documento de las publicaciones que desee solicitar. Si desea consultar los títulos y los números de documento, consulte el apartado “Biblioteca técnica de DB2 en copia impresa o en formato PDF” en la página 1068.

Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

DB2 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.

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.

Acceso a diferentes versiones del Centro de información de DB2

Para los temas de DB2 Versión 9.8, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Para los temas de la versión 9.7 de DB2, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Para los temas de DB2 Versión 9.5, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para los temas de DB2 Versión 9.1, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Para los temas de DB2 Versión 8, vaya al URL del *Centro de información de DB2* en: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Visualización de temas en su idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias del navegador. Si un tema no se ha traducido al idioma preferido, el Centro de información de DB2 visualiza dicho tema en inglés.

- Para visualizar temas en su idioma preferido en el navegador Internet Explorer:
  1. En Internet Explorer, pulse en el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...**. Se abrirá la ventana Preferencias de idioma.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**.

       Nota: La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.
- Para visualizar temas en su idioma preferido en un navegador Firefox o Mozilla:
  1. Seleccione el botón en la sección **Idiomas** del diálogo **Herramientas** —> **Opciones** —> **Avanzado**. Se visualizará el panel Idiomas en la ventana Preferencias.
  2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
     - Para añadir un nuevo idioma a la lista, pulse el botón **Añadir...** a fin de seleccionar un idioma en la ventana Añadir idiomas.
     - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
  3. Borre la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.

En algunas combinaciones de navegador y sistema operativo, puede que también tenga que cambiar los valores regionales del sistema operativo al entorno local y al idioma de su elección.

Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Si ha instalado localmente el Centro de información de DB2, puede obtener las actualizaciones de la documentación de IBM e instalarlas.

Para actualizar el *Centro de información de DB2* instalado localmente es preciso que:

1. Detenga el *Centro de información de DB2* en el sistema, y reinicie el Centro de información en modalidad autónoma. La ejecución del Centro de información en modalidad autónoma impide que otros usuarios de la red accedan al Centro de información y permite al usuario aplicar las actualizaciones. Los Centros de información no administrativos y no root de *DB2* se ejecutan siempre en modalidad autónoma.
2. Utilice la característica Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que desee instalar, puede utilizar la característica Actualizar para obtenerlos e instalarlos.

   Nota: Si su entorno requiere la instalación de actualizaciones del *Centro de información de DB2* en una máquina no conectada a Internet, debe duplicar el sitio de actualizaciones en un sistema de archivos local utilizando una máquina que esté conectada a Internet y tenga instalado el *Centro de información de DB2*. Si muchos usuarios en la red van a instalar las actualizaciones de la documentación, puede reducir el tiempo necesario para realizar las actualizaciones duplicando también el sitio de actualizaciones localmente y creando un proxy para el sitio de actualizaciones.
   Si hay paquetes de actualización disponibles, utilice la característica Actualizar para obtener los paquetes. Sin embargo, la característica Actualizar sólo está disponible en modalidad autónoma.
3. Detenga el Centro de información autónomo y reinicie el *Centro de información de DB2* en su equipo.

Nota: En Windows Vista, los mandatos listados más abajo se deben ejecutar como administrador. Para iniciar un indicador de mandatos o una herramienta gráfica con privilegios de administrador completos, pulse con el botón derecho del ratón el atajo y, a continuación, seleccione **Ejecutar como administrador**.

Para actualizar el Centro de información de *Centro de información de DB2* instalado en el sistema o en el servidor de Intranet:

1. Detenga el *Centro de información de DB2*.
   - En Windows, pulse **Inicio** → **Panel de control** → **Herramientas administrativas** → **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Detener**.
   - En Linux, especifique el mandato siguiente:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Inicie el Centro de información en modalidad autónoma.
   - En Windows:
     a. Abra una ventana de mandatos.
     b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. De manera predeterminada, el *Centro de información de DB2*

está instalado en el directorio *Archivos_de_programa*\IBM\DB2 Information Center\Version 9.5, donde *Archivos_de_programa* representa la ubicación del directorio Archivos de programa.

c. Navegue desde el directorio de instalación al directorio doc\bin.

d. Ejecute el archivo help_start.bat:

```
help_start.bat
```

- En Linux:

a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. De forma predeterminada, el *Centro de información de DB2* se instala en el directorio /opt/ibm/db2ic/V9.5.

b. Navegue desde el directorio de instalación al directorio doc/bin.

c. Ejecute el script help_start:

```
help_start
```

Se inicia el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.

3. Pulse en el botón **Actualizar** ( ). En la derecha del panel del Centro de información, pulse en **Buscar actualizaciones**. Se visualiza una lista de actualizaciones para la documentación existente.
4. Para iniciar el proceso de instalación, compruebe las selecciones que desee instalar y, a continuación, pulse **Instalar actualizaciones**.
5. Cuando finalice el proceso de instalación, pulse **Finalizar**.
6. Detenga el Centro de información autónomo:
   - En Windows, navegue hasta el directorio doc\bin del directorio de instalación y ejecute el archivo help_end.bat:

     ```
     help_end.bat
     ```

     Nota: El archivo help_end de proceso por lotes contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el archivo help_start de proceso por lotes. No utilice Control-C ni ningún otro método para concluir help_start.bat.
   - En Linux, navegue hasta el directorio de instalación doc/bin y ejecute el script help_end:

     ```
     help_end
     ```

     Nota: El script help_end contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el script help_start. No utilice ningún otro método para concluir el script help_start.
7. Reinicie el *Centro de información de DB2*:
   - En Windows, pulse **Inicio** → **Panel de control** → **Herramientas administrativas** → **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Iniciar**.
   - En Linux, especifique el mandato siguiente:

     ```
     /etc/init.d/db2icdv95 start
     ```

El *Centro de información de DB2* actualizado visualiza los temas nuevos y actualizados.

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos DB2. Se proporcionan instrucciones paso a paso a través de lecciones.

Antes de comenzar

Puede ver la versión XHTML de la guía de aprendizaje desde el Centro de información en el sitio http://publib.boulder.ibm.com/infocenter/db2help/.

Algunas lecciones utilizan datos o código de ejemplo. Consulte la guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

Guías de aprendizaje de DB2

Para ver la guía de aprendizaje, pulse el título.

"pureXML" en *pureXML Guide*
Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

"Visual Explain" en *Guía de aprendizaje de Visual Explain*
Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución y determinación de problemas para ayudarle en la utilización de productos de base de datos DB2.

Documentación de DB2
Puede encontrar información sobre la resolución de problemas en la publicación DB2 Troubleshooting Guide o en la sección Conceptos fundamentales sobre bases de datos del Centro de información de DB2. En ellas encontrará información sobre cómo aislar e identificar problemas utilizando herramientas y programas de utilidad de diagnóstico de DB2, soluciones a algunos de los problemas más habituales y otros consejos sobre cómo solucionar problemas que podría encontrar en los productos DB2.

Sitio web de soporte técnico de DB2
Consulte el sitio Web de soporte técnico de DB2 si tiene problemas y desea obtener ayuda para encontrar las causas y soluciones posibles. El sitio de soporte técnico tiene enlaces a las publicaciones más recientes de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR o arreglos de defectos), fixpacks y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al sitio Web de soporte técnico de DB2 en la dirección http://www.ibm.com/software/data/db2/support/db2_9/

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Uso personal: Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se mantengan los avisos sobre la propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

Uso comercial: Puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando se mantengan todos los avisos sobre la propiedad. No puede realizar trabajos derivativos de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Excepto lo expresamente concedido en este permiso, no se conceden otros permisos, licencias ni derechos, explícitos o implícitos, sobre las publicaciones ni sobre ninguna información, datos, software u otra propiedad intelectual contenida en el mismo.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando, a su discreción, el uso de las publicaciones sea en detrimento de su interés o cuando, según determine IBM, las instrucciones anteriores no se cumplan correctamente.

No puede descargar, exportar ni volver a exportar esta información excepto en el caso de cumplimiento total con todas las leyes y regulaciones vigentes, incluyendo todas las leyes y regulaciones sobre exportación de los Estados Unidos.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN "TAL CUAL" Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

Apéndice B. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América

Es posible que IBM no comercialice en otros países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias referentes a información de 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
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y cambios en los productos y programas descritos en esta publicación.

© Copyright IBM Corp. 1993, 2010

Este documento puede proporcionar enlaces o referencias a sitios y recursos que no son de IBM. IBM no representa, no da garantías, ni se compromete con los recursos de terceros ni con los recursos que no son de IBM a los cuales se puede hacer referencia, acceder desde o enlazarse con desde este documento. Un enlace a un sitio que no es de IBM no implica que IBM apruebe el contenido o la utilización de dicho sitio Web o a su propietario. Además, IBM no forma parte ni es responsable de ninguna transacción que el usuario pueda realizar con terceros, aún cuando llegue a conocerlos (o utilice un enlace a ellos) desde un sitio de IBM. De acuerdo a esto, el usuario reconoce y acepta que IBM no es responsable de la disponibilidad de dichos recursos o sitios externos ni tampoco es responsable de ningún contenido, servicio, producto u otros materiales que estén o se encuentren disponibles desde dichos sitios o recursos. Cualquier software que proporcionen terceras partes, estarán sujetos a los términos y condiciones de licencia que acompañen al software.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Este manual puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Este manual 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

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Puede consultarse en línea una lista actualizada de las marcas registradas de IBM en la sección Copyright and trademark information de la web www.ibm.com/legal/copytrade.shtml.

Los siguientes términos son marcas registradas de otras empresas.

- Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.
- Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.
- UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.
- Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y/o en otros países.
- Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

A

acceso remoto
  conexiones no satisfactorias 249
  conexiones satisfactorias 249
  sentencia CONNECT
    EXCLUSIVE MODE, conexión dedicada 249
    ON SINGLE DBPARTITIONNUM, conexión dedicada 249
    SHARE MODE, de sólo lectura para ningún conector 249
    sólo información de servidor, sin operandos 249
activadores
  actualizaciones
    UPDATE, sentencia 1048
  adición de comentarios al catálogo 217
  CREATE TRIGGER, sentencia 586
  descartar 723
  INSERT, sentencia 839
  mensajes de error 586
  no operativos 586
  tablas con tipo 586
actualización posicional de columnas por fila 1048
actualizaciones
  Centro de información de DB2 1073
  vistas actualizables 651
agrupaciones de almacenamientos intermedios 271
  establecimiento de tamaño 29, 271
  suprimir utilizando la sentencia DROP 723
  tamaño de página 271
alias
  adición de comentarios al catálogo 217
  sentencia CREATE ALIAS 264
  suprimir utilizando la sentencia DROP 723
ALIAS, cláusula
  COMMENT, sentencia 217
  DROP, sentencia 723
ALL PRIVILEGES, cláusula
  REVOKE privilegios de tabla, vista o apodo 931
  sentencia GRANT (tabla, vista o apodo) 825
ALLOCATE CURSOR, sentencia
  descripción 23
almacenamiento
  restitución, unidad de trabajo, ROLLBACK 940
ALTER, cláusula
  sentencia GRANT (tabla, vista o apodo) 825
  sentencia REVOKE, eliminar privilegio 931
ALTER AUDIT POLICY, sentencia 25
ALTER BUFFERPOOL, sentencia 29
ALTER DATABASE, sentencia 36
ALTER DATABASE PARTITION GROUP, sentencia 32
ALTER FUNCTION, sentencia 40
ALTER HISTOGRAM TEMPLATE, sentencia 43
ALTER METHOD, sentencia 45
ALTER NICKNAME, sentencia
  descripción 47
ALTER NODEGROUP, sentencia
  ver ALTER DATABASE PARTITION GROUP 32
ALTER PROCEDURE (con fuente), sentencia 59
ALTER PROCEDURE (externo), sentencia 56
ALTER PROCEDURE (SQL), sentencia 61
ALTER SECURITY LABEL COMPONENT, sentencia 63
ALTER SECURITY POLICY, sentencia 66
ALTER SEQUENCE, sentencia 70
ALTER SERVER, sentencia 74
ALTER SERVICE CLASS, sentencia 78
ALTER TABLE, sentencia
  autorización necesaria 85
  diagrama de sintaxis 85
  ejemplos 85
ALTER TABLESPACE, sentencia
  descripción 133
ALTER THRESHOLD, sentencia 144
ALTER TRUSTED CONTEXT, sentencia 151
ALTER TYPE (estructurado), sentencia 160
ALTER USER MAPPING, sentencia 167
ALTER VIEW, sentencia
  autorización 169
  descripción 169
  diagrama de sintaxis 169
ALTER WORK ACTION SET, sentencia 171
ALTER WORK CLASS SET, sentencia 183
ALTER WORKLOAD, sentencia 188
ALTER WRAPPER, sentencia 195
ALTER XSROBJECT, sentencia 197
ámbito
  adición con sentencia ALTER TABLE 85
  adición con sentencia ALTER VIEW 169
  CREATE VIEW, sentencia 651
  definición con columna añadida 85
  definición con la sentencia CREATE TABLE 484
anotaciones cronológicas
  creación de una tabla sin registro de anotaciones inicial 484
antememoria
  EXECUTE, sentencia 760
aplicaciones de SQL incorporado
  EXECUTE IMMEDIATE, sentencia 757
  sentencias de formato de serie de caracteres 757
  visión general 11
apodos
  descripción 416
  privilegios
    otorgar 825
    otorgar control 825
    revocar 931
aritmética
  marcadores de parámetro 872
AS, cláusula
  CREATE VIEW, sentencia 651
ASC, cláusula
  CREATE INDEX, sentencia 385
ASSOCIATE LOCATORS, sentencia 198
ASUTIME
  en sentencia CREATE FUNCTION (escalar externa) 300
  en sentencia CREATE FUNCTION (tabla externa) 326
  en sentencia CREATE PROCEDURE 430, 450
AUDIT, sentencia 200
autorización de administrador de seguridad (SECADM)
  GRANT (autorizaciones de bases de datos), sentencia 789
  revocar 898
autorizaciones
  control público en índices 798

© Copyright IBM Corp. 1993, 2010

autorizaciones *(continuación)*
crear público en esquema 810
otorgar control sobre índices 798
otorgar el control en operaciones de bases de datos 789
otorgar para crear en esquema 810
revocar 898
avisos 1077
ayuda
idioma de configuración 1072
sentencias SQL 1071

B

base de datos, autorizaciones
otorgar
GRANT (autorizaciones de bases de datos), sentencia 789
bases de datos
acceso
otorgamiento de autoridad 789
CREATE TABLESPACE, sentencia 558
BEGIN DECLARE SECTION, sentencia
autorización necesaria 204
descripción 204
normas de invocación 204
BIGINT, tipo de datos
CREATE TABLE, sentencia 484
BINDADD, parámetro
otorgar privilegio 789
BLOB (objetos grandes binarios)
CREATE TABLE, sentencia 484
bloqueos
COMMIT, sentencia 229
durante UPDATE 1048
LOCK TABLE, sentencia 853
restricción del acceso 853
sentencia INSERT, normas por omisión para 839
terminación para unidad de trabajo 940
BUFFERPOOL, cláusula
ALTER TABLESPACE, sentencia 133
CREATE TABLESPACE, sentencia 558
DROP, sentencia 723

C

CALL, sentencia
descripción 206
cargar
otorgar autorización de base de datos 789
CASCADE, norma de supresión 484
CASE, sentencia 212
catálogos
COMMENT, sentencia 217
CCSID (identificador de juego de caracteres codificado)
CREATE TABLE, sentencia 484
en sentencia DECLARE GLOBAL TEMPORARY TABLE 695
Centro de información de DB2
actualización 1073
idiomas 1072
versiones 1072
visualización en distintos idiomas 1072
CHAR VARYING, tipo de datos 484
CHARACTER VARYING, tipo de datos 484
CHECK, cláusula
CREATE VIEW, sentencia 651
cláusula de contenedor
CREATE TABLESPACE, sentencia 558
cláusula-en-partic-bd
CREATE TABLESPACE, sentencia 558
cláusula MANAGED BY
CREATE TABLESPACE, sentencia 558
claves de particionamiento
añadir 85
definición al crear tablas 484
descartar 85
claves exclusivas
ALTER TABLE, sentencia 85
CREATE TABLE, sentencia 484
claves foráneas
adición con ALTER TABLE 85
eliminación con ALTER TABLE 85
nombre de restricción, convenios para 484
claves primarias
añadir
ALTER TABLE, sentencia 85
CREATE TABLE, sentencia 484
descartar 85
privilegios necesarios 825
CLOB (objetos grandes de caracteres)
tipo de datos
creación de columnas 484
CLOSE, sentencia 215
CLOSE en sentencia CREATE INDEX 385
CLUSTER, cláusula
CREATE INDEX, sentencia 385
códigos de retorno
sentencias de SQL ejecutables 11
sentencias incorporadas 11
códigos de retorno de SQL 11
COLLID
en sentencia CREATE FUNCTION (escalar externa) 300
en sentencia CREATE FUNCTION (tabla externa) 326
en sentencia CREATE PROCEDURE 430, 450
COLUMN, cláusula
COMMENT, sentencia 217
columnas
actualización 1048
adición con sentencia ALTER TABLE 85
adición de comentarios al catálogo 217
crear claves de índice 385
insertar valores 839
nombre de restricción
normas FOREIGN KEY 484
nombres
INSERT, sentencia 839
otorgar privilegios de adición 825
valores nulos
en sentencia ALTER TABLE, prevención 85
columnas de identidad
CREATE TABLE, sentencia 484
columnas generadas
CREATE TABLE, sentencia 484
columnas XML
CREATE INDEX, sentencia 385
coma flotante de precisión simple, tipo de datos 484
comentarios
en tabla del catálogo 217
estáticas, sentencias de SQL 11
COMMENT, sentencia 217
COMMIT, sentencia
descripción 229

COMMIT ON RETURN
en sentencia CREATE PROCEDURE 430, 450
condiciones de búsqueda
con DELETE
selección de filas 703
con UPDATE
argumentos y normas 1048
conexiones implícitas
sentencia CONNECT 249
conjuntos de resultados
devolución
procedimientos SQL 240
CONNECT, parámetro
sentencia GRANT...ON DATABASE 789
CONNECT, sentencia
conexión implícita 249
desconexión del servidor actual 249
información de nueva contraseña 249
información del servidor de aplicaciones 249
sin operandos, devolución de información 249
tipo 2 256
CONNECT TO, sentencia
conexión no satisfactoria 249, 256
conexión satisfactoria 249, 256
CONSTRAINT, cláusula 217
contenedores
CREATE TABLESPACE, sentencia 558
contenedores-sistema
CREATE TABLESPACE, sentencia 558
CONTINUE, cláusula
WHENEVER, sentencia 1062
CONTROL, cláusula
revocar 931
sentencia GRANT (tabla, vista o apodo) 825
CONTROL, parámetro
revocar privilegios para paquetes 909
control de acceso basado en etiquetas (LBAC)
ALTER SECURITY LABEL COMPONENT, sentencia 63
ALTER SECURITY POLICY, sentencia 66
CREATE SECURITY LABEL, sentencia 463
CREATE SECURITY LABEL COMPONENT, sentencia 460
CREATE SECURITY POLICY, sentencia 465
GRANT (etiqueta de seguridad), sentencia 813
REVOKE (etiqueta de seguridad), sentencia 921
REVOKE (exención), sentencia 902
sentencia GRANT (exención) 793
control de simultaneidad
LOCK TABLE, sentencia 853
conversiones
serie de caracteres a SQL ejecutable 757
COPY
CREATE INDEX, sentencia 385
correlaciones de particiones
creación de grupos de particiones de base de datos 275
creación
bases de datos, otorgar autorización 789
crear sentencia de SQL sentencia-de-selección
definición 11
invocación dinámica 11
invocación estática 11
CREATE ALIAS, sentencia
descripción 264
CREATE AUDIT POLICY, sentencia 267
CREATE BUFFERPOOL, sentencia
cláusula-excepto-en-particiones-base-de-datos 271
descripción 271
CREATE DATABASE PARTITION GROUP, sentencias 275
CREATE DISTINCT TYPE, sentencia
consulte sentencia CREATE TYPE (diferenciado) 608
CREATE EVENT MONITOR, sentencia 278
CREATE FUNCTION (con fuente o plantilla), sentencia 355
CREATE FUNCTION (escalar externa), sentencia 300
CREATE FUNCTION (fuente), sentencia 355
CREATE FUNCTION, sentencia
descripción 299
escalar, tabla o fila de SQL 367
OLE, tabla externa 346
tabla externa 326
CREATE FUNCTION (SQL, escalar, de tabla o de fila), sentencia 367
CREATE FUNCTION (tabla externa), sentencia 326
CREATE FUNCTION (tabla externa OLE DB), sentencia 346
CREATE FUNCTION MAPPING, sentencia
descripción 378
CREATE HISTOGRAM TEMPLATE, sentencia 383
CREATE INDEX, sentencia
columna XML 385
descripción 385
nombres-columna en claves de índice 385
CREATE INDEX EXTENSION, sentencia 403
CREATE METHOD, sentencia
descripción 410
CREATE NICKNAME, sentencia
descripción 416
CREATE NODEGROUP, sentencia
CREATE DATABASE PARTITION GROUP, sentencias 275
CREATE PROCEDURE (con fuente), sentencia 445
CREATE PROCEDURE (externo), sentencia 430
CREATE PROCEDURE, sentencia
CASE, sentencia 212
DECLARE, sentencia 240
descripción 429
dinámica, sentencia compuesta 231
FOR, sentencia 780
gestores de condiciones 240
GET DIAGNOSTICS, sentencia 784
GOTO, sentencia 787
IF, sentencia 835
ITERATE, sentencia 849
LEAVE, sentencia 851
LOOP, sentencia 855
procedimiento, sentencia compuesta 240
REPEAT, sentencia 892
RESIGNAL, sentencia 894
RETURN, sentencia 896
sentencia de gestor de condiciones 240
SIGNAL, sentencia 1030
variables 240
WHILE, sentencia 1064
CREATE PROCEDURE (SQL), sentencia 450
CREATE ROLE, sentencia 456
CREATE SCHEMA, sentencia 457
CREATE SECURITY LABEL, sentencia 463
CREATE SECURITY LABEL COMPONENT, sentencia 460
CREATE SECURITY POLICY, sentencia 465
CREATE SEQUENCE, sentencia
descripción 467
CREATE SERVER, sentencia
descripción 480
CREATE SERVICE CLASS, sentencia 472
CREATE TABLE, sentencia
diagrama de sintaxis 484
CREATE TABLESPACE, sentencia
descripción 558

CREATE THRESHOLD, sentencia 573
CREATE TRANSFORM, sentencia 582
CREATE TRIGGER, sentencia
descripción 586
CREATE TRUSTED CONTEXT, sentencia 599
CREATE TYPE (diferenciado), sentencia 608
CREATE TYPE (estructurado), sentencia 614
CREATE TYPE (matriz), sentencia 606
CREATE TYPE MAPPING, sentencia
descripción 639
CREATE USER MAPPING, sentencia
descripción 646
CREATE VARIABLE, sentencia 648
CREATE VIEW, sentencia
descripción 651
CREATE WORK ACTION SET, sentencia 666
CREATE WORK CLASS SET, sentencia 674
CREATE WORKLOAD, sentencia 679
CREATE WRAPPER, sentencia
descripción 687
CREATETAB, parámetro
sentencia GRANT...ON DATABASE 789
CURRENT DECFLOAT ROUNDING MODE, registro especial
SET CURRENT DECFLOAT ROUNDING MODE, sentencia 952
CURRENT DEGREE, registro especial
SET CURRENT DEGREE, sentencia 956
CURRENT EXPLAIN MODE, registro especial
SET CURRENT EXPLAIN MODE, sentencia 958
CURRENT EXPLAIN SNAPSHOT, registro especial
SET CURRENT EXPLAIN SNAPSHOT, sentencia 961
CURRENT FUNCTION PATH, registro especial
SET CURRENT FUNCTION PATH, sentencia 1013
SET CURRENT PATH, sentencia 1013
SET PATH, sentencia 1013
CURRENT IMPLICIT XMLPARSE OPTION, registro especial
SET CURRENT IMPLICIT XMLPARSE OPTION, sentencia 966
CURRENT ISOLATION, registro especial
SET CURRENT ISOLATION, sentencia 967
CURRENT OPTIMIZATION PROFILE, registro especial
sentencia SET CURRENT OPTIMIZATION PROFILE 974
CURRENT PATH, registro especial
SET CURRENT FUNCTION PATH, sentencia 1013
SET CURRENT PATH, sentencia 1013
SET PATH, sentencia 1013
CURRENT QUERY OPTIMIZATION, registro especial
SET CURRENT QUERY OPTIMIZATION, sentencia 983
CURRENT REFRESH AGE, registro especial
SET CURRENT REFRESH AGE, sentencia 986
CURSOR FOR RESULT SET, variable 23
cursores
ambiguo 689
apertura 867
conjunto activo, asociación 867
declarar
sintaxis de sentencia de SQL 689
definición 689
determinación de la capacidad de actualización 689
estado cerrado, condiciones previas 867
fila actual 773
movimiento de la posición, utilización de FETCH 773
posiciones de abierto 773
preparación para que lo utilice la aplicación 867
relación con tabla de resultados 689
sólo lectura
condiciones 689
cursores *(continuación)*
supresión, detalles de condición de búsqueda 703
terminación para unidad de trabajo 940
ubicación en la tabla, resultados de FETCH 773
unidad de trabajo
estados condicionales 689
utilización del programa 689
WITH HOLD
cláusula de bloqueo de la sentencia COMMIT, efecto 229
cursores ambiguos 689
cursores de sólo lectura
ambiguo 689

D

datos
integridad
bloqueos 853
datos XML
CREATE INDEX, sentencia 385
db2nodes.cfg, archivo
ALTER DATABASE PARTITION GROUP, sentencia 32
CREATE DATABASE PARTITION GROUP, sentencias 275
sentencia CONNECT (tipo 1) 249
DBADM (administrador de la base de datos), autorización
otorgar 789
DBCLOB, tipo de datos
CREATE TABLE, sentencia 484
declaraciones
inserción en programa 837
DECLARE, sentencias
BEGIN DECLARE SECTION, sentencia 204
END DECLARE SECTION, sentencia 756
SQL compuesto 240
DECLARE CURSOR, sentencia
descripción 689
DECLARE GLOBAL TEMPORARY TABLE, sentencia
descripción 695
definidos por el usuario, tipos
adición de comentarios al catálogo 217
CREATE TRANSFORM, sentencia 582
sentencia CREATE TYPE (diferenciado) 608
tipos de datos diferenciados
CREATE TABLE, sentencia 484
tipos estructurados 484
DELETE, sentencia
descripción 703
DESCRIBE, sentencia
descripción 710
sentencias preparadas
DESCRIBE INPUT, sentencia 711
DESCRIBE OUTPUT, sentencia 715
visión general 710
DESCRIBE INPUT, sentencia 711
DESCRIBE OUTPUT, sentencia 715
detención, valor de clave 403
determinación de problemas
guías de aprendizaje 1075
información disponible 1075
devolución de conjuntos de resultados
de procedimientos SQL 240
diferenciados, tipos
DROP, sentencia 723
sentencia CREATE TYPE (diferenciado) 608
DISCONNECT, sentencia 720

documentación
copia impresa 1068
PDF 1068
términos y condiciones de uso 1075
visión general 1067
DROP, sentencia
descripción 723
transformaciones 723

E

elemento de sintaxis designador de función 18
elemento de sintaxis designador de método 18
elemento de sintaxis designador de procedimiento 18
END DECLARE SECTION, sentencia 756
errores
cursor 867
espacio gestionado por base de datos (DMS)
espacios de tablas
CREATE TABLESPACE, sentencia 558
espacios de tablas
agrupaciones de almacenamientos intermedios 271
añadir
comentarios al catálogo 217
creación
CREATE TABLESPACE, sentencia 558
descartar
DROP, sentencia 723
identificación
CREATE TABLE, sentencia 484
índice
CREATE TABLE, sentencia 484
otorgar privilegios 823
renombrar 888
revocar privilegios 929
suprimir utilizando la sentencia DROP 723
tamaño de página 558
esquemas
adición de comentarios al catálogo 217
CREATE SCHEMA, sentencia 457
implícitos
autorización, revocar 898
otorgamiento de autoridad 789
esquemas implícitos
GRANT (autorizaciones de bases de datos), sentencia 789
REVOKE (autorizaciones de bases de datos), sentencia 898
establecer estado pendiente de integridad 991
estado cerrado
cursores 867
estándares
establecer normas para SQL dinámico 1016
estructura SQLCA
visión general 11
estructurados, tipos
CREATE TRANSFORM, sentencia 582
DROP, sentencia 723
estructuras de almacenamiento
ALTER BUFFERPOOL, sentencia 29
ALTER TABLESPACE, sentencia 133
CREATE BUFFERPOOL, sentencia 271
CREATE TABLESPACE, sentencia 558
etiquetas
GOTO 787
etiquetas de seguridad (LBAC)
ALTER SECURITY LABEL COMPONENT, sentencia 63
CREATE SECURITY LABEL, sentencia 463
etiquetas de seguridad (LBAC) *(continuación)*
CREATE SECURITY LABEL COMPONENT, sentencia 460
GRANT (etiqueta de seguridad), sentencia 813
políticas
ALTER SECURITY POLICY, sentencia 66
CREATE SECURITY POLICY, sentencia 465
REVOKE (etiqueta de seguridad), sentencia 921
excepciones, tablas
SET INTEGRITY, sentencia 991
EXCLUSIVE MODE, conexión 249
EXECUTE, sentencia
descripción 760
incorporado 11
EXECUTE IMMEDIATE, sentencia
descripción 757
incorporado 11
EXPLAIN, sentencia
descripción 767

F

FETCH, sentencia
descripción 773
requisitos previos del cursor para la ejecución 773
FIELDPROC, cláusula
en sentencia ALTER TABLE 85
filas
actualización de los valores de columna, sentencia UPDATE 1048
asignación de valores a la variable del lenguaje principal, SELECT INTO 946
asignación de valores a la variable del lenguaje principal, VALUES INTO 1059
bloqueos, efecto sobre el cursor de WITH HOLD 689
bloqueos en datos de filas, sentencia INSERT 839
claves de índice con cláusula UNIQUE 385
cursor, efecto de cierre en FETCH 215
cursor, ubicación en la tabla de resultados 689
cursor en sentencia FETCH 867
índices 385
insertar 839
otorgar privilegio 825
petición FETCH, selección de fila de cursor 689
restricciones que conducen a anomalías 839
suprimir 703
finalización
unidad de trabajo 229, 940
FLOAT, tipo de datos 484
FLUSH EVENT MONITOR, sentencia 776
FLUSH OPTIMIZATION PROFILE CACHE, sentencia 777
FLUSH PACKAGE CACHE, sentencia 779
FOR, cláusula
CREATE TABLE, sentencia 484
FOR, sentencia 780
FOR BIT DATA, cláusula
CREATE TABLE, sentencia 484
FOREIGN KEY, cláusula 484
FREE LOCATOR, sentencia 783
FREEPAGE en sentencia CREATE INDEX 385
FROM, cláusula
DELETE, sentencia 703
funciones
adición de comentarios al catálogo 217
transformación 582
funciones definidas por el usuario
CREATE FUNCTION, sentencia
con fuente o plantilla 355

funciones definidas por el usuario *(continuación)*
CREATE FUNCTION, sentencia *(continuación)*
descripción 299
escalar de SQL, tabla o fila 367
Escalar externa 300
tabla externa 326
tabla externa OLE DB 346
DROP, sentencia 723
REVOKE (autorizaciones de bases de datos), sentencia 898
FUNCTION, cláusula en la sentencia COMMENT ON 217

G

GBPCACHE
en sentencia CREATE INDEX 385
generación aleatoria en claves de particionamiento 484
gestores de condiciones
declarar 240
GET DIAGNOSTICS, sentencia 784
GO TO, cláusula
WHENEVER, sentencia 1062
GOTO, sentencia 787
GRANT (etiqueta de seguridad), sentencia 813
GRANT (exención), sentencia 793
GRANT (función), sentencia 803
GRANT (privilegio SETSESSIONUSER), sentencia 821
GRANT (privilegios de carga de trabajo), sentencia 832
GRANT (privilegios de esquema), sentencia
descripción 810
GRANT (privilegios de objeto XSR), sentencia 834
GRANT (privilegios de paquete), sentencia 800
GRANT (privilegios de rutina), sentencia
descripción 806
GRANT (privilegios de secuencia), sentencia
descripción 816
GRANT (privilegios de servidor), sentencia
descripción 819
GRANT (privilegios de variable global), sentencia 796
GRANT (privilegios para espacios de tablas), sentencia
descripción 823
GRANT, sentencia
autorización de base de datos
descripción 789
CONTROL ON INDEX
descripción 798
CREATE ON SCHEMA 810
privilegios de apodo, vista o tabla
descripción 825
privilegios de paquete
descripción 800
privilegios para apodos 825
privilegios para tablas 825
privilegios para vistas 825
GRAPHIC, tipo de datos
CREATE TABLE, sentencia 484
grupos de particiones de base de datos
adición de comentarios al catálogo 217
adición de particiones 32
creación 275
creación de correlación de distribución 275
descartar particiones 32
guías de aprendizaje
determinación de problemas 1075
resolución de problemas 1075
Visual Explain 1074

I

identificador de objeto (OID)
columnas
visión general 484
CREATE TABLE, sentencia 484
CREATE VIEW, sentencia 651
IF, sentencia 835
IN
CREATE TABLE, sentencia 484
IN EXCLUSIVE MODE, cláusula
LOCK TABLE, sentencia 853
IN SHARE MODE, cláusula
LOCK TABLE, sentencia 853
INCLUDE, cláusula
CREATE INDEX, sentencia 385
INCLUDE, sentencia 837
INDEX, cláusula
COMMENT, sentencia 217
CREATE INDEX, sentencia 385
sentencia GRANT (tabla, vista o apodo) 825
sentencia REVOKE, eliminar privilegios 931
INDEX, palabra clave
DROP, sentencia 723
índice sobre datos XML
CREATE INDEX, sentencia 385
Sintaxis y descripciones de opciones 385
índices
clave exclusiva, utilización en coincidencias 85
clave primaria, utilización en coincidencia 85
comentarios de especificación de catálogo, adición 217
correspondencia con los valores de fila insertados 839
nombre
restricción de clave primaria 484
restricción de unicidad 484
otorgar control 798, 825
privilegios
revocar 907
renombrar 890
suprimir
utilizando la sentencia DROP 723
índices de tipo 2 385
inicio, valor de clave 403
inserción en almacenamiento intermedio 839
INSERT
insertar valores 839
restricciones que conducen a anomalías 839
INSERT, cláusula
sentencia GRANT (tabla, vista o apodo) 825
sentencia REVOKE, eliminar privilegios 931
INSERT, sentencia
descripción 839
insertables, vistas 651
INTEGER, tipo de datos 484
INTO, cláusula
FETCH, sentencia de sustitución de variables del lenguaje principal 773
INSERT, sentencia, denominación de tabla o vista 839
restricciones en la utilización 839
SELECT INTO, sentencia 946
sentencia DESCRIBE, nombre de área SQLDA 711, 715
VALUES INTO, sentencia 1059
IS, cláusula
COMMENT, sentencia 217
ITERATE, sentencia 849

L

LBAC (control de acceso basado en etiquetas)
  ALTER SECURITY LABEL COMPONENT, sentencia 63
  ALTER SECURITY POLICY, sentencia 66
  componentes de etiqueta de seguridad
    ALTER SECURITY LABEL COMPONENT, sentencia 63
    CREATE SECURITY LABEL COMPONENT, sentencia 460
  CREATE SECURITY LABEL, sentencia 463
  CREATE SECURITY LABEL COMPONENT, sentencia 460
  CREATE SECURITY POLICY, sentencia 465
  etiquetas de seguridad
    ALTER SECURITY LABEL COMPONENT, sentencia 63
    CREATE SECURITY LABEL, sentencia 463
    CREATE SECURITY LABEL COMPONENT, sentencia 460
    GRANT (etiqueta de seguridad), sentencia 813
    REVOKE (etiqueta de seguridad), sentencia 921
  exenciones de normas
    REVOKE (exención), sentencia 902
    sentencia GRANT (exención) 793
  políticas de seguridad
    ALTER SECURITY POLICY, sentencia 66
    CREATE SECURITY POLICY, sentencia 465
  REVOKE (exención), sentencia 902
  sentencia GRANT (etiqueta de seguridad) 813
  sentencia GRANT (exención) 793
  sentencia REVOKE (etiqueta de seguridad) 921
LEAVE, sentencia 851
lenguaje de consulta estructurada (SQL)
  sentencias de control 15
lenguaje REXX
  END DECLARE SECTION, prohibición 756
LOAD, parámetro
  sentencia GRANT...ON DATABASE 789
localizadores
  ASSOCIATE LOCATORS, sentencia 198
  FREE LOCATOR, sentencia 783
LOCK TABLE, sentencia
  descripción 853
LOOP, sentencia
  conceptos básicos de base de datos 855

M

manuales
  copia impresa
    pedido 1070
marcadores de parámetro
  con tipo 872
  EXECUTE, sentencia 760
  normas de contraseña 872
  OPEN, sentencia 867
  PREPARE, sentencia 872
  sin tipo 872
mensajes de error
  códigos de retorno 11
  ejecución de activador 586
  FETCH, sentencia 773
  UPDATE, sentencia 1048
MERGE, sentencia
  descripción 857
METHOD, cláusula
  DROP, sentencia 723
MODE, palabra clave
  LOCK TABLE, sentencia 853
MQT (tablas de consultas materializadas)
  definición 484
  REFRESH TABLE, sentencia 882

N

NICKNAME, cláusula en sentencias DROP 723
niveles de aislamiento
  en sentencia DELETE 703, 839, 946, 1048
NO ACTION, norma de supresión 484
no operativos
  activadores 586
  vistas 651
nombre de cursor
  ALLOCATE 23
  cierre, sentencia CLOSE 215
nombre de tabla
  CREATE TABLE, sentencia 484
nombre de vista
  en sentencia ALTER VIEW 169
nombres
  supresión de filas 703
norma de supresión RESTRICT 484
NOT FOUND, cláusula
  WHENEVER, sentencia 1062
NOT NULL, cláusula
  en sentencia CREATE TABLE 484
NULL CALL
  en sentencia CREATE TYPE (estructurado) 614

O

objetos dependientes
  DROP, sentencia 723
objetos grandes binarios (BLOB) 484
objetos SQL
  suprimir 723
OF, cláusula
  CREATE VIEW, sentencia 651
OID
  véase identificador de objeto (OID) 484
ON, cláusula
  CREATE INDEX, sentencia 385
ON TABLE, cláusula
  GRANT, sentencia 825
  REVOKE, sentencia 931
ON UPDATE, cláusula 484
ONLY, cláusula
  DELETE, sentencia 703
  UPDATE, sentencia 1048
opciones de columna
  CREATE TABLE, sentencia 484
OPEN, sentencia 867
OPTION, cláusula
  CREATE VIEW, sentencia 651

P

paquetes
  ALTER TABLE, SENTENCIA 85
  autorización para crear
    otorgar 789
  comentarios de catálogo 217
  COMMIT, efecto de la sentencia en los cursores 229

paquetes *(continuación)*
privilegios
otorgar 800
revocación utilizando la sentencia REVOKE (privilegios de paquete) 909
revocación utilizando la sentencia REVOKE (privilegios de tabla, vista o apodo) 931
suprimir 723
pedido de manuales de DB2 1070
PIECESIZE
CREATE INDEX, sentencia 385
plantillas de función
descripción 378
precompilación
archivo de texto externo 837
INCLUDE, sentencia 837
sentencias de SQL no ejecutables 11
SQLCA 837
SQLDA 837
PREPARE, sentencia
declarar dinámicamente 872
descripción 872
incorporado 11
sustitución de variable en sentencia OPEN 867
privilegio SETSESSIONUSER
necesario para la sentencia SET SESSION AUTHORIZATION 1020
sentencia GRANT (privilegio SETSESSIONUSER) 821
sentencia REVOKE (privilegio SETSESSIONUSER) 927
privilegios
base de datos
revocar 919
INDEX
revocar 907
paquetes
revocar 909, 931
revocar
REVOKE, sentencia 931
procedimiento, sentencia compuesta 240
procedimientos
autorización para crear
CREATE PROCEDURE (externo), sentencia 430
CREATE PROCEDURE (SQL), sentencia 450
CALL, sentencia 206
creación 430, 450
procedimientos almacenados
CREATE PROCEDURE, sentencia 429
procedimientos SQL
CASE, sentencia 212
DECLARE, sentencia 231, 240
dinámica, sentencia compuesta 231
FOR, sentencia 780
gestores de condiciones
declaración 240
GET DIAGNOSTICS, sentencia 784
GOTO, sentencia 787
IF, sentencia 835
ITERATE, sentencia 849
LEAVE, sentencia 851
LOOP, sentencia 855
procedimiento, sentencia compuesta 240
REPEAT, sentencia 892
RESIGNAL, sentencia 894
RETURN, sentencia 896
SIGNAL, sentencia 1030
variables 231, 240
WHILE, sentencia 1064
PROGRAM
DROP, sentencia 723
PROGRAM TYPE
CREATE FUNCTION (escalar externa), sentencia 300
CREATE FUNCTION (tabla externa), sentencia 326
PUBLIC AT ALL LOCATIONS 825
puntos de salvaguarda
liberar 887
ROLLBACK TO SAVEPOINT 940

R

REAL SQL, tipo de datos
CREATE TABLE, sentencia 484
REFERENCES, cláusula
sentencia GRANT (tabla, vista o apodo) 825
sentencia REVOKE, eliminar privilegios 931
REFRESH TABLE, sentencia
descripción 882
REFRESH DEFERRED 882
REFRESH IMMEDIATE 882
registros
bloqueos de datos de fila 839
RELEASE (conexión), sentencia 885
RELEASE SAVEPOINT, sentencia 887
RENAME, sentencia 890
RENAME TABLESPACE, sentencia 888
rendimiento
recomendación de clave de particionamiento 484
REPEAT, sentencia 892
RESIGNAL, sentencia 894
resolución de problemas
guías de aprendizaje 1075
información en línea 1075
restricciones
adición con ALTER TABLE 85
adición de comentarios al catálogo 217
descartar
con ALTER TABLE 85
restricciones de comprobación
ALTER TABLE, sentencia 85
CREATE TABLE, sentencia 484
INSERT, sentencia 839
restricciones de integridad
adición de comentarios al catálogo 217
restricciones de referencia
adición de comentarios al catálogo 217
restricciones de unicidad
adición con ALTER TABLE 85
ALTER TABLE, sentencia 85
CREATE TABLE, sentencia 484
eliminación con ALTER TABLE 85
RESULTSTATUS, parámetro 784
RETURN, sentencia 896
REVOKE (etiqueta de seguridad), sentencia 921
REVOKE (exención), sentencia 902
REVOKE (función), sentencia 912
REVOKE (privilegio SETSESSIONUSER), sentencia 927
REVOKE (privilegios de carga de trabajo), sentencia 937
REVOKE (privilegios de objeto XSR), sentencia 939
REVOKE (privilegios de paquete), sentencia 909
REVOKE (privilegios de secuencia), sentencia
descripción 923
REVOKE (privilegios de variable global) 904
REVOKE, sentencia
apodo, privilegios 931
base de datos, autorizaciones 898

REVOKE, sentencia *(continuación)*
privilegios de espacio de tablas 929
privilegios de esquema 919
privilegios de índice 907
privilegios de paquete 909
privilegios de rutina 915
privilegios de servidor 925
privilegios de tabla 931
vista, privilegios 931
ROLLBACK, sentencia
descripción 940
sintaxis 940
ROLLBACK TO SAVEPOINT, sentencia
descripción 940
ROWCOUNT
GET DIAGNOSTICS, sentencia 784

S

SAVEPOINT, sentencia
descripción 943
SCHEMA, cláusula
COMMENT, sentencia 217
DROP, sentencia 723
SCOPE, cláusula
ALTER TABLE, sentencia 85
ALTER VIEW, sentencia 169
CREATE TABLE, sentencia 484
CREATE VIEW, sentencia 651
SECADM
autorización de base de datos
GRANT (autorizaciones de bases de datos), sentencia 789
revocar 898
parámetro
GRANT (autorizaciones de bases de datos), sentencia 789
REVOKE (autorizaciones de bases de datos), sentencia 898
secuencias
DROP, sentencia 723
SECURED WITH, cláusula
ALTER TABLE, sentencia 85
CREATE TABLE, sentencia 484
SECURITY LABEL, cláusula
COMMENT, sentencia 217
DROP, sentencia 723
SECURITY LABEL COMPONENT, cláusula
COMMENT, sentencia 217
DROP, sentencia 723
SECURITY POLICY, cláusula
COMMENT, sentencia 217
CREATE TABLE, sentencia 484
DROP, sentencia 723
seguridad
sentencia CONNECT 249
selección completa
CREATE VIEW, sentencia 651
selección completa de fila
UPDATE, sentencia 1048
selección de una sola fila 946
SELECT, cláusula
sentencia GRANT (tabla, vista o apodo) 825
sentencia REVOKE, eliminar privilegios 931
SELECT, sentencia
cursor
normas para marcadores de parámetro 689
SELECT, sentencia *(continuación)*
descripción 948
evaluar
para tabla de resultados del cursor de sentencia OPEN 867
SELECT INTO, sentencia
descripción 946
sentencias
ALTER WRAPPER 195
LEAVE 851
series
creación 757
PREPARE, sentencia 872
SET CURRENT LOCK TIMEOUT 968
SET CURRENT PACKAGE PATH 977
sentencias de SQL
ALLOCATE CURSOR 23
ALTER AUDIT POLICY 25
ALTER BUFFERPOOL 29
ALTER DATABASE 36
ALTER DATABASE PARTITION GROUP 32
ALTER FUNCTION 40
ALTER HISTOGRAM TEMPLATE 43
ALTER METHOD 45
ALTER NICKNAME 47
ALTER NODEGROUP (ver ALTER DATABASE PARTITION GROUP) 32
ALTER PROCEDURE (con fuente) 59
ALTER PROCEDURE (externo) 56
ALTER PROCEDURE (SQL) 61
ALTER SECURITY LABEL COMPONENT 63
ALTER SECURITY POLICY 66
ALTER SEQUENCE 70
ALTER SERVER 74
ALTER SERVICE CLASS 78
ALTER TABLE 85
ALTER TABLESPACE 133
ALTER THRESHOLD 144
ALTER TRUSTED CONTEXT 151
ALTER TYPE (estructurado) 160
ALTER USER MAPPING 167
ALTER VIEW 169
ALTER WORK ACTION SET 171
ALTER WORK CLASS SET 183
ALTER WORKLOAD 188
ALTER XSROBJECT 197
ASSOCIATE LOCATORS 198
AUDIT 200
BEGIN DECLARE SECTION 204
CALL 206
CLOSE 215
COMMENT 217
COMMIT 229
CONNECT (Tipo 1) 249
CONNECT (Tipo 2) 256
CONTINUE, respuesta a excepción 1062
control 15
CREATE ALIAS 264
CREATE AUDIT POLICY 267
CREATE BUFFERPOOL 271
CREATE DATABASE PARTITION GROUP 275
CREATE EVENT MONITOR 278
CREATE FUNCTION (con fuente o plantilla) 355
CREATE FUNCTION (escalar de SQL, tabla o fila) 367
CREATE FUNCTION (escalar externa) 300
CREATE FUNCTION (fuente) 355
CREATE FUNCTION (tabla externa) 326

sentencias de SQL *(continuación)*
CREATE FUNCTION (tabla externa OLE DB) 346
CREATE FUNCTION, visión general 299
CREATE FUNCTION MAPPING 378
CREATE HISTOGRAM TEMPLATE 383
CREATE INDEX 385
CREATE INDEX EXTENSION 403
CREATE METHOD 410
CREATE NICKNAME 416
CREATE NODEGROUP (ver CREATE DATABASE PARTITION GROUP) 275
CREATE PROCEDURE 429
CREATE PROCEDURE (con fuente) 445
CREATE PROCEDURE (externo) 430
CREATE PROCEDURE (SQL) 450
CREATE ROLE 456
CREATE SCHEMA 457
CREATE SECURITY LABEL 463
CREATE SECURITY LABEL COMPONENT 460
CREATE SECURITY POLICY 465
CREATE SEQUENCE 467
CREATE SERVER 480
CREATE SERVICE CLASS 472
CREATE TABLE 484
CREATE TABLESPACE 558
CREATE THRESHOLD 573
CREATE TRANSFORM 582
CREATE TRIGGER 586
CREATE TRUSTED CONTEXT 599
CREATE TYPE (diferenciado) 608
CREATE TYPE (estructurado) 614
CREATE TYPE (matriz) 606
CREATE TYPE MAPPING 639
CREATE USER MAPPING 646
CREATE VARIABLE 648
CREATE VIEW 651
CREATE WORK ACTION SET 666
CREATE WORK CLASS SET 674
CREATE WORKLOAD 679
CREATE WRAPPER 687
DECLARE CURSOR 689
DECLARE GLOBAL TEMPORARY TABLE 695
DELETE 703
DESCRIBE 710
DESCRIBE INPUT 711
DESCRIBE OUTPUT 715
DISCONNECT 720
DROP 723
DROP TRANSFORM 723
END DECLARE SECTION 756
entrada interactiva 11
EXECUTE 760
EXECUTE IMMEDIATE 757
EXPLAIN 767
FETCH 773
FLUSH EVENT MONITOR 776
FLUSH OPTIMIZATION PROFILE CACHE 777
FLUSH PACKAGE CACHE 779
FREE LOCATOR 783
GRANT (autorizaciones de bases de datos) 789
GRANT (etiqueta de seguridad) 813
GRANT (exención) 793
GRANT (función) 803
GRANT (privilegio SETSESSIONUSER) 821
GRANT (privilegios de apodo) 825
GRANT (privilegios de carga de trabajo) 832
GRANT (privilegios de espacio de tablas) 823
sentencias de SQL *(continuación)*
GRANT (privilegios de esquema) 810
GRANT (privilegios de índice) 798
GRANT (privilegios de objeto XSR) 834
GRANT (privilegios de paquete) 800
GRANT (privilegios de rutina) 806
GRANT (privilegios de secuencia) 816
GRANT (privilegios de servidor) 819
GRANT (privilegios de tabla) 825
GRANT (privilegios de variable global) 796
GRANT (privilegios de vista) 825
INCLUDE 837
incorporado 11
INSERT 839
invocación 11
LOCK TABLE 853
MERGE 857
OPEN 867
PREPARE 872
REFRESH TABLE 882
RELEASE (conexión) 885
RELEASE SAVEPOINT 887
RENAME 890
RENAME TABLESPACE 888
REVOKE (autorizaciones de bases de datos) 898
REVOKE (etiqueta de seguridad) 921
REVOKE (exención) 902
REVOKE (función) 912
REVOKE (privilegio SETSESSIONUSER) 927
REVOKE (privilegios de apodo) 931
REVOKE (privilegios de carga de trabajo) 937
REVOKE (privilegios de espacio de tablas) 929
REVOKE (privilegios de esquema) 919
REVOKE (privilegios de índice) 907
REVOKE (privilegios de objeto XSR) 939
REVOKE (privilegios de paquete) 909
REVOKE (privilegios de rutina) 915
REVOKE (privilegios de secuencia) 923
REVOKE (privilegios de servidor) 925
REVOKE (privilegios de tabla) 931
REVOKE (privilegios de variable global) 904
REVOKE (privilegios de vista) 931
ROLLBACK 940
ROLLBACK TO SAVEPOINT 940
SAVEPOINT 943
SELECT 948
SELECT INTO 946
SET COMPILATION ENVIRONMENT 949
SET CONNECTION 950
SET CONSTRAINTS 991
SET CURRENT DECFLOAT ROUNDING MODE 952
SET CURRENT DEFAULT TRANSFORM GROUP 954
SET CURRENT DEGREE 956
SET CURRENT EXPLAIN MODE 958
SET CURRENT EXPLAIN SNAPSHOT 961
SET CURRENT FEDERATED ASYNCHRONY 964
SET CURRENT FUNCTION PATH 1013
SET CURRENT IMPLICIT XMLPARSE OPTION 966
SET CURRENT ISOLATION 967
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 970
SET CURRENT MDC ROLLOUT MODE 972
SET CURRENT OPTIMIZATION PROFILE 974
SET CURRENT PACKAGESET 981
SET CURRENT PATH 1013
SET CURRENT QUERY OPTIMIZATION 983
SET CURRENT REFRESH AGE 986

sentencias de SQL *(continuación)*
SET ENCRYPTION PASSWORD 988
SET EVENT MONITOR STATE 989
SET INTEGRITY 991
SET PASSTHRU 1011
SET PATH 1013
SET ROLE 1015
SET SCHEMA 1016
SET SERVER OPTION 1018
SET SESSION AUTHORIZATION 1020
SET variable 1023
soportadas 2
SQL compuesto (incorporado) 236
TRANSFER OWNERSHIP 1033
UPDATE 1048
VALUES 1061
VALUES INTO 1059
WHENEVER 1062
WITH HOLD
atributo de cursor 689
sentencias de SQL activadas
SET variable 1023
sentencias de SQL ejecutables 11
sentencias de SQL no ejecutables
invocación 11
requisitos del precompilador 11
sentencias de SQL preparadas
ejecución 760
obtención de información
DESCRIBE INPUT, sentencia 711
DESCRIBE OUTPUT, sentencia 715
sustitución de variable del lenguaje principal 760
sentencias de SQL soportadas 2
sentencias explicables
descripción 767
sentencias SQL
visualización de la ayuda 1071
SEQUENCE, cláusula
COMMENT, sentencia 217
series de caracteres
sentencia de SQL, ejecución como 757
serie de sentencia de SQL, normas para la creación 757
servidores
otorgar privilegios 819
SET, cláusula
UPDATE, sentencia 1048
SET, sentencia 1023
SET COMPILATION ENVIRONMENT, sentencia 949
SET CONNECTION, sentencia 950
SET CONSTRAINTS, sentencia 991
SET CURRENT DECFLOAT ROUNDING MODE, sentencia 952
SET CURRENT DEFAULT TRANSFORM GROUP, sentencia 954
SET CURRENT DEGREE, sentencia 956
SET CURRENT EXPLAIN MODE, sentencia 958
SET CURRENT EXPLAIN SNAPSHOT, sentencia 961
SET CURRENT FEDERATED ASYNCHRONY, sentencia 964
SET CURRENT FUNCTION PATH, sentencia 1013
SET CURRENT IMPLICIT XMLPARSE OPTION, sentencia 966
SET CURRENT ISOLATION, sentencia 967
SET CURRENT LOCK TIMEOUT, sentencia 968
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION, sentencia 970
SET CURRENT MDC ROLLOUT MODE, sentencia 972
SET CURRENT OPTIMIZATION PROFILE, sentencia 974
SET CURRENT PACKAGE PATH, sentencia 977
SET CURRENT PACKAGESET, sentencia 981
SET CURRENT PATH, sentencia 1013
SET CURRENT QUERY OPTIMIZATION, sentencia 983
SET CURRENT REFRESH AGE, sentencia 986
SET CURRENT SQLID, sentencia 1016
SET ENCRYPTION PASSWORD, sentencia 988
SET EVENT MONITOR STATE, sentencia 989
SET INTEGRITY, sentencia 991
SET NULL, norma de supresión 484
SET PASSTHRU, sentencia
descripción 1011
independencia de la sentencia COMMIT 229
independencia de la sentencia ROLLBACK 940
SET PATH, sentencia 1013
SET ROLE, sentencia 1015
SET SCHEMA, sentencia 1016
SET SERVER OPTION, sentencia
descripción 1018
independencia de la sentencia COMMIT 229
independencia de la sentencia ROLLBACK 940
SET SESSION AUTHORIZATION, sentencia 1020
SHARE, opción
LOCK TABLE, sentencia 853
SHARE MODE, conexión 249
SIGNAL, sentencia 1030
signo de interrogación
marcador de parámetros EXECUTE 760
signo de interrogación (?) marcador de parámetros EXECUTE 760
sinónimos
DROP ALIAS, sentencia 723
sentencia CREATE ALIAS 264
sintaxis
descripción viii
designador de función 18
designador de método 18
designador de procedimiento 18
elementos 18
elementos comunes 18
SMALLINT, tipo de datos
SQL estático 484
SMS (espacio gestionado por el sistema)
espacios de tablas
CREATE TABLESPACE, sentencia 558
sólo lectura, vistas 651
SPECIFIC FUNCTION, cláusula
COMMENT, sentencia 217
SPECIFIC PROCEDURE, cláusula
COMMENT, sentencia 217
SQL compuesto
dinámico, variables 231
SQL compuesto (incorporado), sentencia
combinación de sentencias en un bloque 236
SQL dinámico
cursores
DECLARE CURSOR, sentencia 11
DESCRIBE INPUT, sentencia
descripción 711
DESCRIBE OUTPUT, sentencia
descripción 715
EXECUTE, sentencia
descripción 760
invocación de sentencias de SQL 11
EXECUTE IMMEDIATE, sentencia
descripción 757

SQL dinámico *(continuación)*
FETCH, sentencia
descripción 773
invocación de sentencias de SQL 11
invocación de sentencias 11
OPEN, sentencia 11
PREPARE, sentencia
descripción 872
invocación de sentencias de SQL 11
utilizando DESCRIBE 711, 715
sentencias compuestas 231
SQL estático
DECLARE CURSOR, sentencia 11
FETCH, sentencia 11
invocación 11
OPEN, sentencia 11
seleccionar 11
sentencias 11
SQL/XML
CREATE INDEX, sentencia 385
SQL92, estándar
normas para SQL dinámico 1016
SQLCA (área de comunicaciones SQL)
entrada cambiada por UPDATE 1048
SQLCA, cláusula
INCLUDE, sentencia 837
SQLCODE
descripción 11
SQLDA (área de descriptores de SQL)
descripciones de variables del lenguaje principal, sentencia OPEN 867
FETCH, sentencia 773
variables necesarias para DESCRIBE 711, 715
SQLDA, cláusula
INCLUDE, sentencia 837
SQLERROR, cláusula
WHENEVER, sentencia 1062
SQLSTATE
descripción 11
SQLWARNING, cláusula de sentencia WHENEVER 1062
STAY RESIDENT
CREATE FUNCTION (escalar externa), sentencia 300
CREATE FUNCTION (tabla externa), sentencia 326
CREATE PROCEDURE, sentencia 430, 450
supervisores de sucesos
CREATE EVENT MONITOR, sentencia 278
DROP, sentencia 723
sentencia FLUSH EVENT MONITOR 776
SET EVENT MONITOR STATE, sentencia 989
suprimibles, vistas
descripción 651
SYNONYM, cláusula
en vez de la cláusula ALIAS 723

T

tablas
actualización por fila y columna, sentencia UPDATE 1048
alias 264, 723
añadir
columnas, ALTER TABLE 85
comentarios al catálogo 217
autorización para crear 484
columnas generadas 85
con tipo, y activadores 586
creación
otorgamiento de autoridad 789
tablas *(continuación)*
creación *(continuación)*
sentencia de SQL, instrucciones 484
esquemas 457
excepciones 991
índices 385
insertar filas 839
modificar 85
nombres
en sentencia ALTER TABLE 85
en sentencia LOCK TABLE 853
otorgar privilegios 825
renombrar 890
restricción del acceso, sentencia LOCK TABLE 853
revocar privilegios 931
suprimir
sentencia DROP, utilización 723
temporales
en sentencia OPEN 867
uniones
consideraciones acerca de claves de particionamiento 484
tablas de consulta materializada (MQT)
definición 484
tablas de consultas materializadas (MQT)
REFRESH TABLE, sentencia 882
tablas de resumen
definición 484
tablas temporales
OPEN, sentencia 867
TABLE, cláusula
COMMENT, sentencia 217
CREATE FUNCTION (tabla externa), sentencia 326
DROP, sentencia 723
TABLE HIERARCHY, cláusula
DROP, sentencia 723
TABLESPACE, cláusula
COMMENT, sentencia 217
términos y condiciones
uso de publicaciones 1075
TIME, tipos de datos
CREATE TABLE, sentencia 484
TIMESTAMP, tipo de datos
CREATE TABLE, sentencia 484
tipo de datos CHARACTER 484
tipo de datos DB2SECURITYLABEL
CREATE TABLE, sentencia 484
tipos de datos
abstracto 160, 614
ALTER TYPE, sentencia 160
CREATE TYPE (estructurado), sentencia 614
definidos por el usuario
CREATE TYPE (diferenciado), sentencia 608
diferenciado
CREATE TYPE (diferenciado), sentencia 608
estructurado
ALTER TYPE (estructurado), sentencia 160
CREATE TYPE (estructurado), sentencia 614
TO, cláusula
GRANT, sentencia
autorización de base de datos 789
privilegios de esquema 810
privilegios de índice 798
privilegios de paquete 800
privilegios de tabla 825
TRANSFER OWNERSHIP, sentencia 1033

transformaciones
DROP, sentencia 723
funciones
CREATE TRANSFORM, sentencia 582
TRIGGER, cláusula
COMMENT, sentencia 217
TYPE, cláusula
COMMENT, sentencia 217
DROP, sentencia 723

U

UNDER, cláusula
CREATE VIEW, sentencia 651
unidades de trabajo (UOW)
cancelar 940
COMMIT, sentencia 229
destrucción de sentencias preparadas 872
iniciación de cursores de cierres 867
referencia a sentencias preparadas 872
sentencia ROLLBACK, efecto 940
terminación 229
terminación destruye las sentencias preparadas 872
terminación sin guardar cambios 940
uniones
CREATE TABLE, sentencia 484
UNIQUE, cláusula
ALTER TABLE, sentencia 85
CREATE INDEX, sentencia 385
CREATE TABLE, sentencia 484
UPDATE, cláusula
sentencia GRANT (tabla, vista o apodo) 825
sentencia REVOKE, eliminar privilegios 931
UPDATE, sentencia
descripción 1048
selección completa de fila 1048
USING, cláusula
CREATE INDEX, sentencia 385
FETCH, sentencia 773
OPEN, listado de variables del lenguaje principal en la sentencia 867
USING DESCRIPTOR, cláusula
OPEN, sentencia 867

V

valores de clave
detención 403
inicio 403
VALUES, cláusula
carga de una fila 839
normas para número de valores 839
VALUES, sentencia 1061
VALUES INTO, sentencia 1059
VARCHAR, tipo de datos
CREATE TABLE, sentencia 484
variables de lenguaje principal
aplicaciones REXX 204
asignación de valores de una fila
SELECT INTO, sentencia 946
BEGIN DECLARE SECTION, sentencia 204
declarar
BEGIN DECLARE SECTION, sentencia 204
cursores 689
END DECLARE SECTION, sentencia 756
END DECLARE SECTION, sentencia 756
variables de lenguaje principal *(continuación)*
enlazar conjunto activo con cursor 867
EXECUTE IMMEDIATE, sentencia 757
inserción en filas 839
series de sentencia 872
sustitución de marcador de parámetros 760
variables del lenguaje principal
asignación de valores de una fila
VALUES INTO, sentencia 1059
FETCH, sentencia 773
sentencias de SQL incorporadas 11
variables del lenguaje principal de la aplicación Assembler 757
variables indicadoras
descripción 757
variables SQL 231, 240
VARIANT
CREATE TYPE (estructurado), sentencia 614
VIEW, cláusula
CREATE VIEW, sentencia 651
DROP, sentencia 723
VIEW HIERARCHY, cláusula
DROP, sentencia 723
vinculación
GRANT, sentencia 800
revocación de todos los privilegios 909
vistas
actualizables 651
actualización de filas por columnas 1048
adición de comentarios al catálogo 217
alias 264, 723
creación 651
esquemas 457
insertables 651
insertar filas en tabla visualizada 839
no operativos 651
nombres de columna 651
otorgar privilegios 825
prevención de la pérdida de definición de vista, WITH CHECK OPTION 1048
privilegio CONTROL
límites en 825
otorgar 825
revocar privilegios 931
sólo lectura 651
suprimibles 651
suprimir utilizando la sentencia DROP 723
WITH CHECK OPTION 1048
vistas con tipo
definir subvistas 651
Visual Explain
guía de aprendizaje 1074

W

WHENEVER, sentencia
cambio del flujo de control 11
descripción 1062
WHERE, cláusula
DELETE, sentencia 703
UPDATE, sentencia 1048
WHILE, sentencia
descripción 1064

X

XML
CREATE INDEX, sentencia 385

SC11-3508-03

Spine information:

DB2 Versión 9.5 para Linux, UNIX y Windows **Versión 9 Release 5** **Consulta de SQL, Volumen 2**